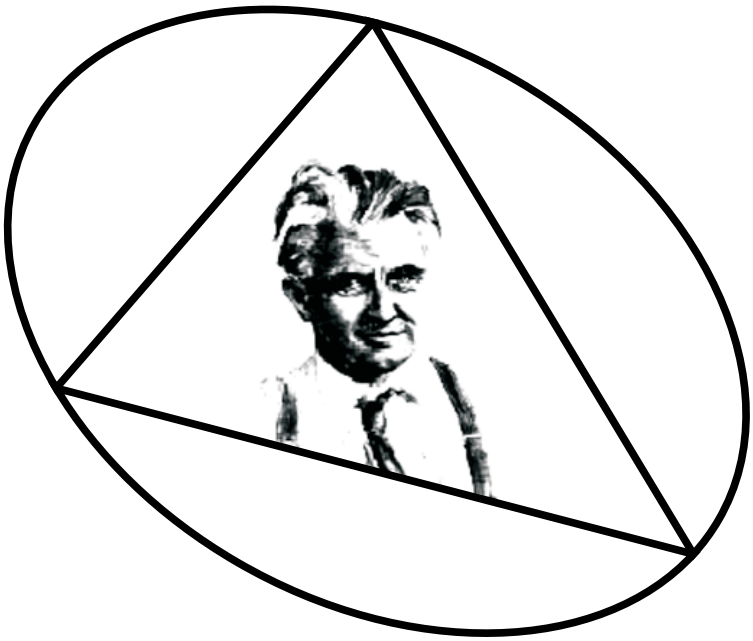
Berechnung von Gröbnerbasen und eine Implementierung des Buchbergeralgorithmus mit *Mathematica*



Diplomarbeit von Tobias Hofmann
Betreuer: W. Koepf
Universität Kassel
September 2003

Inhaltsverzeichnis

Einleitung

Die Theorie der Gröbnerbasen ist ein schönes Beispiel dafür, wie eine Idee zum Lösen eines mathematischen Problems der Schlüssel für die Lösung vieler weiterer Probleme ist, sogar jenseits der Mathematik.
Die Grundidee lässt sich folgendermaßen beschreiben. Wir betrachten das Gleichungssystem

$$\begin{aligned} f_1(x_1, \dots, x_n) &= 0, \\ &\vdots \\ f_s(x_1, \dots, x_n) &= 0, \end{aligned} \tag{1}$$

wobei $f_1, \dots, f_s$ Polynome über einem beliebigem Körper in den Variablen $x_1, \dots, x_n$ sind. Für ein weiteres Polynom f wollen wir nun entscheiden, ob alle Lösungen des obigen polynomiellen Gleichungsystems auch Lösungen von f sind. Wir können dies auf ein algebraisches Problem zurückführen, nämlich darauf, ob das Polynom f ein Element des Ideals I ist, das durch die Polynome $f_1, \dots, f_s$ erzeugt wird. Das Schlüsselwerkzeug zur Lösung dieses „*Ideal Membership Problem*“ genannten Entscheidungsproblems ist eine Gröbnerbasis von I.
Gröbnerbasen sind spezielle Erzeugendensysteme von Polynomidealen, mit denen sich etwa das Ideal Membership Problem einfach durch eine Polynomdivision entscheiden lässt.
Dies ist bei Weitem nicht die einzige Anwendung von Gröbnerbasen. Selbst beim Lösen eines polynomiellen Gleichungssystems, wie (1) es darstellt, sind sie ein nützliches Werkzeug. Solche Gleichungssysteme treten in natürlicher Weise in den verschiedensten Disziplinen auf. Die Chemie z. B. modelliert mit ihnen etwa bestimmte Reaktionssysteme [Fei91] oder beschreibt durch sie molekulare Strukturen im Raum [vdB94]. In der Robotik lassen sich mit polynomiellen Gleichungssystemen die Bewegungsmöglichkeiten von Roboterarmen ausdrücken und damit das kinematische bzw. inverskinematische Problem lösen (siehe [GLGV99] und Kapitel 6 in [CLO98]). Die Geodäsie (Erdvermessung) benutzt sie für die Umrechnung von geodätischen in geozentrische Koordinaten. In der Geometrie lässt sich mit Hilfe solcher Systeme und der Gröbnerbasen gar das Beweisen von Sätzen automatisieren. Und damit ist noch längst nicht alles aufgezählt.
Die Theorie der Gröbnerbasen und der Möglichkeit sie zu berechnen eröffnete ein breites Feld von Forschungsmöglichkeiten. Auch haben sie einen nicht unwesentlichen Teil dazu beigetragen, dass sich die Computeralgebra in den 1990er Jahren zu einer unabhängigen Disziplin in der Mathematik und Informatik entwickelt hat.

Wann immer wir mit polynomiellen Strukturen rechnen wollen, sei es die Bestimmung von Normalformen polynomieller Faktorringe oder etwa die Berechnung von Relationen bzw. Verknüpfungen von Polynomidealen[1], sind in irgend einer Weise auch Gröbnerbasen mit im Spiel. Ihr Erfolg beruht aber nicht nur auf einer Vielzahl von Anwendungen, sondern auch auf dem erzielten Fortschritt in der Computertechnik, der immer leistungsfähigere Rechner hervorbrachte, mit denen die Berechnung von Gröbnerbasen erst möglich wurde.

Die Theorie der Gröbnerbasen geht hauptsächlich auf Bruno Buchberger zurück. Er formulierte in den 1960er Jahren das Konzept dieser speziellen Idealbasen und benannte sie nach seinem Doktorvater Wolfgang Gröbner (siehe Titelblatt). Begründet auf einer Vermutung von Gröbner entwickelte Buchberger auch den nach ihm benannten Algorithmus zur Berechnung von Gröbnerbasen. Dieser ist in seiner ursprünglichen Form zwar einfach zu verstehen, aber auch sehr ineffizient, so dass nach Verfahren gesucht wurde ihn zu beschleunigen. Auch heute ist dies noch ein aktuelles Forschungsthema.

Diese Arbeit ist in drei thematische Bereiche untergliedert.
Wir beginnen mit den grundlegenden Definitionen und Sätzen, die für den Umgang mit Gröbnerbasen unumgänglich sind. Nach der Einführung von Polynomen, affinen Varietäten, Idealen und Monomordnungen behandeln wir den polyvariaten Divisionsalgorithmus und runden das erste Kapitel mit Hilberts Basissatz ab.
In Kapitel 2 führen wir Gröbnerbasen ein, behandeln den Buchbergeralgorithmus und stellen von diesem eine Beispielimplementation in dem Computeralgebrasystem *Mathematica* vor. Die in *Mathematica* integrierte funktionale Programmiersprache ermöglicht uns zwar nicht, das letzte Quentchen an Effizienz aus unserem Algorithmus herauszuholen, aber wir können einen gut lesbaren Code erzeugen, mit dem wir uns auf die wesentlichen Berechnungsschritte beschränken. Den Algorithmus werden wir im weiteren Verlauf durch zwei auf Buchberger zurückgehende Kriterien und die Sugar-Heuristik optimieren.
Im dritten Kapitel beschäftigen wir uns zunächst damit, wie wir mit Hilfe von Gröbnerbasen polynomielle Gleichungssysteme lösen können. Im Anschluss widmen wir uns einer Anwendung aus der Geometrie, in der wir Gröbnerbasen dazu benutzen, aus bekannten geometrischen Sätzen neue zu schließen. Konkret stellen wir die Beziehungen zwischen einer Ellipse und einem Dreieck her, dessen Eckpunkte auf der Ellipse liegen und dessen Schwerpunkt mit dem Ellipsenmittelpunkt übereinstimmt. Dazu entwickeln wir Formeln, mit denen sich die charakteristischen Größen der einen geometrischen Figur aus denen der anderen berechnen lassen.
In Kapitel 2 und 3 verweisen wir auf einige in *Mathematica* implementierte Hilfsfunktionen. Der Code und die Beschreibung dieser Funktionen wurden in den Anhang ausgelagert, da er für das Verständnis der Themen dieser Kapitel nicht wesentlich ist.

[1] z. B.: Gilt für gegebene Polynomideale $I_1, \ldots, I_6$ die Gleichheit $I_1 + I_2 : I_3 = I_4 \cup I_5 \cdot I_6$?

Kapitel 1

Algebraische und geometrische Grundlagen

In diesem Kapitel werden die wesentlichen Notationen und Definitionen für eine sinnvolle Einführung von Gröbnerbasen zusammen gestellt. Grundlegende Begriffe sind etwa Polynomideale und affine Varietäten. Zudem betrachten wir den Divisionsalgorithmus von Polynomen in mehreren Variablen, da er bei der Berechnung dieser Basen eine wesentliche Rolle spielt.

1.1 Polynome

Die grundlegenden Elemente, mit denen wir uns beschäftigen werden, sind *Polynome* in n Variablen $x_1, \dots, x_n$ mit Koeffizienten aus einem beliebigen Körper, den wir mit k bezeichnen. Solche Polynome sind endliche Linearkombinationen (mit Koeffizienten aus dem Körper k) von *Monomen* $\mathbf{x}^\alpha = x_1^{\alpha_1} \dots x_n^{\alpha_n}$ wobei $\alpha = (\alpha_1, \dots, \alpha_n)$.
Die Menge aller Polynome $f = \sum_\alpha a_\alpha \mathbf{x}^\alpha$ mit $a_\alpha \in k$ heißt *Polynomring* über k und wir bezeichnen ihn mit $k[x_1, \dots, x_n]$. Er hat die algebraische Struktur eines kommutativen Ringes mit 1.
Die Elemente a_α sind die *Koeffizienten* des Monoms $\mathbf{x}^\alpha$ und wir nennen $a_\alpha \mathbf{x}^\alpha$ einen *Term* von f, sofern $a_\alpha \neq 0$ ist. Der *totale Grad* des Polynoms, den wir mit $\deg(f)$ bezeichnen, ist das Maximum über alle totalen Monomgrade $|\alpha| = \alpha_1 + \cdots + \alpha_n$.

Wir benötigen auch die funktionale Betrachtungsweise von Polynomen, bei der Punkte $(a_1, \dots, a_n)$ aus dem n-dimensionalen affinen Raum $k^n = \{(a_1, \dots, a_n) \ : \ a_1, \dots, a_n \in k\}$ durch einsetzen in f in den Körper k abgebildet werden. Das Wechselspiel, Polynome einerseits als Elemente einer algebraischen Struktur und andererseits als Funktionen aufzufassen, verbindet die Algebra mit der Geometrie.

1.2 Affine Varietäten

Die fundamentalen geometrischen Objekte, die wir betrachten werden, sind die *affinen Varietäten*. Sie werden durch Polynome $f_1, \dots, f_s \in k[x_1, \dots, x_n]$ wie folgt definiert:

$$\mathbf{V}(f_1, \dots, f_s) := \{(a_1, \dots, a_n) \in k^n \ : \ f_i(a_1, \dots, a_n) = 0 \text{ für alle } 1 \le i \le s\}.$$

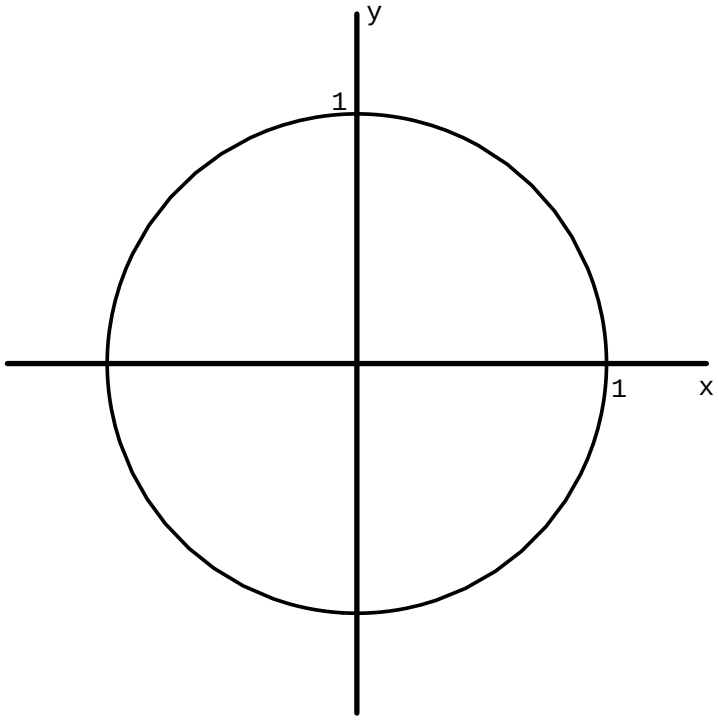
Eine affine Varietät $\mathbf{V}(f_1, \dots, f_s)$ ist folglich die Menge aller Lösungen des polynomiellen Gleichungssystems

$$\begin{aligned} f_1(x_1, \dots, x_n) &= 0, \\ &\vdots \\ f_s(x_1, \dots, x_n) &= 0. \end{aligned}$$

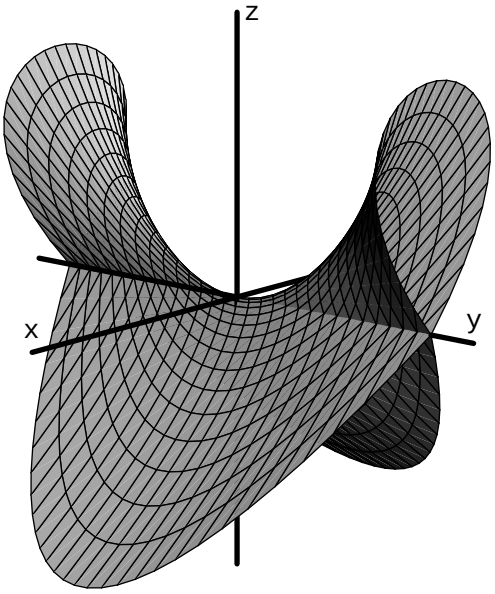
Geometrisch fasst man Varietäten als Teilmengen des n-dimensionalen affinen Raumes k^n auf.

Beispiel 1.1

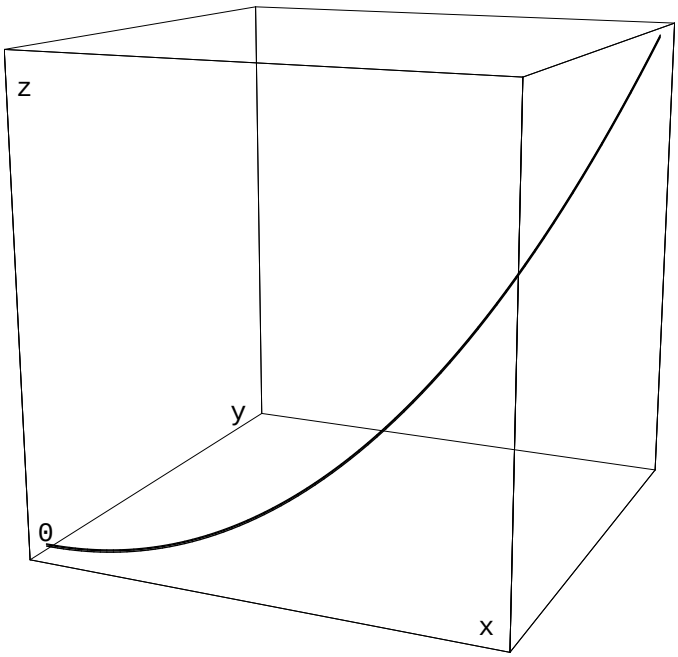
1. *Kreislinie*
   $V_1 = \mathbf{V}(x^2 + y^2 - 1) \subset \mathbb{R}^2$ beschreibt einen Kreis mit Radius 1 um den Ursprung.



2. *Schweinsohrfläche*
   $V_2 = \mathbf{V}(x^2 - y^2z^2 + z^3) \subset \mathbb{R}^3$ bildet eine Fläche im Raum.



3. *Twisted Cubic*
   $V_3 = \mathbf{V}(y - x^2, z - x^3) \subset \mathbb{R}^3$ stellt eine Kurve im Raum dar.



Affine Varietäten lassen sich, wenn wir sie mit einem Computer generieren wollen, besser durch Parameterdarstellungen beschreiben. So haben etwa obige Varietäten folgende Parameterdarstellungen:

1. $V_1 = \left\{\left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}\right) \ : \ t \in \mathbb{R}\right\}$

   V_1 ist die kleinste Varietät, die die Parameterdarstellung enthält. Wie man leicht sieht, gehört der Punkt $(-1, 0)$ nicht zur Parameterdarstellung wohl aber zu der Varietät.

2. $V_2 = \left\{\left(t\left(u^2 - t^2\right), u, u^2 - t^2\right) \ : \ t, u \in \mathbb{R}\right\}$

   In obiger Grafik ist $-1 \leq t, u \leq 1$.

3. $V_3 = \left\{\left(t, t^2, t^3\right) \ : \ t \in \mathbb{R}\right\}$

1.3 Ideale

Zur Beschreibung affiner Varietäten spielt das Ideal als grundlegender algebraischer Begriff eine zentrale Rolle. Wir betrachten hier nur Ideale des Ringes $k[x_1, \ldots, x_n]$. Eine Teilmenge $I \subset k[x_1, \ldots, x_n]$ heißt *Ideal*, wenn

- $0 \in I$,
- $f, g \in I \implies f + g \in I$,
- $f \in I$ und $h \in k[x_1, \ldots, x_n] \implies h \cdot f \in I$

gilt.
Ein naheliegendes Beispiel ist das durch endlich viele Polynome $f_1, \ldots, f_s$ erzeugte Ideal

$$\langle f_1, \ldots, f_s \rangle := \left\{ \sum_{i=1}^{s} h_i \cdot f_i \ : \ h_1, \ldots, h_s \in k[x_1, \ldots, x_n] \right\}.$$

Bei dieser Menge lassen sich die Idealeigenschaften leicht nachprüfen. Die das Ideal erzeugenden Polynome $f_1, \ldots, f_s$ nennen wir eine *Basis* oder ein *Erzeugendensystem*. In Abschnitt 1.6 zeigen wir, dass jedes Polynomideal eine endliche Basis besitzt. Solch eine Basis ist nicht eindeutig und wir werden uns im Kapitel 2 mit Basen beschäftigen, die besonders schöne Eigenschaften besitzen. Dies sind die Gröbnerbasen.

Eine weitere wichtige Klasse von Polynomidealen sind die durch affine Varietäten erzeugten Ideale. Ist $V = \mathbf{V}(f_1, \ldots, f_s) \subset k^n$ definiert durch $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ eine affine Varietät, dann ist die Menge

$$\mathbf{I}(V) := \{f \in k[x_1, \ldots, x_n] \ : \ f(a_1, \ldots, a_n) = 0 \text{ für alle } (a_1, \ldots, a_n) \in V\}$$

aller Polynome, die auf V verschwinden, ein Ideal. Auch dies lässt sich leicht durch Verifikation der Idealeigenschaften nachprüfen. $\mathbf{I}(V)$ wird als *Verschwindungsideal* von V bezeichnet.

Beispiel 1.2

Betrachten wir die Varietät $\{(0,0)\} \subset k^2$. Das Ideal $\mathbf{I}(\{(0,0)\})$ besteht aus allen Polynomen, die im Ursprung verschwinden. Es ist leicht einzusehen, dass

$$\mathbf{I}(\{(0,0)\}) = \langle x, y \rangle$$

gilt.

1.4 Monomordnungen

Für die Berechnung von Gröbnerbasen ist der Divisionsalgorithmus von Polynomen in mehreren Variablen das entscheidende Hilfsmittel. Schlüsselbestandteil dieses Algorithmus ist die zugrundeliegende Ordnung auf den Termen, die wir im folgenden betrachten werden.
Monome $\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ lassen sich eindeutig aus den Exponentenvektoren $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ rekonstruieren[1]. Dies ermöglicht uns eine Ordnung auf $\mathbb{N}^n$ auf Monome zu übertragen. Gilt etwa $\boldsymbol{\alpha} > \boldsymbol{\beta}$ in $\mathbb{N}^n$, so implizieren wir $\mathbf{x}^{\boldsymbol{\alpha}} > \mathbf{x}^{\boldsymbol{\beta}}$.
Eine *Monomordnung* auf $k[x_1, \ldots, x_n]$ ist eine Relation $>$ auf $\mathbb{N}^n$ oder gleichbedeutend eine Relation $>$ auf der Menge der Monome $\mathbf{x}^{\boldsymbol{\alpha}}$, $\boldsymbol{\alpha} \in \mathbb{N}^n$, die folgenden Bedingungen genügt:

(i) $>$ ist eine totale Ordnung auf $\mathbb{N}^n$,

(ii) $\boldsymbol{\alpha} > \boldsymbol{\beta}$ und $\boldsymbol{\gamma} \in \mathbb{N}^n \implies \boldsymbol{\alpha} + \boldsymbol{\gamma} > \boldsymbol{\beta} + \boldsymbol{\gamma}$,

(iii) $>$ ist wohlgeordnet.

Bedingung (i) bedeutet für $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^n$, dass entweder $\boldsymbol{\alpha} > \boldsymbol{\beta}$, $\boldsymbol{\alpha} = \boldsymbol{\beta}$ oder $\boldsymbol{\beta} > \boldsymbol{\alpha}$ gilt. Damit lassen sich die Terme in einem Polynom der Größe nach sortieren. Um das Produkt von Monomen eindeutig anzuordnen, bedienen wir uns der Eigenschaft (ii). Dass ein kleinstes Element existiert, sichert uns (iii). Sofern (i) und (ii) gelten, können wir mit Hilfe von Dicksons Lemma (siehe Abschnitt 1.6) Eigenschaft (iii) auch umformulieren in

(iii') $\boldsymbol{\alpha} \geq 0$ für alle $\boldsymbol{\alpha} \in \mathbb{N}^n$.

Zur Berechnung von Gröbnerbasen sind etwa folgende drei Monomordnungen von Bedeutung. Hierzu seien $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ und $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$.

1. *Lexikographische Ordnung* (*lex*)

   $\boldsymbol{\alpha} >_{lex} \boldsymbol{\beta} \quad :\Longleftrightarrow$ Der erste nichtverschwindende Eintrag von links der Vektordifferenz $\boldsymbol{\alpha} - \boldsymbol{\beta} \in \mathbb{Z}^n$ ist positiv.

   Folglich gilt $\mathbf{x}^{\boldsymbol{\alpha}} >_{lex} \mathbf{x}^{\boldsymbol{\beta}}$, wenn $\boldsymbol{\alpha} >_{lex} \boldsymbol{\beta}$ gilt.

[1] $\mathbb{N} = \mathbb{Z}_{\geq 0}$

2. *Gradlexikographische Ordnung* (*grlex*)

$$\boldsymbol{\alpha} >_{grlex} \boldsymbol{\beta} \iff |\boldsymbol{\alpha}| = \sum_{i=1}^{n} \alpha_i > |\boldsymbol{\beta}| = \sum_{i=1}^{n} \beta_i \text{ , oder } |\boldsymbol{\alpha}| = |\boldsymbol{\beta}| \text{ und } \boldsymbol{\alpha} >_{lex} \boldsymbol{\beta}.$$

3. *Gradinverslexikographische Ordnung* (*grevlex*)

$\boldsymbol{\alpha} >_{grevlex} \boldsymbol{\beta} \iff |\boldsymbol{\alpha}| > |\boldsymbol{\beta}|$, oder $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}|$ und der erste nichtverschwindende Eintrag von rechts der Differenz $\boldsymbol{\alpha} - \boldsymbol{\beta}$ ist negativ.

Zu prüfen wäre nun, ob diese „Ordnungen“ tatsächlich den drei Bedingungen der Monomordnung genügen. Diese Eigenschaften lassen sich aber leicht mit Hilfe der Wohlordnung von $\mathbb{N}$ bestätigen.
Bei allen drei Monomordnungen besitzen die Variablen untereinander folgende Ordnung:

$$x_1 > x_2 > \ldots > x_n.$$

Wir nennen eine Ordnung eine *Gradordnung*, wenn der totale Grad eines Monoms das wichtigste Vergleichskriterium ist. Demnach sind die Monomordnungen *grlex* und *grevlex* Gradordnungen. Mit Monomordnungen lassen sich nun die Terme von Polynomen der Größe nach sortieren.

Beispiel 1.3

Sei $f = 2xy^2z - x^2y + 7y^4 + 3x^2z^2 \in \mathbb{Q}[x,y,z]$ also $x > y > z$. Dann ergeben sich für f folgende Termanordnungen:

$$\begin{aligned} \textit{lex:} &\quad f = -x^2y + 3x^2z^2 + 2xy^2z + 7y^4 \\ \textit{grlex:} &\quad f = 3x^2z^2 + 2xy^2z + 7y^4 - x^2y \\ \textit{grevlex:} &\quad f = 7y^4 + 2xy^2z + 3x^2z^2 - x^2y \end{aligned}$$

Abschließend führen wir noch ein paar nützliche Bezeichnungen ein. Für ein von Null verschiedenes Polynom $f = \sum_{\boldsymbol{\alpha}} a_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} \in k[x_1, \ldots, x_n]$ definieren wir bezüglich einer Monomordnung:

(i) *Multigrad* von f: $\quad \operatorname{mdeg}(f) := \max\{\boldsymbol{\alpha} \in \mathbb{N}^n \ : \ a_{\boldsymbol{\alpha}} \neq 0\}$

(ii) *Leitkoeffizient* von f: $\quad \operatorname{LC}(f) := a_{\operatorname{mdeg}(f)} \in k$

(iii) *Leitmonom* von f: $\quad \operatorname{LM}(f) := \mathbf{x}^{\operatorname{mdeg}(f)}$

(iv) *Leitterm* von f: $\quad \operatorname{LT}(f) := \operatorname{LC}(f) \cdot \operatorname{LM}(f)$

Beispiel 1.4

Für das Polynom $f = 2xy^2z - x^2y + 7y^4 + 3x^2z^2$ aus obigem Beispiel ergeben sich bezüglich den besprochenen Monomordnungen folgende Werte:

	lex	*grlex*	*grevlex*
mdeg(f)	$(2,1,0)$	$(2,0,2)$	$(0,4,0)$
LC(f)	-1	3	7
LM(f)	x^2y	x^2z^2	y^4
LT(f)	$-x^2y$	$3x^2z^2$	$7y^4$

1.5 Der Divisionsalgorithmus

Mit dem Divisionsalgorithmus für Polynome in einer Variablen lassen sich für $f, g \in k[x]$, $g \neq 0$, Polynome $q, r \in k[x]$ bestimmen, so dass $f = qg + r$ gilt und entweder $r = 0$ oder $\deg(r) < \deg(g)$ ist. Diese Darstellung von f ist eindeutig, denn aus $f = q_1g + r_1 = q_2g + r_2$ folgt, dass $r_1 - r_2$ ein Vielfaches von g ist. Dies ist aber nur möglich, wenn $r_1 - r_2 = 0$, da nach Voraussetzung die Grade von r_1 und r_2 kleiner sind als der Grad von g (vergleiche [CLO98] Seite 37-39).
Dieser Algorithmus lässt sich für das Lösen des „*Ideal Membership Problems*“ verwenden, das der Frage nachgeht, ob ein gegebenes Polynom f in einem Ideal $I \subset k[x]$ enthalten ist. Da $k[x]$ ein Hauptidealring ist, existiert ein $g \in k[x]$ mit $I = \langle g \rangle$. Das Polynom f ist genau dann ein Element von I, wenn in der mit dem Divisionsalgorithmus berechneten Darstellung $f = qg + r$ das Polynom r verschwindet.

Der Ring $k[x_1, \ldots, x_n]$ ist allerdings kein Hauptidealring. Sei etwa $I = \langle x, y \rangle \subsetneq k[x, y]$. Falls $k[x, y]$ ein Hauptidealring wäre, würde ein $f \in k[x, y]$ mit $\langle f \rangle = I$ existieren. Wegen $\langle f \rangle \subset I$ würde f sowohl x als auch y teilen und somit konstant sein. Dann aber wäre $I = k[x, y]$.
Im Allgemeinen wird folglich ein Ideal I von mehreren Polynomen $f_1, \ldots, f_s$ erzeugt und wir können ein $f \in k[x_1, \ldots, x_n]$ mit diesen Erzeugern in der Form

$$f = q_1 f_1 + \cdots + q_s f_s + r \tag{1.1}$$

mit $q_1, \ldots, q_s, r \in k[x_1, \ldots, x_n]$ darstellen. Finden wir eine derartige Darstellung in der r verschwindet, ist f in I enthalten.

Der Divisionsalgorithmus, mit dem wir f wie in (1.1) zerlegen können, lässt sich wie folgt beschreiben.

Algorithmus 1 (Divisionsalgorithmus)

Input: *Polynome $f, f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ und eine Monomordnung.*

Output: *$q_1, \ldots, q_s, r \in k[x_1, \ldots, x_n]$, so dass $f = q_1 f_1 + \cdots + q_s f_s + r$ gilt und kein Monom von r durch einen der Leitterme $\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_s)$ teilbar ist.*

1: *Initialisiere alle q_i's bzw. r auf Null und weise p das Polynom f zu.*

2: *Solange $p \neq 0$ tue folgendes*

3: *wenn eines der $LT(f_i)$'s $LT(p)$ teilt,*
wähle solch ein i und setze $q_i := q_i + \frac{LT(p)}{LT(f_i)}$ und $p := p - \frac{LT(p)}{LT(f_i)} f_i$,
ansonsten setze $r := r + LT(p)$ und $p := p - LT(p)$.

4: *Rückgabe von $q_1, \dots, q_s, r$.*

Es ist natürlich etwas gewagt, hier von einem Algorithmus zu sprechen, wo doch im 3. Schritt ziemlich freizügig mit der Wahl von i umgegangen wird. Im folgenden Beispiel wird hierauf konkret eingegangen und auch ein Vorschlag gegeben den Algorithmus zu „determinieren“. Dieses und das nächste Beispiel konkretisieren zudem die Unterschiede zwischen den Divisionsalgorithmen in $k[x]$ und $k[x_1, \dots, x_n]$.

Beispiel 1.5

Sei $f = xy^2 - 2$, $f_1 = xy - 1$, $f_2 = y + 1$ und wir führen bezüglich *grlex*-Ordnung mit $x > y$ schrittweise den „Divisionsalgorithmus“ durch.

	$xy-1$ $y+1$		$xy-1$ $y+1$		$xy-1$ $y+1$
xy^2-2	y	xy^2-2	xy	xy^2-2	xy
$-(xy^2-y)$		$-(xy^2+xy)$		$-(xy^2+xy)$	
$y-2$	1	$-xy-2$	$-x$	$-xy-2$	-1
$-(y+1)$		$-(-xy-x)$		$-(-xy+1)$	
-3		$x-2$		-3	

Wie oben erwähnt geht aus der zweiten Zeile im Schritt 3 des Algorithmus nicht hervor, welches Polynom f_i wir für die Division zu wählen haben, wenn wie hier $LT(f_1) = xy$ und $LT(f_2) = y$ beide $LT(f) = xy^2$ teilen. Die drei Rechnungen zeigen uns, dass sich je nach Wahl unterschiedliche Darstellungsformen für f ergeben. Im Gegensatz zum Divisionsalgorithmus in einer Variablen gibt es also weder eine deterministische Rechenvorschrift noch eine eindeutige Darstellung von f mittels der f_i's. Denn sowohl $f = y \cdot f_1 + 1 \cdot f_2 - 3$ als auch $f = 0 \cdot f_1 + (xy - x) \cdot f_2 + (x - 2)$ bzw. $f = -1 \cdot f_1 + xy \cdot f_2$ genügen (1.1). Erst durch die Forderung das kleinste i im 3. Schritt des Algorithmus zu wählen, ist eine eindeutige Darstellung von f mittels der f_i's bestimmt. Diese lässt sich dann determisistisch berechnen.
Interessieren wir uns vordergründig für den Rest r bei der Division von f durch das geordnete Tupel $(f_1, \dots, f_s)$, so schreiben wir

$$r = \overline{f}^{(f_1,\dots,f_s)}.$$

Beispiel 1.6

Nun werden wir $f = x^2y^2 + y^3 - x^2 - y$ durch die Polynome $f_1 = xy + 1$ und $f_2 = y^2 - 1$ dividieren. Wie im ersten Beispiel tun wir dies unter Berücksichtigung der *grlex*-Ordnung mit $x > y$.

	$xy+1$	y^2-1	Rest
$x^2y^2 + y^3 - x^2 - y$	xy		
$-(x^2y^2 + xy)$			
$y^3 - x^2 - xy - y$		y	
$-(y^3 - y)$			
$-x^2 - xy$			$-x^2$
$-(-x^2)$			
$-xy$	-1		
$-(-xy - 1)$			
1			

Hier tritt ein Phänomen auf, das es bei der Polynomdivision in einer Variablen auch nicht gibt. Im dritten Berechnungsschritt ist der Leitterm $-x^2$. Dieser ist weder durch $\mathrm{LT}(f_1) = xy$ noch durch $\mathrm{LT}(f_2) = y^2$ teilbar. Nachdem wir $-x^2$ in die „Restspalte“ geschrieben haben, lässt sich die Division fortsetzen und wir erhalten

$$f = (xy - 1) \cdot f_1 + y \cdot f_2 + (-x^2 + 1). \tag{1.2}$$

Im Algorithmus ist dieses Vorgehen in der dritten Zeile in Schritt 3 umgesetzt.

Die Tatsache, dass die Darstellung von f mittels der f_i's nicht eindeutig ist, erschwert es der Frage nachzugehen, ob f ein Element von I ist. Schauen wir uns nochmals Beispiel 1.6 an. Der Divisionsalgorithmus liefert für f die Darstellung (1.2) und folglich ist $\overline{f}^{(f_1,f_2)} = -x^2 + 1$. Wären wir im Polynomring in einer Variablen, würde hieraus folgen, dass f kein Element von I ist. Es gilt aber auch

$$f = 0 \cdot f_1 + (x^2 + y) \cdot f_2 + 0,$$

also $\overline{f}^{(f_2,f_1)} = 0$ und demnach ist f doch ein Element von I. Dies motiviert die Behandlung von Gröbnerbasen, denn mit ihnen lässt sich, wie wir in Kapitel 2 (insbesondere auf Seite 16) sehen werden, das „Ideal Membership Problem“ lösen.

1.6 Monomideale und Hilberts Basissatz

Im Zusammenhang mit Gröbnerbasen spielen Monomideale eine große Rolle. Ein Ideal $I \subset k[x_1, \ldots, x_n]$ nennen wir *Monomideal*, falls eine Untermenge $A \subset \mathbb{N}^n$ existiert, so dass

$$I = \langle \mathbf{x}^A \rangle := \langle \{\mathbf{x}^\alpha \ : \ \alpha \in A\} \rangle$$

gilt, d. h. sofern eine Erzeugermenge ausschließlich aus Monomen besteht. Ein Polynom f ist demnach genau dann ein Element eines Monomideals I, wenn sämtliche Terme von f in I liegen. Dies lässt sich leicht nachprüfen, denn es gilt

$$\mathbf{x}^{\beta} \in I \iff \exists\, \alpha \in A : \mathbf{x}^{\alpha} \mid \mathbf{x}^{\beta}. \tag{1.3}$$

Der folgende Satz macht eine wichtige Aussage über Monomideale.

Satz 2 (Dicksons Lemma) *Jedes Monomideal $I = \langle \mathbf{x}^A \rangle \subset k[x_1, \dots, x_n]$ ist endlich erzeugt. Genauer: Für jedes $A \subset \mathbb{N}^n$ existiert eine endliche Untermenge $B \subset A$, so dass $\langle \mathbf{x}^B \rangle = \langle \mathbf{x}^A \rangle$.*

Beweisskizze (Siehe [CLO98], Seite 69/70)
Induktion über n: $n = 1$ ist klar, da $k[x]$ ein Hauptidealring ist. Für den Induktionsschritt sei $I \subset k[x_1, \dots, x_{n-1}, y]$ ein Monomideal und $J \subset k[x_1, \dots, x_{n-1}]$ die „Projektion“ von I auf $k[x_1, \dots, x_{n-1}]$, die nach Induktionsvoraussetzung eine endliche Basis $\langle \mathbf{x}^{\alpha(1)}, \dots, \mathbf{x}^{\alpha(s)} \rangle$ besitzt. Es existieren $m_i \in \mathbb{N}$ mit $\mathbf{x}^{\alpha_i} y^{m_i} \in I$. Sei m das Maximum der m_i's und für jedes $k \in \{0, \dots, m-1\}$ sei das Ideal $J_k \subset k[x_1, \dots, x_{m-1}]$ erzeugt durch Monome $\mathbf{x}^{\beta}$, so dass $\mathbf{x}^{\beta} y^k \in I$ gilt. Auch die J_k's sind nach Induktionsvoraussetzung endlich erzeugt, etwa $J_k = \langle \mathbf{x}^{\alpha_k(1)}, \dots, \mathbf{x}^{\alpha_k(s_k)} \rangle$. Nun lässt sich zeigen, dass I von den Monomen $\mathbf{x}^{\alpha_k(1)} y^k, \dots, \mathbf{x}^{\alpha_k(s_k)} y^k$ aus den J_k's erzeugt wird, wobei $k \in \{0, \dots, m\}$ und $J_m = J$ ist. □

Wählen wir uns nun eine feste Monomordnung. Jedes Polynom $f \in k[x_1, \dots, x_n]$ besitzt bezüglich dieser Ordnung einen eindeutigen Leitterm $\mathrm{LT}(f)$. Für ein Ideal $I \subset k[x_1, \dots, x_n]$ sei

$$\mathrm{LT}(I) := \{ c\mathbf{x}^{\alpha} \ : \ \exists f \in I : \mathrm{LT}(f) = c\mathbf{x}^{\alpha} \}$$

die Menge aller Leitterme von Elementen aus I und $\langle \mathrm{LT}(I) \rangle$ das *Ideal der Leitterme*. Nach Dicksons Lemma existiert zu jedem Ideal I eine endliche Menge $G = \{g_1, \dots, g_s\} \subset I$, so dass

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

gilt, wobei $\mathrm{LT}(G) := \{\mathrm{LT}(g_1), \dots, \mathrm{LT}(g_s)\}$. Umgekehrt kann es jedoch sein, dass G eine endliche Erzeugermenge von I ist, die Ideale $\langle \mathrm{LT}(G) \rangle$ und $\langle \mathrm{LT}(I) \rangle$ aber nicht identisch sind. Dann ist aber zumindest $\langle \mathrm{LT}(G) \rangle$ in $\langle \mathrm{LT}(I) \rangle$ echt enthalten.

Beispiel 1.7

Sei $g_1 = x^2 + x$, $g_2 = xy + y + 1 \in \mathbb{Q}[x, y]$, $G = \{g_1, g_2\}$ und $I = \langle G \rangle$. Bezüglich *grlex*-Ordnung gilt $x = -y \cdot g_1 + x \cdot g_2$, also $x \in \langle \mathrm{LT}(I) \rangle$. Jedoch ist $x \notin \langle \mathrm{LT}(G) \rangle = \langle x^3, x^2 y \rangle$.

Demgegenüber folgt mit dem Divisionsalgorithmus und Eigenschaft (1.3) für eine endliche Menge $G \subset I \subset k[x_1, \dots, x_n]$

$$\langle \mathrm{LT}(G) \rangle = \langle \mathrm{LT}(I) \rangle \implies \langle G \rangle = I. \tag{1.4}$$

Hiermit können wir nun folgendes berühmte Resultat verifizieren.

Satz 3 (Hilberts Basissatz) *Jedes Ideal $I \subset k[x_1, \dots, x_n]$ ist endlich erzeugt, d. h. $k[x_1, \dots, x_n]$ ist noethersch. Genauer: Es existiert eine endliche Menge $G \subset I$ mit $\langle G \rangle = I$.*

Wie für alle noetherschen Ringe haben wir die zu Hilberts Basissatz äquivalente Aussage:

Korollar 4 (Ascending Chain Condition, ACC) *Sei $I_1 \subset I_2 \subset I_3 \subset \dots$ eine aufsteigende Kette von Idealen aus $k[x_1, \dots, x_n]$, dann existiert ein $n \geq 1$, so dass $I_n = I_{n+1} = I_{n+2} = \dots$ gilt.*

Eine zweite Konsequenz aus Hilberts Basissatz ist von geometrischer Art. Hierzu definieren wir für ein beliebiges Ideal $I \subset k[x_1, \dots, x_n]$ die affine Varietät

$$\mathbf{V}(I) := \{(a_1, \dots, a_n) \in k^n \ : \ f(a_1, \dots, a_n) = 0 \text{ für alle } f \in I\}.$$

Da $k[x_1, \dots, x_n]$ noethersch ist, ist I durch endlich viele Polynome $f_1, \dots, f_s$ erzeugt und es lässt sich zeigen, dass

$$\mathbf{V}(I) = \mathbf{V}(f_1, \dots, f_s). \tag{1.5}$$

Affine Varietäten lassen sich also stets durch endlich viele Polynome charakterisieren. Zudem gilt auch für alle $f_1, \dots, f_s, g_1, \dots, g_t \in k[x_1, \dots, x_n]$

$$\mathbf{V}(f_1, \dots, f_s) = \mathbf{V}(g_1, \dots, g_t), \text{ sofern } \langle f_1, \dots, f_s \rangle = \langle g_1, \dots, g_t \rangle. \tag{1.6}$$

Kapitel 2

Gröbnerbasen und ihre Berechnung

Nun steht der Einführung von Gröbnerbasen nichts mehr im Weg. Nach der Definition und der Betrachtung einiger Eigenschaften dieser speziellen Basen werden wir uns eingehend mit deren Berechnung beschäftigen. Der Buchbergeralgorithmus dient uns hier als Grundlage. Dieser ist nicht streng deterministisch beschrieben und so lässt er uns einige Freiheiten bei der Berechnung von Gröbnerbasen. Doch diese Freiheiten haben wiederum großen Einfluss auf die Komplexität des Algorithmus bei der computerunterstützten Berechnung. Es ist sinnvoll, durch geeignete Strategien diesen Freiraum gewinnbringend, zum Vorteil der Laufzeit und des Speicherbedarfs, auszuschöpfen. Die derzeit beste „Strategie“ ist die Sugar-Heuristik. Hierauf wird in diesem Kapitel ausführlich eingegangen.

2.1 Gröbnerbasen

Idealbasen sind nicht eindeutig. Dies nutzen wir, um mit den Fragestellungen, die sich beim Umgang mit Idealen ergeben, besser umgehen zu können. Schauen wir uns etwa das im Abschnitt 1.5 angesprochene *Ideal Membership Problem* an. Im Beispiel 1.7 aus Kapitel 1 wurde uns verdeutlicht, dass wir nicht von der Gleichheit

$$\langle \mathrm{LT}(G)\rangle = \langle \mathrm{LT}(\langle G\rangle)\rangle \text{ mit } G = \{g_1, \dots, g_s\} \subset k[x_1, \dots, x_n] \tag{2.1}$$

ausgehen können. Nehmen wir an, (2.1) würde gelten. Dann errechnet der mehrdimensionale Divisionsalgorithmus für f ein eindeutiges Restpolynom r, unabhängig von der Divisionsreihenfolge durch die Polynome g_i.
Um dies zu sehen, seien $f, g_1, \dots, g_s \in k[x_1, \dots, x_n]$, $G = \{g_1, \dots, g_s\}$ und es gelte die Gleichheit (2.1). Seien $f = q_1g_1 + \cdots + q_sg_s + r_1 = p_1g_1 + \cdots + p_sg_s + r_2$ zwei vom Divisionsalgorithmus, so wie er im Abschnitt 1.5 beschrieben ist, berechnete Darstellungen von f. Da diese von der Divisionsreihenfolge der g_i's abhängig sind, werden sie nicht notwendigerweise identisch sein, wie im Beispiel 1.6 aus Kapitel 1 zu sehen ist. In jedem Fall ist kein Term von r_1 bzw. r_2 durch irgendeinen Leitterm der g_i's teilbar.

Nun gilt

$$r_1 - r_2 = (p_1 - q_1)\,g_1 + \ldots + (p_s - q_s)\,g_s \in \langle G\rangle$$

und

$$\mathrm{LT}(r_1 - r_2) \in \langle \mathrm{LT}(\langle G\rangle)\rangle = \langle \mathrm{LT}(G)\rangle.$$

Wegen (1.3) existiert jedoch ein Polynom $g \in G$, so dass $\mathrm{LT}(g)$ das Monom $\mathrm{LT}(r_1 - r_2)$ teilt. Folglich ist $\mathrm{LT}(r_1 - r_2) = 0$, also $r_1 = r_2$ und damit sind die Restpolynome eindeutig.

Schauen wir uns ein Beispiel an. Wir verwenden, wie auch im Weiteren, das Computeralgebrasystem *Mathematica* in der Version 4.1. Hier ist der Divisionsalgorithmus unter dem Funktionsnamen **`PolynomialReduce`** implementiert. *Mathematica* beschreibt die Syntax wie folgt.

```
In[1]:= ?PolynomialReduce

PolynomialReduce[poly, {poly1, poly2, ... }, {x1, x2, ... }] yields a
list representing a reduction of poly in terms of the polyi. The list
has the form {{a1, a2, ... }, b}, where b is minimal and a1 poly1 +
a2 poly2 + ... + b is exactly poly.
```

Als erstes Argument erwartet die Funktion ein Polynom. Dieses teilt sie durch die Polynome aus der Liste, die als zweites Argument übergeben wird. Durch die Reihenfolge der Polynome in dieser Liste, `polyi` hat bei der Division den Vorrang vor `polyi+1`, ist der Algorithmus deterministisch. Über das dritte Argument wird mit einer Variablenliste die Ordnung der vorkommenden Variablen festgelegt. Bei der Einführung der Monomordnungen im Abschnitt 1.4 haben wir gesehen, dass bei den drei betrachteten Monomordnungen die Variablennamen $x_1, \ldots, x_n$ des Polynomrings $k[x_1, \ldots, x_n]$ wie folgt angeordnet werden

$$x_1 > x_2 > \ldots > x_n.$$

Mathematica ist es egal, welche Namen die Variablen haben. Wollen wir etwa im Polynomring $\mathbb{Q}[t, o, b, i]$ rechnen, so legen wir mit der Liste **`{t, o, b, i}`** die Ordnung $t > o > b > i$ fest, also $x_1 = t$, $x_2 = o$, $x_3 = b$, $x_4 = i$.
Worüber sich *Mathematica* bei der Kurzhilfe ausschweigt, sind die Optionen, mit denen wir die Funktionsweise der Funktion modifizieren können. Sie werden auch als Argumente übergeben. Die wichtigste Option, die **`PolynomialReduce`** unterstützt, ist die Einstellungsmöglichkeit der zugrundeliegenden Monomordnung. Folgende Werte sind möglich:

`MonomialOrder` → **`Lexicographic`**	*lex*-Ordnung,
`MonomialOrder` → **`DegreeLexicographic`**	*grlex*-Ordnung,
`MonomialOrder` → **`DegreeReverseLexicographic`**	*grevlex*-Ordnung.

Standardmäßig wird die lexikographische Ordnung benutzt. Eine weitere Option von **`PolynomialReduce`** wird etwa in Beispiel 2.4.3 erläutert.

Beispiel 2.1

Betrachten wir das Ideal $I = \langle g_1, g_2 \rangle = \langle x - z, y - z \rangle$. Es genügt der Gleichung (2.1), d. h. es gilt $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), \mathrm{LT}(g_2) \rangle = \langle x, y \rangle$ für $x > y > z$ unter *lex*-Ordnung. In Abschnitt 2.2 wird dies auch gezeigt. Wir wollen $f = xyz \in \mathbb{Q}[x, y, z]$ durch die Erzeuger von I teilen und uns das Restpolynom anschauen. Da es, abhängig von der Divisionsreihenfolge, mehrere Vorgehensweisen bei der Division gibt, betrachten wir alle, also in diesem Fall zwei.

```
In[2]:= g1 = {x - z, y - z};
        g2 = {y - z, x - z};
        f = x y z;

In[3]:= PolynomialReduce[f, g1, {x, y, z}]
Out[3]= {{y z, z^2}, z^3}

In[4]:= PolynomialReduce[f, g2, {x, y, z}]
Out[4]= {{x z, z^2}, z^3}
```

Wir erhalten

$$f = yz \cdot g_1 + z^2 \cdot g_2 + z^3 \quad \text{und} \quad f = z^2 \cdot g_1 + xz \cdot g_2 + z^3.$$

Mathematica berechnet für f, je nach Divisionsreihenfolge von g_1 und g_2, zwei unterschiedliche Darstellungen. Wie wir aber oben gezeigt haben, stimmt bei beiden Darstellungen das Restpolynom überein, d. h.

$$\overline{xyz}^{(x-z,y-z)} = \overline{xyz}^{(y-z,x-z)} = z^3.$$

Die Eindeutigkeit des Restpolynoms ist von entscheidender Bedeutung bei unseren weiteren Ausführungen. Geben wir zunächst dem Erzeugendensystem eines Ideals, das der Eigenschaft (2.1) genügt, einen Namen.

Definition 5 (Gröbnerbasis) *Sei $I \subseteq k[x_1, \ldots, x_n]$ ein Ideal. Für eine fest gewählte Monomordnung heißt eine endliche Menge $G \subset I$ eine Gröbnerbasis von I, wenn $\langle LT(G) \rangle = \langle LT(I) \rangle$ gilt.*

Mit den Resultaten aus Abschnitt 1.6 lässt sich leicht die Existenz von Gröbnerbasen zu jedem Ideal $I \neq \{0\}$ folgern. Zudem wird aus (1.4) ersichtlich, dass eine Gröbnerbasis von I wirklich eine Basis dieses Ideals ist.

Wenden wir uns nun wieder dem anfangs erwähnten *Ideal Membership Problem* zu. Wir wollen entscheiden, ob das Polynom f ein Element des Ideals $I = \langle g_1, \ldots, g_t \rangle$ ist. Stellt die Erzeugermenge $\{g_1, \ldots, g_t\}$ eine Gröbnerbasis von I dar, so ist das Restpolynom r bei Division von f durch $(g_1, \ldots, g_t)$ eindeutig. Folglich gehört f genau dann zu I, wenn r das Nullpolynom ist. Dieses Resultat wollen wir im folgenden Satz festhalten.

Satz 6 *Sei $G = \{g_1, \dots, g_t\} \subset k[x_1, \dots, x_n]$ eine Gröbnerbasis von I, dann gilt*

$$f \in I \iff \overline{f}^{\{g_1,\dots,g_t\}} = 0.$$

In diesem Satz wurde bewusst die Mengenschreibweise anstelle der Tupelschreibweise im Ausdruck $\overline{f}^{\{g_1,\dots,g_t\}}$ benutzt. Sie verdeutlicht die Unabhängigkeit des Restpolynoms von der Divisionsreihenfolge durch die g_i's. Diese Schreibweise ist aber nur sinnvoll, sofern $\{g_1, \dots, g_t\}$ eine Gröbnerbasis darstellt.

2.2 Der Buchbergeralgorithmus

Kennen wir zu einem Ideal I eine Gröbnerbasis, so können wir das *Ideal Membership Problem* lösen. Nun gehen wir der Frage nach, wie sich entscheiden lässt, ob eine gegebene Basis eine Gröbnerbasis ist und wie sich gegebenenfalls Gröbnerbasen berechnen lassen. Folgende einfache Konstruktion wird dabei eine entscheidende Rolle spielen.

Definition 7 (S-Polynom) *Seien $f, g \in k[x_1, \dots, x_n]$ von Null verschiedene Polynome, $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) = \mathrm{mdeg}(f)$, $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n) = \mathrm{mdeg}(g)$ und $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_n)$ mit $\gamma_i = \max\{\alpha_i, \beta_i\}$. Wir nennen*

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

das S-Polynom von f und g.

Hier ist $\mathbf{x}^{\boldsymbol{\gamma}}$ das kleinste gemeinsame Vielfache von $\mathrm{LM}(f)$ und $\mathrm{LM}(g)$. Sind f und g Elemente eines Ideals I, so auch $S(f,g)$, da $\frac{\mathbf{x}^{\boldsymbol{\gamma}}}{\mathrm{LT}(f)}, \frac{\mathbf{x}^{\boldsymbol{\gamma}}}{\mathrm{LT}(g)} \in k[x_1, \dots, x_n]$.

Beispiel 2.2

Wir berechnen das S-Polynom von $f = y - x^2, g = z - x^3 \in \mathbb{R}[x,y,z]$ bezüglich *lex*-Ordnung. Die Varietät dieser Polynome ist die aus Abschnitt 1.2 bekannte *Twisted Cubic*. Mit obigen Bezeichnungen ist $\boldsymbol{\gamma} = (3,0,0)$ und

$$\begin{aligned} S(f,g) &= \frac{x^3}{\mathrm{LT}(f)} \cdot f - \frac{x^3}{\mathrm{LT}(g)} \cdot g \\ &= \frac{x^3}{-x^2} \cdot (y - x^2) - \frac{x^3}{-x^3} \cdot (z - x^3) \\ &= -xy + x^3 + z - x^3 \\ &= -xy + z. \end{aligned}$$

S-Polynome wurden definiert, um Leitterme zu beseitigen. Wie im obigen Beispiel zu

sehen, ist der Leitterm des S-Polynoms $\mathrm{LT}(S(f,g)) = -xy$ kleiner als die Leitterme von f bzw. g. Aus diesem Verhalten resultiert ein Kriterium zur Entscheidung, ob eine gegebene Idealbasis eine Gröbnerbasis ist.
Betrachten wir hierzu bezüglich einer Monomordnung die zwei kleinsten Erzeugerpolynome f_i, f_j eines Ideals $\langle f_1, \dots, f_s\rangle = I \subset k[x_1, \dots, x_n]$. Gilt

$$\mathrm{LT}(S(f_i, f_j)) < \min\{\mathrm{LT}(f_i), \mathrm{LT}(f_j)\}$$

wie im Beispiel 2.2, so folgt

$$\mathrm{LT}(S(f_i, f_j)) \neq \langle \mathrm{LT}(f_1), \dots, \mathrm{LT}(f_s)\rangle.$$

Da aber $\mathrm{LT}(S(f_i, f_j))$ ein Element von $\langle \mathrm{LT}(I)\rangle$ ist, folgt

$$\langle \mathrm{LT}(f_1), \dots, \mathrm{LT}(f_s)\rangle \subsetneq \langle \mathrm{LT}(I)\rangle$$

und demnach ist $\{f_1, \dots, f_s\}$ keine Gröbnerbasis. Wir können also S-Polynome als potentielle Kandidaten verwenden, um eine Idealbasis als Nicht-Gröbnerbasis zu entlarven.
Damit ist aber das Potential der S-Polynome noch nicht ausgeschöpft. Folgender Satz gibt uns ein Entscheidungskriteriumin über das Vorliegen einer Gröbnerbasis in die Hand.

Satz 8 (Buchbergers S-Paar Kriterium) *Sei I ein Polynomideal. Eine Basis $G = \{g_1, \dots, g_t\}$ von I ist genau dann eine Gröbnerbasis von I, wenn gilt:*

$$\overline{S(g_i, g_j)}^{G} = 0 \quad \textit{für alle } 1 \le i < j \le t.$$

Beweisskizze (siehe [CLO98], Seite 81-84)
$\Longrightarrow$: Da $S(g_i, g_j) \in I$, ist mit Satz 6 auch der Divisionsrest von $S(g_i, g_j)$ durch G gleich Null.
$\Longleftarrow$: Wir werden für ein beliebiges $f \in I \setminus \{0\}$ zeigen, dass $\mathrm{LT}(f) \in \langle \mathrm{LT}(G)\rangle$ gilt. Sei

$$f = \sum_{i=1}^{t} h_i g_i \tag{1}$$

eine Darstellung von f mit $h_i \in k[x_1, \dots, x_n]$, sei $m(i) = \mathrm{mdeg}(h_i g_i)$ und $\delta = \max\{m(i)\}$. Es gilt

$$\mathrm{mdeg}(f) \le \delta. \tag{2}$$

Liegt keine Gleichheit vor, heben sich in Darstellung (1) ein paar Leitterme der $h_i g_i$'s weg, so dass $\mathrm{LT}(f)$ nicht in $\langle \mathrm{LT}(G)\rangle$ liegt. Folglich haben wir die Gleichheit in (2) zu zeigen. Hierzu wählen wir eine Darstellung (1) mit minimalem δ für f und schreiben diese zunächst um:

$$f = \sum_{m(i)=\delta} \mathrm{LT}(h_i) g_i + \sum_{m(i)=\delta} (h_i - \mathrm{LT}(h_i)) g_i + \sum_{m(i)<\delta} h_i g_i\,. \tag{3}$$

Angenommen der Multigrad von f ist kleiner als δ, dann gilt dies auch für den Multigrad der ersten Summe, da die Multigrade der zweiten und dritten Summe schon

konstruktionsmäßig kleiner sind.
Betrachten wir nun die erste Summe. Jedes Monom hat Multigrad $\boldsymbol{\delta}$. Es lässt sich zeigen (siehe [CLO98], Seite 81/82), dass eine Summe dieser Gestalt mit $\mathrm{LT}(h_i) = c_i\,\mathbf{x}^{\boldsymbol{\alpha}(i)}$ als Linearkombination der S-Polynome $S(\mathbf{x}^{\boldsymbol{\alpha}(j)}g_j, \mathbf{x}^{\boldsymbol{\alpha}(k)}g_k)$ darstellbar und deren Multigrad kleiner als $\boldsymbol{\delta}$ ist. Wir formen um:

$$\begin{aligned}
\sum_{m(i)=\boldsymbol{\delta}} \mathrm{LT}(h_i)g_i &= \sum_{m(i)=\boldsymbol{\delta}} c_i\,\mathbf{x}^{\boldsymbol{\alpha}(i)}g_i \\
&= \sum_{j,k} c_{jk} S(\mathbf{x}^{\boldsymbol{\alpha}(j)}g_j, \mathbf{x}^{\boldsymbol{\alpha}(k)}g_k) \\
&= \sum_{j,k} c_{jk}\left(\frac{\mathbf{x}^{\boldsymbol{\delta}}}{\mathbf{x}^{\boldsymbol{\alpha}(j)}\mathrm{LT}(g_j)}\mathbf{x}^{\boldsymbol{\alpha}(j)}g_j - \frac{\mathbf{x}^{\boldsymbol{\delta}}}{\mathbf{x}^{\boldsymbol{\alpha}(k)}\mathrm{LT}(g_k)}\mathbf{x}^{\boldsymbol{\alpha}(k)}g_k\right) \\
&= \sum_{j,k} c_{jk}\,\mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_{jk}} S(g_j, g_k) \quad \text{mit} \quad \mathbf{x}^{\boldsymbol{\gamma}_{jk}} = \mathrm{kgV}(\mathrm{LM}(g_j), \mathrm{LM}(g_k))
\end{aligned}$$

Nach Voraussetzung ist der Rest bei Division der S-Polynome durch G gleich Null. Demnach erhalten wir mit dem Divisionsalgorithmus eine Darstellung

$$S(g_j, g_k) = \sum_{i=1}^{t} a_{ijk} g_i \tag{4}$$

mit $a_{ijk} \in k[x_1, \ldots, x_n]$, in der

$$\mathrm{mdeg}(a_{ijk}g_i) \leq \mathrm{mdeg}(S(g_j, g_k)) \tag{5}$$

gilt. Für obige Gleichungskette folgt weiter

$$\begin{aligned}
&= \sum_{j,k} c_{jk}\,\mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_{jk}} \sum_i a_{ijk} g_i \\
&= \sum_{i,j,k} c_{ijk} b_{ijk} g_i \qquad \text{mit} \quad b_{ijk} = \mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_{jk}} a_{ijk} \\
&= \sum_i \tilde{h}_i g_i \qquad \text{mit} \quad \tilde{h}_i = \sum_{jk} c_{ijk} b_{ijk}\,.
\end{aligned}$$

Für den Multigrad ergibt sich

$$\mathrm{mdeg}(b_{ijk}g_i) \leq \mathrm{mdeg}(\mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_{jk}} S(g_j, g_k)) = \mathrm{mdeg}(\tilde{h}_i g_i) < \boldsymbol{\delta}.$$

Wenn wir in (3) die Summe $\sum_{m(i)=\boldsymbol{\delta}} \mathrm{LT}(h_i)g_i$ durch $\sum_i \tilde{h}_i g_i$ ersetzen, sehen wir, dass in der Darstellung von f nur Kombinationen der g_i's auftauchen, deren Terme kleineren Multigrad als $\boldsymbol{\delta}$ besitzen. Dies widerspricht der Annahme eine Darstellung mit kleinstmöglichem $\boldsymbol{\delta}$ gewählt zu haben. □

Da wir uns nun über die Wichtigkeit der S-Polynome im Klaren sind, schreiben wir uns eine *Mathematica*-Funktion **`SPol`**, die das S-Polynom zweier Polynome `pol1` und `pol2` berechnet.

```
In[5]:= SPol[pol1_, pol2_, vars_,
           opts___?OptionQ] :=
         Module[{lcm},
           lcm = LCMMonomial[
               LT[pol1, vars, opts],
               LT[pol2, vars, opts],
               vars];
           ExpandPoly[
             (lcm/LT[pol1, vars, opts])pol1-
               (lcm/LT[pol2, vars, opts])pol2, vars]
         ];
```

Zuzüglich der Parameter `pol1` und `pol2`, die als Argumente die Polynome erhalten, deren S-Polynom berechnet werden soll, verlangt **`SPol`** über den Parameter `vars` eine Variablenliste `{x1, x2, ... }` zur Identifizierung der Polynomvariablen und deren Reihenfolge. Als viertes Argument können wir eine Monomordnung übergeben. Die Option ist identisch mit der auf Seite 14 von **`PolynomialReduce`**. Standardmäßig finden demzufolge die Berechnungen bezüglich *lex*-Ordnung statt.
Im Funktionsrumpf wird als erstes das kleinste gemeinsame Vielfache `lcm` $= \mathbf{x}^{\gamma} =$ kgV(LT(`pol1`), LT(`pol2`)) berechnet. Die hierzu aufgerufenen Funktionen **`LCMMonomial`** und **`LT`** sind im Anhang in den Abschnitten A.3 bzw. A.4 ab Seite 75 aufgelistet und erläutert. In ausmultiplizierter Form wird dann das S-Polynom, wie in Definition 7 beschrieben, zurückgeliefert. **`ExpandPoly`** (siehe A.2 Seite 75) multipliziert hierzu das Polynom nur bezüglich seiner Variablen aus und lässt die Koeffizienten unberührt.

Beispiel 2.3

Testen wir nun mit Buchbergers S-Paar Kriterium (Satz 8) und der Funktion **`SPol`** einige Erzeugendensysteme von Polynomidealen darauf, ob sie Gröbnerbasen sind.

1. In Beispiel 2.1 war die Rede davon, dass die Polynome $g_1 = x - z$, $g_2 = y - z$ aus $\mathbb{Q}[x, y, z]$ unter lexikographischer Ordnung eine Gröbnerbasis bilden. Prüfen wir dies nach:

   (a) Definiere die Polynome g_1 und g_2.

   ```
   In[6]:= g1 = x - z; g2 = y - z;
   ```

   (b) Berechne das S-Polynom s von g_1 und g_2.

   ```
   In[7]:= s = SPol[g1, g2, {x, y, z}]
   Out[7]= x z - y z
   ```

   (c) Dividiere s durch g_1 und g_2 und schaue den Rest an.

   ```
   In[8]:= PolynomialReduce[s, {g1, g2}, {x, y, z}]
   Out[8]= {{z, -z}, 0}
   ```

Als zweiter Eintrag in der Rückgabeliste von **PolynomialReduce** steht der Divisionsrest. Dieser ist Null und folglich bilden nach Satz 8 die Polynome g_1 und g_2 eine Gröbnerbasis.

2. Wie sieht es mit den Polynomen der *Twisted Cubic* (Beispiel 2.2) aus? Bilden $g_1 = y - x^2$ und $g_2 = z - x^3$ aus $\mathbb{R}[x, y, z]$ eine Gröbnerbasis?

```
In[9]:= g1 = y - x^2; g2 = z - x^3;

In[10]:= s = SPol[g1, g2, {x, y, z}]
Out[10]= z - x y

In[11]:= PolynomialReduce[s, {g1, g2}, {x, y, z}]
Out[11]= {{0, 0}, z - x y}
```

PolynomialReduce bestätigt, was das S-Polynom $s = -xy + z$ schon vermuten lässt. Da $\mathrm{LT}(s) = -xy$ weder durch $\mathrm{LT}(g_1) = -x^2$ noch durch $\mathrm{LT}(g_2) = -x^3$ teilbar ist, erhalten wir ein von Null verschiedenes Restpolynom. Wir haben also hier keine Gröbnerbasis vorliegen.
Es ist zudem egal, welche der drei uns bekannten Monomordnungen wir zugrundelegen. Denn die Leitterme von g_1, g_2 und s sind unter all diesen Ordnungen identisch.

3. Nun betrachten wir ein Beispiel mit drei Erzeugern. Sei $G = \{g_1, g_2, g_3\} = \{xz - y^2, xy - z, y - x^2\} \subset \mathbb{Q}[x, y, z]$. Wir prüfen diesmal, ob G unter *grlex*-Ordnung eine Gröbnerbasis bildet. Da

$$S(g_i, g_j) = S(g_j, g_i)$$

gilt, genügt es, wenn wir den Divisionsrest $\overline{S(g_i, g_j)}^G$ für $1 \leq i < j \leq 3$ berechnen.

```
In[12]:= g1 = y - x^2; g2 = x y - z; g3 = x z - y^2;
         G = {g1, g2, g3}
Out[12]= {y - x^2, x y - z, x z - y^2}

In[13]:= s12 = SPol[g1, g2, {x, y, z},
           MonomialOrder → DegreeLexicographic]
Out[13]= x z - y^2

In[14]:= PolynomialReduce[s12, G, {x, y, z},
           MonomialOrder → DegreeLexicographic]
Out[14]= {{0, 0, 1}, 0}
```

```
In[15]:= s13 = SPol[g1, g3, {x, y, z},
             MonomialOrder → DegreeLexicographic]
Out[15]= x y^2 - y z

In[16]:= PolynomialReduce[s13, G, {x, y, z},
           MonomialOrder → DegreeLexicographic]
Out[16]= {{0, y, 0}, 0}


In[17]:= s23 = SPol[g2, g3, {x, y, z},
             MonomialOrder → DegreeLexicographic]
Out[17]= y^3 - z^2

In[18]:= PolynomialReduce[s23, G, {x, y, z},
           MonomialOrder → DegreeLexicographic]
Out[18]= {{0, 0, 0}, y^3 - z^2}
```

Sofern nur einer der Divisionsreste ungleich dem Nullpolynom ist, so wie hier das zuletzt berechnete $r = y^3 - z^2$, bedeutet das, dass G bezüglich gradlexikographischer Ordnung keine Gröbnerbasis ist.

Für Erzeugendensysteme mit bis zu drei Polynomen mögen diese Berechnungen der Restpolynome bezüglich des Aufwandes noch akzeptabel sein. Umfasst eine Idealbasis dagegen zehn Polynome, wird man sicher schnell die Lust an dieser Verifikationstortur von 45 Berechnungsschritten verlieren.
Schreiben wir uns doch ein Programm, das uns die Arbeit abnimmt und entscheidet, ob ein Erzeugendensystem eine Gröbnerbasis ist.

```
In[19]:= IsGroebnerBasis[G_, vars_,
           opts___?OptionQ] :=
         Module[{B, s},
           B = Flatten[Table[{i, j},
                {i, 1, Length[G] - 1},
                {j, i + 1, Length[G]}], 1];
           s = Length[B];
           Do[
             If[
               PolynomialReduce[
                    SPol[G[[B[[i, 1]]]], G[[B[[i, 2]]]],
                      vars, opts], G, vars, opts][[2]] =!=
                 0,
               Return[Return[False]]
             ]
             , {i, 1, s}];
           True
         ];
```

An den Parameter **G** wird das Erzeugendensystem übergeben, das daraufhin überprüft wird, ob es eine Gröbnerbasis ist. Parameter **vars** legt wieder mittels einer übergebe-

nen Variablenliste die Ordnung der Variablen untereinander fest. Als Option können wir hier nicht nur die Monomordnung übergeben. Alle Optionen, die **PolynomialReduce** anbietet, werden unterstützt.
In der lokalen Variablen **B** werden die Polynomindexpaare für die S-Polynomberechnung abgelegt. Sind etwa t Basispolynome vorhanden, werden in **B** alle Tupel (i,j) mit $1 \leq i < j \leq t$ gespeichert. Variable **s** enthält die Anzahl dieser Indexpaare als Oberschranke für die zu berechnenden S-Polynome. Innerhalb der **Do**-Schleife wird sequentiell für jedes Polynompaar der Divisionsrest der S-Polynome durch die Erzeuger berechnet. Sobald solch ein Rest verschieden von Null ist, wird die Funktion mit dem Rückgabewert **False** verlassen. Nur wenn alle Divisionsreste verschwinden, liefert die Funktion den Wert **True** zurück.
Mit der etwas merkwürdig anmutenden Anweisung **Return[Return[False]]** umgehen wir eine Ungereimtheit der *Mathematica*-Funktion **Return[*ausdr*]**. Laut Dokumentation bewirkt die Anweisung die Rückgabe des Wertes ***ausdr*** und das Verlassen der Funktion. Befindet sich das **Return** jedoch in einer **Do**-Schleife, wird nur diese verlassen. Mit dem Doppel-**Return**-Konstrukt werden nun die Schleifendurchläufe beendet und auch wirklich die Funktion mit Rückgabewert **False** verlassen.

Fortsetzung Beispiel 2.3

4. Betrachten wir nochmals die Basis $G = \{xz - y^2, xy - z, y - x^2\}$ aus dem letztem Beispiel und prüfen nach, ob sie nicht vielleicht bezüglich einer anderen Monomordnung eine Gröbnerbasis bildet.

```
In[20]:= IsGroebnerBasis[G, {x, y, z},
           MonomialOrder → Lexicographic]
Out[20]= False


In[21]:= IsGroebnerBasis[G, {x, y, z},
           MonomialOrder →
             DegreeReverseLexicographic]
Out[21]= True
```

   Wie wir sehen, ist G bezüglich *grevlex*-Ordnung eine Gröbnerbasis.

5. Abschließend widmen wir uns noch einem umfangreicheren Beispiel. Sei $G \subset \mathbb{Q}[x,y,z,w]$ das aus folgenden neun Polynomen bestehende Erzeugendensystem:

$$
\begin{aligned}
&x^2w + y^2z,\\
&-xzw + y^2z,\\
&x^3w^3 + yz^3w^3,\\
&x^2w + xzw,\\
&-xyz^2w^3 - xz^3w^4,\\
&xy^2z + y^2z^2,\\
&-y^4z - y^2z^2w,\\
&-y^6z^2 + y^3z^4w^2,\\
&-yz^8w^4 - yz^7w^3.
\end{aligned}
$$

Nun ist G schon nicht mehr leicht zu überschauen und wer hätte jetzt die Lust 36 S-Polynome und deren Reste modulo G zu berechnen? Lassen wir unsere Funktion entscheiden, ob G unter einer uns bekannten Monomordnung eine Gröbnerbasis bildet.

```
In[22]:= IsGroebnerBasis[G, {x, y, z, w},
           MonomialOrder → Lexicographic]
Out[22]= False
In[23]:= IsGroebnerBasis[G, {x, y, z, w},
           MonomialOrder → DegreeLexicographic]
Out[23]= False
In[24]:= IsGroebnerBasis[G, {x, y, z, w},
           MonomialOrder →
             DegreeReverseLexicographic]
Out[24]= True
```

Wie oben ist G nur bezüglich der *grevlex*-Ordnung eine Gröbnerbasis.

Haben wir ein Ideal $I = \langle f_1, \dots, f_s \rangle \subset k[x_1, \dots, x_n]$ gegeben, so entscheidet die Funktion **IsGroebnerbasis** durch Prüfung der Disisionsreste $\overline{S(f_i, f_j)}^G$ auf Null, ob eine Gröbnerbasis vorliegt. Angenommen $\langle f_1, \dots, f_s \rangle$ ist keine Gröbnerbasis. Dann ist für mindestens zwei Polynome f_i, f_j das Restpolynom $\overline{S(f_i, f_j)}^G$ ungleich dem Nullpolynom. Dieser Rest ist aber ein Element des Ideals I, wie folgende Betrachtung leicht verdeutlicht.

Sei $s_{ij} = S(f_i, f_j) \in I$ und $r_{ij} = \overline{s_{ij}}^G$. Mit dem Divisionsalgorithmus erhalten wir eine Darstellung

$$s_{ij} = h_1 f_1 + \cdots + h_s f_s + r_{ij}.$$

Durch die Umformung in

$$r_{ij} = s_{ij} - h_1 f_1 - \cdots - h_s f_s$$

wird klar, dass der Rest r_{ij} ein Element von I ist.

Fügen wir r_{ij} zu unserer Erzeugermenge hinzu, so ist auch $r_{ij} = \overline{S(f_i, f_j)}^G = 0$. Aus diesen Betrachtungen lässt sich ein Algorithmus konstruieren.
Wir berechnen für sämtliche S-Polynome der Erzeugerpolynome die Divisionsreste. Sofern ein Rest verschieden von Null ist, fügen wir ihn der Erzeugermenge hinzu. Der Algorithmus bricht ab, wenn die Divisionsreste aller S-Polynome Null sind.
Natürlich drängen sich nun zwei Fragen auf:

- Terminiert dieser „Algorithmus“ überhaupt
- und wenn er dies tut, erhalten wir am Ende tatsächlich eine Gröbnerbasis?

Eine positive Antwort auf diese Fragen gibt uns folgender Satz, der diese Berechnungsvorschrift nochmals konkretisiert.

Satz 9 *Sei $I = \langle f_1, \dots, f_s \rangle \neq \{0\}$ ein Ideal aus $k[x_1, \dots, x_n]$. Eine Gröbnerbasis für I kann in endlich vielen Schritten mit folgendem Algorithmus konstruiert werden:*

Algorithmus 10 (Buchbergeralgorithmus)

Input: *$f_1, \dots, f_s \in k[x_1, \dots, x_n]$ und eine Monomordnung.*

Output: *Eine Gröbnerbasis $G = \{g_1, \dots, g_s\}$ für I bezüglich der übergebenen Monomordnung.*

1: $G := (f_1, \dots, f_s)$

2: *Wiederhole:*

3: $G' := G$

4: *Für jedes Paar $\{p, q\} \subset G'$ mit $p \neq q$ wiederhole:*

5: $s := \overline{S(p,q)}^{G'}$

6: *Ist $s \neq 0$, so füge s zu G hinzu.*

7: *Gilt $G = G'$, verlasse die Schleife und gib G zurück.*

Beweis
Zeigen wir zuerst, dass der Algorithmus eine Gröbnerbasis berechnet.
Zu Beginn des Algorithmus ist G eine Teilmenge von I. Da G nur mit modulo G reduzierten S-Polynomen, die auch Elemente von I sind (siehe Seite 16), erweitert wird, gilt dies auch in jedem Schritt des Algorithmus.
Er terminiert, wenn $G' = G$ gilt, d. h. wenn alle S-Polynome modulo G zu Null reduziert werden. Folglich ist G nach Satz 8 (Buchbergers S-Paar Kriterium) eine Gröbnerbasis.
Nun zeigen wir, dass der Algorithmus terminiert.
In der Hauptschleife besteht G aus G' (dem alten G) zuzüglich den von Null verschiedenen Resten der S-Polynome der Elemente von G'. Demnach gilt

$$\langle \mathrm{LT}(G') \rangle \subset \langle \mathrm{LT}(G) \rangle. \qquad (*)$$

Ist G' ungleich G, dann ist auch $\langle \mathrm{LT}(G') \rangle$ *echt* enthalten in $\langle \mathrm{LT}(G) \rangle$ nach Konstruktion der Reste, die zu G hinzugefügt werden. Mit (*) wird durch die Schleife eine aufsteigende Idealkette erzeugt, die nach Korollar 4 (ACC) aus Kapitel 1 nach endlich vielen Schritten stationär wird, also $G = G'$ gilt und der Algorithmus terminiert. □

Jetzt hält uns nichts mehr davon ab den Buchbergeralgorithmus in *Mathematica* zu implementieren, um endlich Gröbnerbasen berechnen zu können. Auf die Tatsache,

dass *Mathematica* mit der Funktion **GroebnerBasis** schon eine Version des Buchbergeralgorithmus implementiert hat, werden wir zunächst nicht näher eingegen. Dies verschieben wir auf Kapitel 3 und schreiben uns eine eigene Funktion. Dabei wandeln wir Algorithmus 10 ein wenig ab und implementieren ihn in einer leicht optimierten Form:

```
In[25]:= Buchberger[G_, vars_, opts___?OptionQ] :=
           Module[{Gb, s, B, sred},
             Gb = G;
             s = Length[Gb];
             B = Flatten[Table[{i, j}, {i, 1, s - 1},
                  {j, i + 1, s}], 1];
             While[Length[B] > 0,
               sred = PolynomialReduce[
                    SPol[Gb[[B[[1, 1]]]], Gb[[B[[1, 2]]]],
                      vars, opts],
                    Gb, vars, opts][[2]];
               If[sred =!= 0,
                 s++;
                 AppendTo[Gb, sred];
                 B = Join[B, Table[{i, s},
                       {i, 1, s - 1}]];
               ];
               B = Delete[B, 1];
             ];
             ExpandPoly[Gb, vars]
           ];
```

Die Funktion **Buchberger** erwartet die gleichen Argumente wie **IsGroebnerBasis**. Auch speichert **B** wieder die aus **IsGroebnerBasis** bekannten Polynomindexpaare. Abweichend von Algorithmus 10 wurde hier nur eine Schleife implementiert, die abbricht, wenn alle Indexpaare aus **B** berüchsichtigt und auch entsprechend daraus entfernt wurden. Abgearbeitet wird die Liste **B** vorerst sequentiell in der Reihenfolge, wie die Indexpaare erzeugt wurden. Im Abschnitt 2.4 werden wir später durch eine intelligentere Sortierung der Indexpaare in **B** den Algorithmus optimieren. Bezüglich des aus **B** gewählten Indexpaares werden nun die Polynome aus der Erzeugerliste **Gb** zur Berechnung des S-Polynoms herangezogen. In **sred** wird dann das S-Polynom modulo **Gb** abgespeichert. Ist dies von Null verschieden, wird es in die Liste **Gb** aufgenommen und die Indexpaarliste **B** wird mit allen Indexpaaren, die das hinzugefügte Polynom berücksichtigen, erweitert. Schlussendlich, wenn die Liste **B** leer ist, enthält **Gb** die Gröbnerbasis, die vom Algorithmus zurückgegeben wird.

Beispiel 2.4

Stürzen wir uns jetzt mit unserer neuen Funktion auf die Erzeugendensysteme aus Beispiel 2.3, um deren Gröbnerbasen zu berechnen.

1. In Beispiel 2.3.1 haben wir schon gesehen, dass $\{x-z, y-z\}$ eine Gröbnerbasis bezüglich *lex*-Ordnung ist. Dem fügt natürlich auch unser Buchbergeralgorithmus nichts mehr hinzu:

```
In[26]:= Buchberger[{x - z, y - z}, {x, y, z}]
Out[26]= {x - z, y - z}
```

2. Das Erzeugendensystem $G = \{y - x^2, z - x^2\}$, dessen Varietät die Twisted Cubic ist, bildet keine Gröbnerbasis, wie Beipiel 2.3.2 offenbarte. Berechnen wir jeweils eine für die drei uns bekannten Monomordnungen.

```
In[27]:= Buchberger[G, {x, y, z},
           MonomialOrder → Lexicographic]
Out[27]= {-x^2 + y, -x^3 + z, -x y + z, -y^2 + x z, y^3 - z^2}

In[28]:= Buchberger[G, {x, y, z},
           MonomialOrder → DegreeLexicographic]
Out[28]= {-x^2 + y, -x^3 + z, -x y + z, -y^2 + x z, y^3 - z^2}

In[29]:= Buchberger[G, {x, y, z},
           MonomialOrder →
             DegreeReverseLexicographic]
Out[29]= {-x^2 + y, -x^3 + z, -x y + z, -y^2 + x z}
```

   Es ist übrigens kein Zufall, dass die Gröbnerbasis bezüglich der *grevlex*-Ordnung kleiner ausfällt. In vielen Fällen ist das so, wie die Beispiele 2.3.3 - 2.3.5 bestätigen. Darum ist auch bei Gröbnerbasisberechnungen umfangreicherer Erzeugendensysteme diese Ordnung den anderen vorzuziehen, sofern die Monomordnung unerheblich ist.

3. Bisher hatten unsere betrachteten Polynome immer ganzzahlige Koeffizienten. Im folgenden Beispiel lassen wir auch rationale Funktionen als Koeffizienten zu. Dies wird über die Option **CoefficientDomain → RationalFunctions** realisiert, die von **PolynomialReduce** interpretiert wird. Alle Buchstabensymbole, die nicht in der Variablenliste als Argumente übergeben wurden, werden dann automatisch als Parameter der Koeffizienten aufgefasst.

```
In[30]:= G = {b z^4 + z, b x^2 + a x y, a x^3 y + 1/a x z};

In[31]:= Buchberger[G, {x, y, z},
           CoefficientDomain → RationalFunctions]
Out[31]= {z + b z^4, b x^2 + a x y, a x^3 y + x z/a,
           a^2 x y^3 z/b^3 + x z^2/(a^2 b), -a^2 x y^3/b^2 - x z/a^2}
```

4. Nehmen wir uns nun dem üppig ausfallenden Erzeugendensystem aus Beispiel 2.3.5 an. Vergleichen wir diesmal auch die Dauer der Gröbnerbasenberechnung

bezüglich der verschiedenen Ordnungen.

```
In[32]:= G = {x^2 w + y^2 z, -x z w + y^2 z, x^3 w^3 + y z^3 w^3,
              x^2 w + x z w, -x y z^2 w^3 - x z^3 w^4,
              x y^2 z + y^2 z^2, -y^4 z - y^2 z^2 w,
              -y^6 z^2 + y^3 z^4 w^2 , -y z^8 w^4 - y z^7 w^3};

In[33]:= (gbLex = Buchberger[G, {x, y, z, w},
                   MonomialOrder → Lexicographic])//
           Timing
Out[33]= {21.01 Second, {w x^2 + y^2 z,
             -w x z + y^2 z, w^3 x^3 + w^3 y z^3, w x^2 + w x z,
             -w^3 x y z^2 - w^4 x z^3, x y^2 z + y^2 z^2,
             -y^4 z - w y^2 z^2, -y^6 z^2 + w^2 y^3 z^4,
             -w^3 y z^7 - w^4 y z^8, -w^2 y^2 z^2 - w^3 y z^3,
             -w^4 y z^6 - w^5 y z^7, w^4 y z^5 + w^5 y z^6,
             w^4 y z^4 + w^5 y z^5, -w^3 y z^6 - w^4 y z^7,
             w^3 y z^5 + w^4 y z^6, -w^3 y z^4 - w^4 y z^5}}

In[34]:= gbLex//Length
Out[34]= 16

In[35]:= (gbGrlex = Buchberger[G, {x, y, z, w},
                    MonomialOrder → DegreeLexicographic])//
           Timing
Out[35]= {5.62 Second, {w x^2 + y^2 z,
             -w x z + y^2 z, w^3 x^3 + w^3 y z^3, w x^2 + w x z,
             -w^3 x y z^2 - w^4 x z^3, x y^2 z + y^2 z^2,
             -y^4 z - w y^2 z^2, -y^6 z^2 + w^2 y^3 z^4,
             -w^3 y z^7 - w^4 y z^8, w^2 y^2 z^3 - w^2 y^3 z^3}}

In[36]:= gbGrlex//Length
Out[36]= 10

In[37]:= (gbGrevlex = Buchberger[G, {x, y, z, w},
                      MonomialOrder →
                        DegreeReverseLexicographic])//
           Timing
Out[37]= {4.23 Second,
           {w x^2 + y^2 z, -w x z + y^2 z, w^3 x^3 + w^3 y z^3,
            w x^2 + w x z, -w^3 x y z^2 - w^4 x z^3,
            x y^2 z + y^2 z^2, -y^4 z - w y^2 z^2,
            -y^6 z^2 + w^2 y^3 z^4, -w^3 y z^7 - w^4 y z^8}}

In[38]:= gbGrevlex//Length
Out[38]= 9
```

In diesem Beispiel sind die Größenunterschiede der Gröbnerbasen nicht zu übersehen. Bezüglich der *lex*-Ordnung fügt unser Buchbergeralgorithmus dem Erzeugendensystem noch sieben Polynome hinzu und bei der *grlex*-Ordnung eins. Da die Eingangspolynome bezüglich der *grevlex*-Ordnung schon eine Gröbnerbasis bilden, wird das Erzeugendensystem auch nicht erweitert. Dementsprechend fallen auch die Unterschiede der Berechnungsdauern aus. Da die absoluten Werte rechnerspezifisch sind, setzen wir sie zueinander in Relation. Hier können wir vorsichtig von einem $15 : 4 : 3$ Verhältnis sprechen. Verallgemeinerbar ist das natürlich nicht, tendentiell ist es aber so, dass die Berechnungen unter *lex*-Ordnung am längsten dauern, während sie bezüglich der *grevlex*-Ordnung am schnellsten sind.
Wie hier zu sehen ist, ordnet *Mathematica* in der Ausgabe die Variablen in alphabetischer und nicht in der uns vorgegebenen Reihenfolge ($x > y > z > w$). Das soll uns aber nicht weiter stören, da die Berechnungen mit unserer übergebenen Variablenreihenfolge durchgeführt wurden.

5. Mit dem folgenden Beispiel betrachten wir noch eine eher unangenehme Eigenschaft von Gröbnerbasen, die sich vornehmlich unter lexikographischer Ordnung bemerkbar macht. Sei $G = \{17x^2y - 59xz^2, 75x^3 - 13y^2z^3, 31xyz + 83z^3\}$ das Erzeugendensystem, von dem wir bezüglich *lex*-Ordnung die Gröbnerbasis berechnen.

```
In[39]:= G = {17x^2y - 59x z^2, 75x^3 - 13y^2z^3,
              31x y z + 83x z^3};

         Buchberger[G, {x, y, z},
           MonomialOrder → Lexicographic]
```

$$\begin{aligned}\text{Out[39]}= \Big\{ & 17\,x^2\,y - 59\,x\,z^2,\ 75\,x^3 - 13\,y^2\,z^3,\\ & 31\,x\,y\,z + 83\,x\,z^3,\ -\frac{59}{17}\,x^2\,z^2 + \frac{13\,y^3\,z^3}{75},\\ & -\frac{59\,x\,z^3}{17} - \frac{18343\,y^3\,z^4}{137175},\\ & -\frac{13}{75}\,y^3\,z^4 - \frac{1079\,y^2\,z^6}{2325},\\ & \frac{221\,y^4\,z^3}{4425} - \frac{1522469\,y^2\,z^7}{4252425},\\ & -\frac{13}{75}\,y^2\,z^5 - \frac{3270573628760971\,y^2\,z^{12}}{33074023518163125}\Big\}\end{aligned}$$

Wir haben hier ein Erzeugendensystem aus Polynomen gewählt, deren Koeffizienten mal nicht hauptsächlich aus Einsen bestehen. Wie unschwer zu erkennen ist, haben die Polynome, die der Algorithmus diesem Erzeugendensystem nach und nach hinzufügt, immer „unschönere“ Koeffizienten. Zurückzuführen ist dies auf die Konstruktion der S-Polynome. Der hinzukommende Aufwand mit derartigen Koeffizienten umzugehen ist hierbei nicht zu unterschätzen. Dieses Beispiel ist aber noch harmlos und sollte nur einen Eindruck vermitteln.

Gröbnerbasen, die mit obigem Buchbergeralgorithmus berechnet wurden, sind meist größer als nötig. Ist etwa G eine Gröbnerbasis von $I \subset k[x_1, \ldots, x_n]$ und $g \in G$, so ist $G \backslash \{g\}$ auch eine Gröbnerbasis von I, sofern der Leitterm $\mathrm{LT}(g)$ ein Element des Monomideals $\langle \mathrm{LT}(G \backslash \{g\}) \rangle$ ist. Diese Tatsache motiviert folgende Definition.

Definition 11 (minimale Gröbnerbasis) *Eine minimale Gröbnerbasis eines Polynomideals I ist eine Gröbnerbasis G von I mit folgenden Eigenschaften:*

- $LC(g) = 1$ *für alle $g \in G$, d. h. die Polynome sind monisch.*
- *Für alle $g \in G$ gilt $LT(g) \notin \langle LT(G \backslash \{g\}) \rangle$.*

Wir können sogar die Basis noch weiter verkleinern. Zwar haben wir mit einer minimalen Gröbnerbasis die Anzahl der Erzeugerpolynome auf ein Minimum reduziert, da aber z. B. $\{x^2, xy\}$ und $\{x^2 + xy, xy\}$ zwei minimale Gröbnerbasen des selben Ideals sind, würden wir der Ersten wohl den Vorzug geben. Dazu die folgende Definition:

Definition 12 (reduzierte Gröbnerbasis) *Eine reduzierte Gröbnerbasis eines Polynomideals I ist eine Gröbnerbasis G von I mit folgenden Eigenschaften:*

- *Die Polynome sind wie bei minimalen Gröbnerbasen monisch.*
- *Für alle $g \in G$ liegt kein Monom von g in $\langle LT(G \backslash \{g\}) \rangle$, d. h. für jedes $g \in G$ gilt $g = \overline{g}^{G \backslash \{g\}}$.*

Reduzierte Gröbnerbasen besitzen folgende angenehme Eigenschaft.

Satz 13 *Sei $I \subset k[x_1, \ldots, x_n]$ ein von Null verschiedenes Polynomideal. Dann besitzt I bezüglich einer Monomordnung genau eine reduzierte Gröbnerbasis.*

Beweisskizze (Siehe [CLO98], Seite 90)
Sei G eine minimale Gröbnerbasis von $I \subset k[x_1, \ldots, x_n]$. Ist $g \in G$ bezüglich G reduziert, d. h. gilt $g = \overline{g}^{G \backslash \{g\}}$, dann ist g bezüglich jeder minimalen Gröbnerbasis von I reduziert, da sich die Reduziertheit nur auf die Leitterme bezieht.
Nehmen wir uns ein Element $g \in G$ und modifizieren mit $g' = \overline{g}^{G \backslash \{g\}}$ die minimale Gröbnerbasis G zu $G' = G \backslash \{g\} \cup \{g'\}$. Dann ist G' auch eine minimale Gröbnerbasis des Ideals I. Reduzieren wir nun alle Elemente von G auf diese Art, ist auch G reduziert.
Um die Eindeutigkeit zu zeigen, seien G und $\tilde{G}$ reduzierte Gröbnerbasen von I. Insbesondere sind diese Gröbnerbasen minimal, womit auch

$$\mathrm{LT}(G) = \mathrm{LT}(\tilde{G})$$

gilt, wie sich zeigen lässt. Seien $g \in G, \tilde{g} \in \tilde{G}$ mit $\mathrm{LT}(g) = \mathrm{LT}(\tilde{g})$. Um die Gleichheit $g = \tilde{g}$ zu zeigen, betrachten wir die Differenz $g - \tilde{g}$. Da $g - \tilde{g} \in I$ und G bzw. $\tilde{G}$ Gröbnerbasen sind, folgt $\overline{g - \tilde{g}}^{G} = 0$. Durch die Elimination der Leitterme wegen $\mathrm{LT}(g) = \mathrm{LT}(\tilde{g})$ ist kein Term dieser Differenz durch ein Monom aus der Leittermmenge

der reduzierten Gröbnerbasen $\mathrm{LT}(G) = \mathrm{LT}(\tilde{G})$ teilbar. Folglich gilt $\overline{g - \tilde{g}}^{G} = g - \tilde{g}$ also auch $g = \tilde{g}$. □

In *Mathematica* könnte die Berechnung reduzierter Gröbnerbasen wie folgt implementiert werden.

```
In[40]:= BuchbergerRed[G_, vars_,
           opts___?OptionQ] :=
         Module[{Gb, g, s, ltlist, lclist, lmlist},
           (*Berechnung einer minimalen
              GroebnerBasis*)
           Gb = Buchberger[G, vars, opts];
           ltlist = Map[LT[#1, vars, opts]&, Gb];
           {lclist, lmlist} =
            Transpose[ Map[SplitTerm[#1, vars]&,
               ltlist]];
           Gb = Expand[Gb/lclist];
           s = Length[lmlist];
           j = 1;
           Do[
            r = PolynomialReduce[lmlist[[j]],
                 Delete[lmlist, j], vars, opts][[2]];
            If[r === 0,
              lmlist = Delete[lmlist, j];
              Gb = Delete[Gb, j],
              j ++
            ]
            , {i, 1, s}];
           (*Berechnung der reduzierten
              GroebnerBasis*)
           s = Length[Gb];
           Do[
            g = PolynomialReduce[Gb[[i]],
                 Delete[Gb, i], vars, opts][[2]];
            Gb[[i]] = g
            , {i, 1, s}];
           ExpandPoly[Gb, vars]
         ];
```

In dieser Version von **BuchbergerRed** wird zunächst eine Gröbnerbasis **Gb** bezüglich der übergebenen Monomordnung berechnet. Würden wir über den Parameter **G** als Argument bereits eine Gröbnerbasis verlangen, wäre dies natürlich unnötig. Die Reduktion der Gröbnerbasis vollzieht sich in zwei Schritten. Erst wird die Basis minimalisiert, dann reduziert. Zur Minimalisierung wird mit Hilfe der erzeugten Leitmonomliste **lmlist** und Leitkoeffizientenliste **lclist** die Basis normiert und darauf alle Polynome g entfernt, deren Leitterme (jetzt Leitmonome) Elemente von $\langle \mathrm{LT}(\mathbf{Gb} \backslash g) \rangle$ sind. Die hierzu verwendete Funktion **SplitTerm** ist im Anhang A.5 aufgelistet und beschrieben. Um die Basis zu reduzieren, werden schließlich alle Polynome bezüglich der anderen reduziert, wie im Beweis zu Satz 13 beschrieben.

Mit der nun gewonnenen Eindeutigkeit reduzierter Gröbnerbasen können wir jetzt auch entscheiden, ob zwei Polynommengen dasselbe Ideal erzeugen:
Legen wir uns auf eine Monomordnung fest und berechnen die reduzierten Gröbnerbasen der Polynommengen. Diese erzeugen genau dann dasselbe Ideal, wenn die Gröbnerbasen übereinstimmen.

Beispiel 2.5

1. Mit dem Erzeugendensystem aus Beispiel 2.3.5 auf Seite 19 haben wir in Beispiel 2.4.4 auf Seite 25 ganz stattliche Gröbnerbasen bezüglich der drei Monomordnungen bestimmt. Schauen wir einmal, ob nicht auch weniger umfangreiche Gröbnerbasen zur Erzeugung des Ideals ausreichen, indem wir die reduzierten Gröbnerbasen berechnen.

```
In[41]:= BuchbergerRed[G, {x, y, z, w},
           MonomialOrder → Lexicographic]
Out[41]= {w x z - y^2 z, w x^2 + y^2 z,
           x y^2 z + y^2 z^2, y^4 z + w y^2 z^2,
           w^2 y^2 z^2 + w^3 y z^3, w^3 y z^4 + w^4 y z^5}
```

   Unter *lex*-Ordnung waren zehn Polynome überflüssig. Durch Streichung dieser erhalten wir eine aus sechs Polynomen bestehende minimale Gröbnerbasis. Im zweiten Teil des Algorithmus wird nun die Basis reduziert, indem das Erzeugerpolynom x^2w+xzw durch das lexikographisch kleinere Polynom x^2w+y^2z erzetzt wird. Alle weiteren Erzeuger sind bezüglich der anderen Polynome schon reduziert.

```
In[42]:= BuchbergerRed[G, {x, y, z, w},
           MonomialOrder → DegreeLexicographic]
Out[42]= {w x z - y^2 z, w^2 y^2 z^2 + w^3 y z^3,
           w x^2 + y^2 z, x y^2 z + y^2 z^2,
           y^4 z + w y^2 z^2, -w^2 y^2 z^3 + w^2 y^3 z^3}
```

   Bezüglich *grlex*-Ordnung wird die Gröbnerbasis, wie bei der *lex*-Ordnung, auf sechs Polynome minimiert, wobei hier sogar noch zwei Polynome reduziert werden.
   In reduzierter Form haben die Gröbnerbasen bezüglich *lex*- und *grlex*-Ordnung die gleiche Mächtigkeit. Die Berechnungsdauern waren jedoch sehr unterschiedlich, da unter lexikographischer Ordnung weit mehr Polynome dem Erzeugendensystem beigefügt wurden, auch wenn einige davon letztendlich überflüssig waren. Die *lex*-Ordnung ist aber für einige Berechnungen unentbehrlich, etwa für das Lösen polynomieller Gleichungssysteme (siehe Abschnitt 3.1), so dass wir auch nicht auf sie verzichten können. Es gibt aber ein Verfahren, mit dem sich Gröbnerbasen bezüglich *lex*-Ordnung aus Gröbnerbasen bezüglich einer anderen Monomordnung berechnen lassen, um die langen Berechnungszeiten zu umgehen. Der Algorithmus, der dieses leistet, wird mit „*GröbnerWalk*“ bezeich-

net. Wir wollen hier aber nicht näher darauf eingehen, sondern verweisen auf [CKM97]. *Mathematica* berechnet Gröbnerbasen bezüglich *lex*-Ordnung auch auf diese Art, wie in [Tra00] dokumentiert wird.

```
In[43]:= BuchbergerRed[G, {x, y, z, w},
           MonomialOrder →
             DegreeReverseLexicographic]
Out[43]= {-w x z + y^2 z, w^3 x z^2 + w^3 y z^3,
          w x^2 + w x z, w^3 x y z^2 + w^4 x z^3}
```

Unter *grevlex*-Ordnung lassen sich fünf Polynome eliminieren und eins reduzieren.

2. Jetzt schauen wir uns ein Beispiel zu dem oben aufgeführten Entscheidungsproblem über die Gleichheit von Idealen an. Seien $I = \langle G_I \rangle, J = \langle G_J \rangle, K = \langle G_K \rangle$, wobei $G_I, G_J, G_K \subset \mathbb{Q}[x, y, z]$ wie folgt definiert sind:

```
In[44]:= GI = {y^4 z^2 - 1, x^3 y + z, 1 + y^2 z};
         GJ = {x^3 + x^3 y - y z^2 - y^2 z^2, -x^3 + y z^2, 1 + y^2 z};
         GK = {x^3 + x^3 y - y z^2 - y^2 z^2, -x^3 + y z^2, x^3 y + z};
```

Wenn die reduzierten Gröbnerbasen übereinstimmen, erzeugen die G_i's dasselbe Ideal. Berechnen wir sie bezüglich *lex*-Ordnung.

```
In[45]:= BuchbergerRed[GI, {x, y, z}]
         BuchbergerRed[GJ, {x, y, z}]
         BuchbergerRed[GK, {x, y, z}]
Out[45]= {1 + y^2 z, x^3 - y z^2}
Out[45]= {x^3 - y z^2, 1 + y^2 z}
Out[45]= {x^3 - y z^2, z + y^2 z^2}
```

Folglich sind I und J identisch und von K verschieden. Bei genauerer Betrachtung der Idealerzeugerpolynome stellen wir fest, dass sämtliche Polynome aus G_K in G_I bzw. G_J enthalten sind. Schreiben wir eine Funktion, mit der wir dies verifizieren können.

```
In[46]:= IsSubIdeal[U_, G_, vars_,
           opts___?OptionQ] :=
         Module[{Gb},
           Gb = BuchbergerRed[G, vars, opts];
           Transpose[
                Map[PolynomialReduce[#, Gb, vars,
                     opts]&, U]][[2]] ===
             Table[0, {i, Length[U]}]
         ]
```

Diese Funktion prüft, ob $\langle U\rangle \subset \langle G\rangle$ gilt. Mit der Folgerung

$$u \in I \text{ für alle } u \in U \quad \Longrightarrow \quad \langle U\rangle \subset I$$

können wir die Berechnung auf das Lösen des *Ideal Membership Problems* beschränken. Das Polynom u ist genau dann ein Element von I, wenn $\overline{u}^G = 0$ gilt, sofern G eine Gröbnerbasis von I ist, wie in Satz 6 auf Seite 16 geschildert wurde. **`IsSubIdeal`** bestimmt folglich die Reste modulo G aller Polynome in U und liefert **`True`** zurück, sofern diese alle Null sind.

```
In[47]:= IsSubIdeal[G3, G1, {x, y, z}]
Out[47]= True
```

2.3 Buchbergers Verbesserungen des Algorithmus

Der Buchbergeralgorithmus, wie wir ihn im letzten Abschnitt eingeführt und programmiert haben, ist noch nicht sehr effizient und kann an einigen Stellen erheblich beschleunigt werden. Hierzu schauen wir uns in diesem Abschnitt zwei auf Buchberger zurückzuführende Kriterien an, die Aufschluss über die möglichen Resultate der Polynomreduktion geben, ohne sie tatsächlich durchzuführen. Für das erste Buchbergerkriterium benötigen wir ein allgemeineres Verständnis von dem Rest des Divisionsalgorithmus. Sei hierzu eine Monomordnung gewählt, $G = \{g_1, \ldots, g_t\} \subset k[x_1, \ldots, x_n]$ und $f \in k[x_1, \ldots, x_n]$. Wir sagen f *reduziert sich zu Null modulo G* und schreiben

$$f \to_G 0,$$

sofern sich f als Summe

$$f = a_1 g_1 + \cdots + a_t g_t$$

mit

$$\operatorname{mdeg}(f) \leq \operatorname{mdeg}(a_i g_i)$$

darstellen lässt, wobei $a_i \in k[x_1, \ldots, x_n]$. Die Beziehung zwischen dieser Definition und dem Divisionsalgorithmus ist leicht zu sehen. Gilt $\overline{f}^G = 0$, so gilt auch $f \to_G 0$. Andererseits lässt sich aus $f \to_G 0$ nicht das Verschwinden des Divisionsrestes modulo G folgern, sofern G keine Gröbnerbasis ist, da der Rest von der Divisionsreihenfolge der g_i's abhängt (siehe Beispiel 1.5 und 1.6 im Abschnitt 1.5). Schauen wir uns nun nochmals *Buchbergers S-Paar Kriterium* (Satz 8) an. Im Beweis dieses Satzes (siehe (4) und (5)) wurde statt $\overline{S(g_j, g_k)}^G = 0$ nur

$$S(g_j, g_k) = \sum_{i=1}^{t} a_{ijk}\, g_i$$

mit

$$\operatorname{mdeg}(a_{ijk}\, g_i) \leq \operatorname{mdeg}(S(g_j, g_k))$$

verwendet, was lediglich $f \to_G 0$ bedeutet. Folglich können wir *Buchbergers S-Paar Kriterium* wie folgt umformulieren:

Satz 14 *Eine Basis $G = \{g_1, \dots, g_t\}$ eines Ideals I ist genau dann eine Gröbnerbasis, wenn für alle $i \neq j$*

$$S(g_i, g_j) \to_G 0.$$

Mit dieser Vorarbeit können wir nun das erste Buchbergerkriterium formulieren. Hierzu betrachten wir zwei Polynome $f, g \in k[x_1, \dots, x_n]$, deren Leitterme relativ prim sind, d. h.

$$\text{kgV}(\text{LM}(f), \text{LM}(g)) = \text{LM}(f) \cdot \text{LM}(g).$$

Das S-Polynom dieser Polynome, die wir hierfür als $f = \text{LM}(f) + p$ und $g = \text{LM}(g) + q$ schreiben, hat dann folgende Gestalt:

$$\begin{aligned} S(f,g) &= \text{LM}(g) \cdot f - \text{LM}(f) \cdot g \\ &= (g - q) \cdot f - (f - p) \cdot g \\ &= p \cdot g - q \cdot f. \end{aligned} \tag{1}$$

Da das Leitmonom von g weder $\text{LM}(f)$ noch $\text{LM}(q)$ teilt, sind $\text{LM}(p \cdot g)$ und $\text{LM}(q \cdot f)$ verschieden und können sich im S-Polynom nicht wegheben. Folglich gilt

$$\text{mdeg}(S(f,g)) = \max\{\text{mdeg}(p \cdot g), \text{mdeg}(q \cdot f)\}. \tag{2}$$

(1) und (2) bedeuten aber nichts anderes als $S(f,g) \to_G 0$. Dies halten wir im folgenden Satz fest.

Satz 15 (erstes Buchbergerkriterium) *Sei $G \subset k[x_1, \dots, x_n]$ eine endliche Menge und $f, g \in G$. Gilt*

$$LCM(LM(f), LM(g)) = LM(f) \cdot LM(g),$$

dann lässt sich $S(f,g)$ modulo G zu Null reduzieren.

Hiermit ist uns nun ein Kriterium gegeben, mit dem wir zeitaufwändige Polynomdivisionen einsparen können. Wie gewohnt arbeiten wir die Indexpaarliste $\mathtt{B} = \{(i,j) : 1 \leq i < j \leq t\}$, wobei t die Anzahl der Erzeugerpolynome ist, ab. Jedoch berechnen und reduzieren wir ein S-Polynom der Polynome g_i und g_j mit $(i,j) \in \mathtt{B}$ nur, wenn $\text{LM}(g_i)$ und $\text{LM}(g_j)$ relativ prim sind. Denn mit obigem Satz folgt, dass dann das S-Polynom zu Null reduziert wird und wir eine Polynomdivision einsparen können. Das klingt zwar noch nicht so spektakulär, aber der Divisionsalgorithmus würde hier nicht notwendigerweise als Rest Null berechnen, was eine unnötige Aufnahme dieses Restes in die Basis und damit weitere Polynomdivisionen nach sich ziehen würde.
Wegen der hohen Berechnungsdauer reduzierter S-Polynome bezüglich der Indexpaare aus **B**, werden diese Paare auch *kritische Paare* genannt.
Implementieren wir nun das erste Buchbergerkriterium in unseren Buchbergeralgorithmus. Wir nennen diesen neuen Algorithmus **`BuchbergerK1`**.

```
In[48]:= BuchbergerK1[G_, vars_, opts___?OptionQ] :=
           Module[{Gb = G, s = Length[G], B, f, g,
               LMf, LMg, lcm, sred},
             B = Flatten[Table[{i, j}, {i, 1, s - 1},
                  {j, i + 1, s}], 1];
    While[Length[B] > 0,
      {f, g} = Gb[[B[[1]]]];
      B = Delete[B, 1];
      LMf = SplitTerm[LT[f, vars, opts],
           vars][[2]];
      LMg = SplitTerm[LT[g, vars, opts],
           vars][[2]];
      lcm = LCMMonomial[LMf, LMg, vars];
      If[lcm - LMf * LMg =!= 0,
        sred =
          PolynomialReduce[
             SPol[f, g, vars, opts], Gb, vars,
             opts][[2]];
        If[sred =!= 0,
          s++;
          AppendTo[Gb, sred];
          B = Join[B, Table[{i, s},
               {i, 1, s - 1}]]
        ]]];
    ExpandPoly[Gb, vars]
  ];
```

Beispiel 2.6

Vergleichen wir nun unserere Buchbergeralgorithmen anhand eines Erzeugendensystems, das auch ein paar Polynome mit relativ primen Leitmonomen enthält.

```
In[49]:= G = {b z^4 + z, b/a x^2 + a x y, a x^3 y + x z + c z};
```

```
In[50]:= Buchberger[G, {x, y, z},
            CoefficientDomain → RationalFunctions]//
          Timing
```

$$\text{Out[50]= } \Big\{7.33\,\text{Second},\ \Big\{z + b\,z^4,\ \frac{b\,x^2}{a} + a\,x\,y,\ a\,x^3\,y + c\,z + x\,z,\ \frac{a^4\,x\,y^3\,z}{b^3} + \frac{c\,z^2}{a\,b} + \frac{x\,z^2}{a\,b},\ -\frac{a^4\,x\,y^3}{b^2} - \frac{c\,z}{a} - \frac{x\,z}{a},\ -\frac{b^2\,c\,x\,z^2}{a^5} - \frac{b\,c\,y\,z^2}{a^3},\ -\frac{b^2\,c\,x\,z}{a^5} - \frac{b\,c\,y\,z}{a^3},\ -\frac{a^2\,y^4\,z^2}{b} + \frac{b^2\,c\,z^3}{a^5} - \frac{b\,y\,z^3}{a^3},\ -\frac{a^2\,y^4\,z}{b} + \frac{b^2\,c\,z^2}{a^5} - \frac{b\,y\,z^2}{a^3}\Big\}\Big\}$$

```
In[51]:= BuchbergerK1[G, {x, y, z},
           CoefficientDomain → RationalFunctions]//
         Timing
```

$$\text{Out[51]= } \Big\{1.74\,\text{Second},\ \Big\{z + b\,z^4,\ \frac{b\,x^2}{a} + a\,x\,y,\ a\,x^3\,y + c\,z + x\,z,\ -\frac{a^4\,x\,y^3}{b^2} - \frac{c\,z}{a} - \frac{x\,z}{a},\ -\frac{b^2\,c\,x\,z}{a^5} - \frac{b\,c\,y\,z}{a^3},\ -\frac{a^2\,y^4\,z}{b} + \frac{b^2\,c\,z^2}{a^5} - \frac{b\,y\,z^2}{a^3}\Big\}\Big\}$$

Der Buchbergeralgorithmus berechnete bezüglich *lex*-Ordnung für G in 7,3 Sekunden eine Gröbnerbasis mit 9 Polynomen. Demnach wurden 36 Divisionen durchgeführt. Die Gröbnerbasis, die unser optimierter Buchbergeralgorithmus erzeugte, besitzt drei Polynome weniger, er benötigte also höchstens 15 Divisionen. Tatsächlich kam er mit 11 Divisionen aus und brauchte für die ganze Berechnung etwa 1,7 Sekunden. Der Algorithmus ist zwar komplexer geworden, aber diese Zahlen untermauern, dass sich der Mehraufwand, das Testen auf relative Primalität der Polynome, zeitmäßig lohnt.

Schauen wir uns nun Buchbergers zweites Kriterium an. Auch mit diesem können wir wieder zeitintensive Divisionen einsparen. Doch ist dieses noch tiefgreifender als das erste Kriterium. Es besagt, dass wir bestimmte S-Polynome ignorieren können, auch wenn sie zu dem Zeitpunkt ihrer Betrachtung noch gar nicht zu Null reduziert werden können. Dazu schauen wir uns zunächst die zugrundeliegende Theorie an, um ein weiteres mal Buchbergers S-Paar Kriterium verallgemeinern zu können.
Im Beweis dieses Kriteriums (Satz 8 auf Seite 17) sind die S-Polynome zur Beseitigung der Leitterme konstruiert worden. Diese „Beseitigung“ schauen wir uns nun in allgemeinerer Form an, indem wir Syzygien studieren.

Definition 16 (Syzygien) *Sei $F = (f_1, \dots, f_t)$ ein t-Tupel mit Polynomen aus dem Ring $k[x_1, \dots, x_n]$ als Komponenten. Wir bezeichnen mit*

$$LT(F) := (LT(f_1), \dots, LT(f_t))$$

das Leittermtupel von F. Eine Syzygie der Leitterme $LT(f_1), \dots, LT(f_t)$ von F ist ein t-Tupel $S = (s_1, \dots, s_t) \in k[x_1, \dots, x_n]^t$ mit

$$S \bullet LT(F) = \sum_{i=1}^{t} s_i \cdot LT(f_i) = 0.$$

$\mathcal{S}(F)$ bezeichne die Menge aller Syzygien der Leitterme von F.
Bestehen die Komponenten von $S \in \mathcal{S}(F)$ nur aus Termen

$$S = (c_1 \mathbf{x}^{\alpha(1)}, \dots, c_t \mathbf{x}^{\alpha(t)}),$$

*mit $c_i \in k$ und $\boldsymbol{\alpha}(i) + \mathrm{mdeg}(f_i) = \boldsymbol{\alpha}$ für $c_i \neq 0$, so nennen wir S **homogen vom Multigrad** $\boldsymbol{\alpha} \in \mathbb{N}^n$.*

Beispiel 2.7

1. Sei $F = (x^2y + x, xz^2 - yz, y^3 + z) \in \mathbb{Q}[x, y, z]^3$, also $\mathrm{LT}(F) = (x^2y, xz^2, y^3)$ unter *lex*-Ordnung. Dann erhalten wir etwa folgende Syzygien:

   - $S_1 = (0, 0, 0)$
   - $S_2 = (y^2, 0, x^2)$
   - $S_3 = (0, x^4y^6z^{73}, x^5y^3z^{75})$
   - $S_4 = (y^2z - z^2, xy, -x^2z)$

   S_1 ist die triviale Syzygie. S_2 und S_3 sind jeweils homogene Syzygien vom Multigrad $\alpha = (2, 3, 0)$ bzw. $\alpha = (5, 6, 75)$. Wegen

   $$(y^2z - z^2) \cdot \mathrm{LT}(x^2y + x) + xy \cdot \mathrm{LT}(xz^2 - yz) + (-x^2z) \cdot \mathrm{LT}(y^3 + z) = 0$$

   folgt, dass auch $S_4 \in \mathcal{S}(F)$. Da aber die erste Komponente von S_4 kein Monom ist, ist diese Syzygie inhomogen.

2. Sei $\mathbf{e}_i \in k[x_1, \ldots, x_n]^t$ der i-te Einheitsvektor. Dann lässt sich $S \in \mathcal{S}(F)$ in der Form

   $$S = \sum_{i=1}^{t} s_i \mathbf{e}_i$$

   schreiben. Betrachten wir nun für $\{f_i, f_j\} \subset F$ und $\mathbf{x}^\gamma = \mathrm{kgV}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))$ folgende Konstruktion:

   $$S_{ij} := \frac{\mathbf{x}^\gamma}{\mathrm{LT}(f_i)}\mathbf{e}_i - \frac{\mathbf{x}^\gamma}{\mathrm{LT}(f_j)}\mathbf{e}_j. \tag{2.2}$$

   Diese aus den S-Polynomen konstruierte Darstellung bildet auch eine Syzygie, wie leicht zu sehen ist. Ihr verdanken übrigens die S-Polynome ihren Namen, denn S-Polynom ist die Abkürzung des Wortes *Syzygienpolynom* und es gilt die Beziehung

   $$S_{ij} \bullet F = S(f_i, f_j).$$

   Der Einfachheit halber setzen wir $S_{ij} := S_{ji}$, falls $j < i$.

Seien $S_1 = (s_1, \ldots, s_t), S_2 = (u_1, \ldots, u_t) \in \mathcal{S}(F)$ und $f \in k[x_1, \ldots, x_n]$. Aus den leicht zu verifizierenden Eigenschaften

- $S_1 + S_2 := (s_1 + u_1, \ldots, s_t + u_t) \in \mathcal{S}(F)$ (Vektoraddition)
- $f \cdot S_1 := (f \cdot s_1, \ldots, f \cdot s_t) \in \mathcal{S}(F)$ (skalare Multiplikation)

folgt, dass die Menge der Syzygien $\mathcal{S}(F)$ einen $k[x_1, \ldots, x_n]$-Modul bildet. Demnach existiert auch eine Basis $T \subset \mathcal{S}(F)$ für diesen Modul. Wie wir eine solche Basis finden können, schauen wir uns im Folgenden an.

Seien $S = (s_1, \dots, s_t) \in \mathcal{S}(F), \boldsymbol{\alpha} \in \mathbb{N}^n$ und $s_{i\boldsymbol{\alpha}}$ der Term von s_i (sofern einer existiert), so dass $\mathrm{mdeg}(s_{i\boldsymbol{\alpha}} \cdot f_i) = \boldsymbol{\alpha}$. Dann gilt für $S_{\boldsymbol{\alpha}} := (s_{1\boldsymbol{\alpha}}, \dots, s_{t\boldsymbol{\alpha}})$

$$S_{\boldsymbol{\alpha}} \bullet \mathrm{LT}(F) = \sum_{i=1}^{t} s_{i\boldsymbol{\alpha}} \mathrm{LT}(f_i) = 0.$$

Die Komponenten von $S_{\boldsymbol{\alpha}}$ bestehen nur aus Termen $c_i \mathbf{x}^{\boldsymbol{\alpha}(i)}$, womit $S_{\boldsymbol{\alpha}}$ homogen ist. Da

$$S = \sum_{\boldsymbol{\alpha}} S_{\boldsymbol{\alpha}}$$

gilt, haben wir gezeigt, dass sich jede Syzygie in homogene Syzygien zerlegen lässt. Diese Zerlegung ist zudem eindeutig. Sei etwa $S = \sum_{\boldsymbol{\alpha}} \tilde{S}_{\boldsymbol{\alpha}}$ eine weitere Darstellung von S, wobei $\tilde{S}_{\boldsymbol{\alpha}}$ homogene Syzygien mit Multigrad $\boldsymbol{\alpha}$ sind. Jede i-te Komponente der $\tilde{S}_{\boldsymbol{\alpha}}$'s ist nun entweder Null, oder hat Multigrad $\boldsymbol{\alpha} - \mathrm{mdeg}(f_i)$. Damit entsprechen aber die $\tilde{S}_{\boldsymbol{\alpha}}$'s genau den $S_{\boldsymbol{\alpha}}$'s.
Mit folgendem Satz erhalten die S_{ij}-Syzygien aus Beispiel 2.7.2 den Status von Basispolynomen.

Satz 17 *Für ein gegebenes $F = (f_1, \dots, f_t)$ lässt sich jede Syzygie $S \in \mathcal{S}(F)$ als*

$$S = \sum_{i<j} h_{ij} S_{ij},$$

mit $h_{ij} \in k[x_1, \dots, x_n]$ und $S_{ij} = \frac{\mathbf{x}^{\boldsymbol{\gamma}}}{LT(f_i)} \mathbf{e}_i - \frac{\mathbf{x}^{\boldsymbol{\gamma}}}{LT(f_j)} \mathbf{e}_j$ (siehe (2.2)) schreiben.

Beweis
Nach obigen Betrachtungen können wir $S = (s_1, \dots, s_t)$ als homogen mit Multigrad $\boldsymbol{\alpha}$ annehmen. S hat mindestens zwei von Null verschiedene Komponenten, sonst wäre die Summe $\sum_{i=1}^{t} s_i \cdot \mathrm{LT}(f_i)$ nicht Null. Seien zwei dieser Komponenten etwa $s_i = c_i \mathbf{x}^{\boldsymbol{\alpha}(i)}$ und $s_j = c_j \mathbf{x}^{\boldsymbol{\alpha}(j)}$ mit $i < j$. Aus $\boldsymbol{\alpha}(i) + \mathrm{mdeg}(f_i) = \boldsymbol{\alpha}(j) + \mathrm{mdeg}(f_j) = \boldsymbol{\alpha}$ und $\mathbf{x}^{\boldsymbol{\gamma}} = \mathrm{kgV}(\mathrm{LM}(f_i), \mathrm{LM}(f_j))$ folgt, dass $\mathbf{x}^{\boldsymbol{\alpha}}$ ein Vielfaches von $\mathbf{x}^{\boldsymbol{\gamma}}$ ist. Betrachten wir nun die i-te Komponente des Ausdrucks $S' := S - c_i \mathrm{LC}(f_i) \mathbf{x}^{\boldsymbol{\alpha}-\boldsymbol{\gamma}} S_{ij}$:

$$\begin{aligned} S'_i &= s_i - c_i \mathrm{LC}(f_i) \mathbf{x}^{\boldsymbol{\alpha}-\boldsymbol{\gamma}} \left(\frac{\mathbf{x}^{\boldsymbol{\gamma}}}{\mathrm{LT}(f_i)} \right) \\ &= c_i \mathbf{x}^{\boldsymbol{\alpha}(i)} - c_i \frac{\mathbf{x}^{\boldsymbol{\alpha}}}{\mathrm{LM}(f_i)} \\ &= c_i \mathbf{x}^{\boldsymbol{\alpha}(i)} - c_i \mathbf{x}^{\boldsymbol{\alpha}(i)} \\ &= 0 \end{aligned}$$

S' unterscheidet sich von S nur in der i-ten und j-ten Komponente. Wegen $S'_i = 0$, haben wir aus S eine Syzygie erstellt, die mehr Nulleinträge enthält als S selbst. Da bei nicht trivialen Syzygien, wie oben erwähnt, immer zwei oder mehr Komponenten von Null verschieden sind, ist entweder S' der Nullvektor, also $S = \left(c_i \mathrm{LC}(f_i) \mathbf{x}^{\boldsymbol{\alpha}-\boldsymbol{\gamma}}\right) \cdot S_{ij}$, oder wir führen obigen Vorgang fort, etwa mit $S'' = S' - c_j \mathrm{LC}(f_j) \mathbf{x}^{\boldsymbol{\alpha}-\boldsymbol{\gamma}} S_{jk}$ mit $j \neq k$ (usw.) und erhalten nach endlich vielen Schritten eine Kombination aus den S_{ij}'s als

Darstellung für S. □

Für eine Basis von $\mathcal{S}(F)$ brauchen wir aber nicht unbedingt alle Syzygien S_{ij}, wie das folgende Beispiel zeigt.

Beispiel 2.8

Sei $F = (xyz + y^2, x^2z^3 - xy, y^3z + z) \in \mathbb{Q}[x,y,z]^3$, also $\mathrm{LT}(F) = (xyz, x^2z^3, y^3z)$ bezüglich *lex*-Ordnung und es ergeben sich korrespondierend zu den S-Polynomen die drei Syzygien

- $S_{12} = (xz^2, -y, 0)$,
- $S_{13} = (y^2, 0, -x)$,
- $S_{23} = (0, y^3, -x^2z^2)$.

Da $S_{23} = y^2 \cdot S_{12} - xz^2 \cdot S_{13}$ gilt, bildet $\{S_{12}, S_{13}\}$ auch schon eine Basis für $\mathcal{S}(F)$.

Jetzt ist es an der Zeit, Buchbergers S-Paar Kriterium wieder einmal zu verallgemeinern.

Satz 18 *Eine Basis $G = (g_1, \ldots, g_t)$ eines Ideals I ist genau dann eine Gröbnerbasis, wenn für alle $S = (s_1, \ldots, s_t)$ einer homogenen Basis der Syzygien $\mathcal{S}(G)$*

$$S \bullet G = \sum_{i=1}^{t} s_i g_i \to_G 0.$$

Beweis
$\Longrightarrow$: Aus Satz 14 folgt $S(g_i, g_j) \to_G 0$, also $S_{ij} \bullet G \to_G 0$ mit S_{ij} aus (2.2). Nach Satz 17 bilden die S_{ij}'s eine homogene Basis von $\mathcal{S}(G)$.
$\Longleftarrow$: In Anlehnung an die Beweisskizze des Satzes 8 seien $f = \sum_{i=1}^{t} h_i g_i$ eine Darstellung von f und $\boldsymbol{\delta} = \max\{m(i)\}$ minimal unter allen möglichen Arten f mit den g_i's darzustellen. Auch hier führen wir mit $\mathrm{mdeg}(f) < \boldsymbol{\delta}$ einen Widerspruch herbei.
Nach (3) in diesem Beweis (Seite 17) impliziert $\mathrm{mdeg}(f) < \boldsymbol{\delta}$, dass $\sum_{m(i)=\boldsymbol{\delta}} \mathrm{LT}(h_i) g_i$ auch einen echt kleineren Multigrad als $\boldsymbol{\delta}$ besitzt. Damit ist $\sum_{m(i)=\boldsymbol{\delta}} \mathrm{LT}(h_i)\mathrm{LT}(g_i) = 0$ und

$$S = \sum_{m(i)=\boldsymbol{\delta}} \mathrm{LT}(h_i)\mathbf{e}_i$$

eine homogene Syzygie in $\mathcal{S}(G)$. Nach Voraussetzung existiert eine homogene Basis $\{S_1, \ldots, S_m\}$ von $\mathcal{S}(G)$ mit $S_j \bullet G \to_G 0$ für alle j. Folglich hat S eine Darstellung

$$S = u_1 S_1 + \cdots + u_m S_m \tag{1}$$

mit $u_j \in k[x_1, \ldots, x_n]$. Schreiben wir die u_j's als Summe von Termen und multiplizieren sie mit den S_j's aus, um S als Summe homogener Syzygien auszudrücken. Dann

können wir in (1) alle Syzygien unberücksichtigt lassen, deren Multigrad kleiner als $\boldsymbol{\delta}$ ist, da einerseits S selbst homogen vom Multigrad $\boldsymbol{\delta}$ ist und eine solche Darstellung eindeutig ist (wie auf Seite 38 geschildert wurde).
Für jedes j ist also u_j entweder Null oder $u_j s_j$ ist homogen vom Multigrad $\boldsymbol{\delta}$. Hat S_j den Multigrad $\boldsymbol{\gamma}_j$, können wir u_j in der Form $u_j = c_j \mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_j}$ mit $c_j \in k$ schreiben, sofern $u_j \neq 0$. Folglich kann (1) auch als

$$S = \sum_j c_j \mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_j} S_j$$

geschrieben werden, wobei über die j's summiert wird, für die $u_j \neq 0$. Durch Skalarmultiplikation mit G ergibt sich

$$\begin{aligned}\sum_{m(i)=\boldsymbol{\delta}} \mathrm{LT}(h_i) g_i &= S \bullet G \\ &= \sum_j c_j \mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_j} S_j \bullet G.\end{aligned}$$

Nach Voraussetzung gilt $S_j \bullet G \to_G 0$, also

$$S_j \bullet G = \sum_{i=1}^{t} a_{ij} g_i,$$

mit $a_{ij} \in k[x_1, \ldots, x_n]$ und

$$\mathrm{mdeg}(a_{ij} g_i) \leq \mathrm{mdeg}(S_j \bullet G) \tag{2}$$

in Analogie zu (4) und (5) aus dem Beweis von Satz 8. Für obige Gleichungskette folgt weiter

$$\begin{aligned}&= \sum_j c_j \mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_j} \sum_i a_{ij} g_i \\ &= \sum_{i,j} c_{ij} b_{ij} g_i \qquad \text{mit} \quad b_{ij} = \mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_j} a_{ij} \\ &= \sum_i \tilde{h}_i g_i \qquad \text{mit} \quad \tilde{h}_i = \sum_j c_{ij} b_{ij}\,.\end{aligned}$$

Für den Multigrad ergibt sich wegen (2)

$$\mathrm{mdeg}(b_{ij} g_i) \leq \mathrm{mdeg}(\mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_j} S_j \bullet G).$$

Da $S_j \bullet \mathrm{LT}(G) = 0$ und S_j homogen vom Multigrad $\boldsymbol{\gamma}_j$, folgt

$$\mathrm{mdeg}(\tilde{h}_i g_i) = \mathrm{mdeg}(\mathbf{x}^{\boldsymbol{\delta}-\boldsymbol{\gamma}_j} S_j \bullet G) < \boldsymbol{\delta}.$$

So haben wir wieder eine widerspruchshervorrufende Darstellung von f gefunden, in der nur Kombinationen der g_i's mit Multigrad kleiner als $\boldsymbol{\delta}$ auftauchen. □

Wir brauchen also nicht mehr alle S-Polynome $S(g_i, g_j)$ (mit nicht relativ primem g_i

und g_j) im Buchbergeralgorithmus zu berücksichtigen, sondern nur diejenigen, deren Syzygien eine Basis von $\mathcal{S}(G)$ bilden. Wie uns Beispiel 2.8 zeigt, sind dafür nicht alle S_{ij}'s als Erzeuger von Nöten. Entwickeln wir nun ein Entscheidungskriterium, das uns darüber Auskunft gibt, welche S_{ij}'s wir ignorieren können.
Seien $g_i, g_j, g_k \in G = (g_1, \ldots, g_t)$, $\mathbf{x}^{\gamma_{ij}} = \mathrm{kgV}(\mathrm{LM}(g_i), \mathrm{LM}(g_j))$ und $\mathbf{x}^{\gamma_{ik}}$ bzw. $\mathbf{x}^{\gamma_{jk}}$ entsprechend definiert. Ist $\mathbf{x}^{\gamma_{ij}}$ ein Vielfaches von $\mathrm{LM}(g_k)$, so natürlich auch von $\mathbf{x}^{\gamma_{ik}}$ bzw. $\mathbf{x}^{\gamma_{jk}}$. Wir rechnen:

$$\begin{aligned}
S_{ij} &= \frac{\mathbf{x}^{\gamma_{ij}}}{\mathrm{LT}(g_i)}\mathbf{e}_i - \frac{\mathbf{x}^{\gamma_{ij}}}{\mathrm{LT}(g_j)}\mathbf{e}_j \\
&= \frac{\mathbf{x}^{\gamma_{ij}}}{\mathrm{LT}(g_i)}\mathbf{e}_i - \frac{\mathbf{x}^{\gamma_{ij}}}{\mathrm{LT}(g_k)}\mathbf{e}_k - \frac{\mathbf{x}^{\gamma_{ij}}}{\mathrm{LT}(g_j)}\mathbf{e}_j + \frac{\mathbf{x}^{\gamma_{ij}}}{\mathrm{LT}(g_k)}\mathbf{e}_k \\
&= \frac{\mathbf{x}^{\gamma_{ij}}}{\mathbf{x}^{\gamma_{ik}}}\left(\frac{\mathbf{x}^{\gamma_{ik}}}{\mathrm{LT}(g_i)}\mathbf{e}_i - \frac{\mathbf{x}^{\gamma_{ik}}}{\mathrm{LT}(g_k)}\mathbf{e}_k\right) - \frac{\mathbf{x}^{\gamma_{ij}}}{\mathbf{x}^{\gamma_{jk}}}\left(\frac{\mathbf{x}^{\gamma_{jk}}}{\mathrm{LT}(g_j)}\mathbf{e}_j - \frac{\mathbf{x}^{\gamma_{jk}}}{\mathrm{LT}(g_k)}\mathbf{e}_k\right) \\
&= \frac{\mathbf{x}^{\gamma_{ij}}}{\mathbf{x}^{\gamma_{ik}}}S_{ik} - \frac{\mathbf{x}^{\gamma_{ij}}}{\mathbf{x}^{\gamma_{jk}}}S_{jk}
\end{aligned}$$

Unter obigen Annahmen lässt sich demnach S_{ij} durch S_{ik} und S_{jk} darstellen. Folglich ist S_{ij} als Erzeuger überflüssig, sofern S_{ik} und S_{jk} schon Basiselemente sind. Dieses Resultat wollen wir im folgenden Satz festhalten.

Satz 19 (zweites Buchbergerkriterium) *Zu $G = (g_1, \ldots, g_t)$ sei die Menge $T \subset \{S_{ij} : 1 \le i < j \le t\}$ eine Basis von $\mathcal{S}(G)$ und seien $g_i, g_j, g_k \in G$, so dass $LM(g_k)$ das Monom $\mathbf{x}^{\gamma_{ij}} := \mathrm{kgV}(LM(g_i), LM(g_j))$ teilt. Sind $S_{ik}, S_{jk} \in T$, dann ist $T\backslash\{S_{ij}\}$ auch eine Basis von $\mathcal{S}(G)$.*

Verbessern wir nun den **`BuchbergerK1`**-Algorithmus um unsere hinzugewonnenen Erkenntnisse über Syzygien. Dabei seien

$$[i,j] := \begin{cases} (i,j) & \text{für } i < j, \\ (j,i) & \text{für } i > j, \end{cases} \qquad \text{bzw.} \qquad S[i,j] := \begin{cases} S(i,j) & \text{für } i < j, \\ S(j,i) & \text{für } i > j. \end{cases}$$

Wir verschonen bei der Gröbnerbasisberechnung ein S-Polynom $S(g_i, g_j)$ vor der zeitaufwändigen Division, wenn entweder g_i und g_j relativ prim sind (Kriterium 1), oder ein Index $k \notin \{i, j\}$ existiert, für den einerseits $[i,k]$ und $[j,k]$ nicht in der Indexpaarliste **`B`** enthalten sind (d. h. die S-Polynome $S[i,k]$ und $S[j,k]$ sind für die Gröbnerbasis schon berücksichtigt wurden) und andererseits $\mathrm{LT}(g_k)$ das Monom $\mathbf{x}^{\gamma_{ij}} = \mathrm{kgV}(\mathrm{LM}(g_i), \mathrm{LM}(g_j))$ teilt (Kriterium 2).
Der Programmcode zur Überprüfung des zweiten Kriteriums wird gegenüber dem Code des ersten komplexer sein. Wir lagern ihn der Übersichtlichkeit wegen in einer Funktion namens **`Criterion`** aus. Um **`Criterion`** vollen Zugriff auf sämtliche lokalen Variablen von **`BuchbergerK2`** zu ermöglichen, ohne ihr eine lange Parameterliste aufzuzwängen, implementieren wir sie als interne Funktion.

```
In[52]:= BuchbergerK2[G_, vars_, opts___?OptionQ] :=
           Module[{Gb = G, t = Length[G], B1, B2 = {},
               f, g, LMf, LMg, lcm, sred},
             Criterion[] := Module[{},
                 Do[
                   If[
                     MemberQ[B2,
                         Sort[{B1[[1, 1]], k}]] &&
                       MemberQ[B2,
                         Sort[{B1[[1, 2]], k}]] &&
                       DivideQ[LT[Gb[[k]], vars, opts],
                         lcm, vars],
                       Return[Return[True]]
                     ], {k, 1, t}];
                 False
               ];
             B1 = Flatten[Table[{i, j}, {i, 1, t - 1},
                  {j, i + 1, t}], 1];
             While[Length[B1] > 0,
               AppendTo[B2, B1[[1]]];
               {f, g} = Gb[[B1[[1]]]];
               LMf = SplitTerm[LT[f, vars, opts],
                    vars][[2]];
               LMg = SplitTerm[LT[g, vars, opts],
                    vars][[2]];
               lcm = LCMMonomial[LMf, LMg, vars];
               If[lcm - LMf * LMg =!= 0 &&
                   Not[Criterion[]],
                 sred =
                   PolynomialReduce[
                       SPol[f, g, vars, opts], Gb, vars,
                       opts][[2]];
                 If[sred =!= 0,
                   t++;
                   AppendTo[Gb, sred];
                   B1 = Join[B1, Table[{i, t},
                        {i, 1, t - 1}]]
                 ]];
               B1 = Delete[B1, 1];
             ];
             ExpandPoly[Gb, vars]
           ];
```

Unsere bekannte Liste der kritischen Paare **B** heisst nun **B1**. Die zu Beginn leere Liste **B2** verwaltet alle aus **B1** eliminierten kritischen Paare. Mit ihr wird in **Criterion** überprüft, ob $S[i,k]$ bzw. $S[j,k]$ für die Gröbnerbasis bereits berücksichtigt wurden. Abgesehen von unserer neuen inneren Funktion ist **BuchbergerK2** weitgehend mit **BuchbergerK1** identisch.
Criterion liefert genau dann **True** zurück, wenn das zweite Buchbergerkriterium er-

füllt ist, das gerade betrachtete S-Polynom also nicht berücksichtigt zu werden braucht. Die hierzu aufgerufene Hilfsfunktion **DivideQ** ist im Anhang A.6 Seite 77 beschrieben.

Beispiel 2.9

Berechnen wir zum Vergleich noch einmal eine Gröbnerbasis unter lexikographischer Ordnung für das Erzeugendensystem $G = \{bz^4+z, \frac{b}{a}x^2+axy, ax^3y+xz+cz\} \subset \mathbb{Q}[x,y,z]$ aus Beispiel 2.6.

```
In[53]:= BuchbergerK2[G, {x, y, z},
           CoefficientDomain → RationalFunctions]//
         Timing
Out[53]= {1.09 Second,
          {z + b z^4, (b x^2)/a + a x y, a x^3 y + c z + x z,
           -(a^4 x y^3)/b^2 - (c z)/a - (x z)/a, -(b^2 c x z)/a^5 - (b c y z)/a^3,
           -(a^2 y^4 z)/b + (b^2 c z^2)/a^5 - (b y z^2)/a^3}}
```

Mit integriertem zweiten Buchbergerkriterium haben wir die Berechnungzeit für dieses Beispiel von 1.74 Sekunden (Beispiel 2.6) auf 1.09 Sekunden drücken können. Im nächsten Abschnitt werden wir weitere Berechnungen mit **BuchbergerK2** durchführen, vor allem auch an komplexeren Erzeugendensystemen.

2.4 Sugar, eine weitere Beschleunigungsstrategie

Auch in diesem Abschnitt beschäftigen wir uns weiterhin mit Beschleunigungsverfahren des Buchbergeralgorithmus. Dabei legen wir nun unser Hauptaugenmerk auf die Abarbeitungsreihenfolge der kritischen Paare. In den bisherigen Algorithmen haben wir diese Paare in der Reihenfolge abgearbeitet, in der sie (in der Liste **B** der kritischen Paare) auftraten. Nun versuchen wir durch geschicktes Umsortieren dieser Paare einen weiteren Effizienzgewinn zu erzielen. Wir verwenden dazu die in [GMN+91] beschriebene Sugar-Strategie. Diese Strategie ist eine Heuristik, die durch viele Tests bestätigt wurde. Sie wird durch die angenehmen Berechnungseigenschaften bei Erzeugendensystemen aus homogenen Polynomen motiviert. Diese Eigenschaften werden wir im Folgenden kurz zusammenfassen.

Definition 20 *Ein Polynom $f \in k[x_1,\ldots,x_n]$ heißt **homogen** vom totalen Grad d, falls alle Monome von f den totalen Grad d haben. Ein Ideal $I \subset k[x_1,\ldots,x_n]$ heißt **homogen**, falls es von homogenen Polynomen erzeugt wird.*

Jedes Polynom f kann als Summe homogener Polynome f_d aufgefasst werden, wobei d den totalen Grad des Polynoms darstellt. Beispielsweise haben wir

$$f = \underbrace{x^7 + 3x^2y^5}_{\deg=7} + \underbrace{5x^4 - x^2y^2 + 3y^4}_{\deg=4} + \underbrace{x^2 + 2xy}_{\deg=2} = f_7 + f_4 + f_2.$$

Die f_d's sind die *homogenen Komponenten* von f. Ist I ein homogenes Ideal, so liegt für $f \in I$ auch jede homogene Komponente f_d in I. Der Buchbergeralgorithmus ist bei der Berechnung von Gröbnerbasen homogener Ideale sehr effizient, was sich auf einige Eigenschaften dieser Ideale zurückführen lässt. Sind etwa bereits die Erzeugerpolynome homogen, so auch alle bei der Berechnung auftretenden Polynome. Zudem ist jede reduzierte Gröbnerbasis homogener Ideale homogen, unabhängig von der Gestalt der Erzeugerpolynome.
Diese angenehmen Eigenschaften könnten wir zur Effizienzsteigerung bei der Berechnung von Gröbnerbasen nutzen. Hierzu konkretisieren wir erst einmal die Sachlage.

Ist $f(x_1, \dots, x_n) \in k[x_1, \dots, x_n]$ vom totalen Grad d mit der Summendarstellung $f = \sum_{i=0}^{d} f_i$ seiner homogenen Komponenten, so heißt $f^h(x_0, \dots, x_n) \in k[x_0, \dots, x_n]$ mit der Darstellung

$$\begin{aligned} f^h(x_0, \dots, x_n) &= \sum_{i=0}^{d} f_i(x_1, \dots, x_n)\, x_0^{d-i} \\ &= f_d(x_1, \dots, x_n) + f_{d-1}(x_1, \dots, x_n)\, x_0 \\ &\quad + \cdots + f_0(x_1, \dots, x_n)\, x_0^d \end{aligned}$$

die *Homogenisierung von* f. Es gilt

$$f^h = x_0^d \cdot f\left(\frac{x_1}{x_0}, \dots, \frac{x_n}{x_0}\right).$$

Für $f(x_1, x_2) = x_1^4 x_2 + 2x_1 x_2^2 - x_1 + 1$ etwa ist $f^h(x_0, x_1, x_2) = x_1^4 x_2 + 2x_0^2 x_1 x_2^2 - x_0^4 x_1 + x_0^5$ die Homogenisierung. Zur *Dehomogenisierung* wird die Variable x_0 auf 1 gesetzt. Wir erhalten somit unser ursprüngliches Polynom wieder:

$$f(x_1, \dots, x_n) = f^h(1, x_1, \dots, x_n) = f.$$

Auch Ideale lassen sich homogenisieren.

$$I^h = \langle f^h \in k[x_0, \dots, x_n] \ : \ f \in I \rangle$$

heißt die *Homogenisierung des Ideals* I. Die *Dehomogenisierung* sieht wie folgt aus:

$$\mathrm{dehom}(I^h) = \langle g(1, x_1, \dots, x_n) \ : \ g \in I^h \rangle$$

Damit erhalten wir auch wieder unser ursprüngliches Ideal, d. h. es gilt

$$\mathrm{dehom}(I^h) = I.$$

Mit diesen Bezeichnungen läge folgendes Verfahren zur Berechnung einer Gröbnerbasis eines Ideals $I = \langle f_1, \dots, f_s \rangle \subset k[x_1, \dots, x_n]$ nahe:
Wir homogenisieren die Erzeugerpolynome $f_1, \dots, f_s$ zu $f_1^h, \dots, f_s^h$, für die wir effizient mit dem Buchbergeralgorithmus eine Gröbnerbasis berechnen können. Anschließend dehomogenisieren wir die Gröbnerbasis und sind fertig. Leider gilt jedoch im Allgemeinen

$$\langle f_1^h, \dots, f_s^h \rangle \neq I^h.$$

Wir würden also mit dem Buchbergeralgorithmus nicht wirklich eine Gröbnerbasis von I^h berechnen. Dieser Missestand wäre dann mit weiteren zeitbehafteten Korrekturen zu beseitigen, die den Zeitgewinn wieder relativieren.

Die *Sugar-Strategie* benutzt nun die Vorteile der Homogenisierung, hat aber nicht deren Nachteile. Sie ist eine durch Tests bestätigte heuristische Art den Buchbergeralgorithmus zu beschleunigen. Hierzu die grundlegende Definition.

Definition 21 (Sugar) *Für jedes im Buchbergeralgorithmus vorkommende Polynom f definieren wir den Sugar S_f von f durch die folgenden Eigenschaften:*

1. *Ist f ein Erzeugerpolynom, so sei $S_f = \deg(f)$ der totale Grad von f.*
2. *Ist f ein Polynom und $t = \mathbf{x}^\alpha$ ein Monom, sei $S_{tf} = \deg t + S_f$.*
3. *Ist $f = g + h$, sei $S_f = \max\{S_g, S_h\}$.*

Beispiel 2.10

Seien $f = xy^3 + 1$ und $g = x^2y + x$ aus $\mathbb{Q}[x, y]$. Dann ist $S_f = 4$ und $S_g = 3$. Für das S-Polynom von f und g ergibt sich:

$$\begin{aligned} S(f,g) &= \frac{\mathrm{kgV}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(f)} \cdot f - \frac{\mathrm{kgV}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(g)} \cdot g \\ &= \frac{x^2y^3}{xy^3} \cdot (xy^3 + 1) - \frac{x^2y^3}{x^2y} \cdot (x^2y + x) \\ &= x \cdot (xy^3 + 1) - y^2 \cdot (x^2y + x) \\ &= x - xy^2. \end{aligned}$$

Es hat also den totalen Grad 3. Hingegen erhalten wir für den Sugar des S-Polynoms $S_{fg} := S_{S(f,g)}$ folgenden Wert:

$$\begin{aligned} S_{fg} &= \max\{(\deg(x^2y^3) - \deg(xy^3)) + S_f, (\deg(x^2y^3) - \deg(x^2y)) + S_g\} \\ &= \max\{(5-4)+4, (5-3)+3\} \\ &= 5. \end{aligned}$$

Dieser Wert entspricht dem totalen Grad des S-Polynoms der homogenisierten Polynome f^h und g^h:

$$\begin{aligned} S(f^h, g^h) &= \frac{x^2y^3}{xy^3} \cdot (xy^3 + x_0^4) - \frac{x^2y^3}{x^2y} \cdot (x^2y + xx_0^2) \\ &= xx_0^4 - y^2xx_0^2, \end{aligned}$$

also $\deg(S(f^h, g^h)) = 5$.

Den Sugar können wir, wie dieses Beispiel bestätigt, als eine „Phantomhomogenisierung“ auffassen. Grob gesprochen besteht die Sugar-Strategie des Buchbergeralgorithmus darin, die Polynome mit kleinem Sugar bei der Berechnung zu bevorzugen.

Es gibt verschiedene Varianten der Implementierung. In [GMN+91] werden zwei vorgestellt, die *sloppy*-Variante und die *fuzzy*-Variante. Bei der sloppy-Variante haben die kleinsten gemeinsamen Vielfachen der Leitmonome vornehmlich Einfluss, wenn es darum geht überflüssige kritische Paare zu eliminieren. Dagegen spielt bei der fuzzy-Variante der Sugar eine vordergründige Rolle beim Eliminieren von kritischen Paaren. In den meisten Fällen zeigt sich kein merklicher Unterschied zwischen diesen Berechnungsvarianten (siehe [GMN+91], Seite 54).

Wir implementieren im Folgenden den Buchbergeralgorithmus mit Sugar-Strategie nach der sloppy-Variante und halten uns in etwa an die Vorgehensweise, wie sie in [GMN+91] beschrieben ist. Jedoch bezieht sich diese Beschreibung auf eine imperative Programmiersprache mit auf den Algorithmus zugeschnittenen Datenstrukturen.[1] Da wir die Sugar-Strategie in *Mathematica* implementieren wollen, passen wir diese Beschreibung unseren Gegebenheiten an und modifizieren sie an gegebener Stelle, wo dies durch Tests bestätigt sinnvoll erschien. Der Algorithmus wird etwas umfangreicher ausfallen, so dass wir ihn abschnittweise beschreiben werden. Im Anhang A.10 Seite 82 ist er nochmals kompakt aufgelistet.

```
In[54]:= BuchbergerSugar[G_, vars_,
           opts___?OptionQ] :=
         Module[{Gb, t = Length[G], B, BNew, τ,
            cp, sred},
           Gb = {G,
              Table[TotalDegree[G[[i]], vars],
                {i, 1, t}],
              Table[
                SplitTerm[LT[G[[i]], vars, opts],
                   vars][[2]], {i, 1, t}]
           };
           B = Flatten[Table[{
                 {i, j},
                 Max[Gb[[2, i]]-
                       TotalDegree[Gb[[3, i]], vars],
                    Gb[[2, j]] - TotalDegree[
                       Gb[[3, j]], vars]]+
                  TotalDegree[
                    τ = LCMMonomial[Gb[[3, i]],
                       Gb[[3, j]], vars], vars],
                 τ
               }, {j, 1, t}, {i, 1, j - 1}], 1];
```

[1] So werden etwa die Polynome als, der zugrundeliegenden Monomordnung nach, sortierte Liste von (Koeffizient, Monom)-Paaren verwaltet. Dies hat den Vorteil, dass sich Polynome leicht addieren und mit Monomen multiplizieren lassen. Auch ist ein Algorithmus zur Leittermbestimmung überflüssig. Hingegen überlassen wir bei unserer Implementierung *Mathematica* die Verwaltung der Polynome und nehmen damit in Kauf Berechnungen mehrfach oder aus imperativer Sicht wesentlich umständlicher durchzuführen. Wir erheben aber nicht den Anspruch unseren in *Mathematica* implementierten Algorithmus mit einem imperativ programmierten zu vergleichen. Durch die überschaubare Größe des Programmcodes bei *Mathematica*-Programmen wollen wir hingegen die Verfahren des Algorithmus hervorheben.

Dem Algorithmus geben wir den Namen **BuchbergerSugar**. Die Parameter sind mit denen der vorher beschriebenen Buchbergeralgorithmen identisch. In der Liste **Gb** wurden bisher die Erzeuger des Ideals gespeichert, die wir zu einer Gröbnerbasis erweitert haben. Zuzüglich legen wir jetzt in zwei weiteren Listen jeweils den entsprechenden Sugar $s_i := S_{f_i}$ bzw. das Leitmonom $\tau_i := \mathrm{LM}(f_i)$ ab, weil wir diese Größen im Laufe des Algorithmus immer wieder benötigen. Wir können demnach **Gb** als eine $3 \times t$-Matrix mit den Spaltenvektoren (f_i, s_i, τ_i) auffassen. Dabei ist **t** die aktuelle Anzahl der Polynome in **Gb**. Die Tripel (f_i, s_i, τ_i) speichern wir nicht als Zeilenvektoren ab, damit wir leichter auf alle Polynome zugreifen können, ohne die Matrix **Gb** vorher zu transponieren.

Gb[[1]] und nicht **Transpose[Gb][[1]]**

Der Sugar der Erzeugerpolynome ist nach Definition der totale Grad der jeweiligen Polynome. So ist bei der Initialisierung von **Gb**

$$s_i = \deg(f_i).$$

Die für die Berechnung aufgerufene Funktion **TotalDegree** ist im Anhang A.7 auf Seite 78 näher erläutert. Sie berechnet den totalen Grad eines Polynoms.
Auch die Liste **B** der kritischen Paare erweitern wir. Zuzüglich zu den kritischen Paaren (i,j) mit $1 \le i < j \le t$ speichern wir den Sugar $s_{ij} := S_{S(f_i,f_j)}$ und das Leitmonom τ_{ij} der Paare ab. Im Gegensatz zur Polynomliste **Gb** werden die Tripel $((i,j), s_{ij}, \tau_{ij})$ als Zeilenvektoren abgelegt, damit wir über den Ausdruck **B[[k]]** auf ein bestimmtes Tripel zugreifen können. Unter dem Leitmonom τ_{ij} des Paares (i,j) verstehen wir das Leitmonom des kleinsten gemeinsamen Vielfaches der Polynome f_i und f_j aus **Gb**, also $\tau_{ij} = \mathrm{LM}(\mathrm{kgV}(f_i, f_j))$. Der Sugar s_{ij} lässt sich nach obiger Definition mit der Formel

$$s_{ij} = \max\{s_i - \deg(\tau_i), s_j - \deg(\tau_j)\} + \deg(\tau_{ij})$$

errechnen.

```
B = Delete[B,
    Position[
      Table[B[[k, 3]]-
          Gb[[3, B[[k, 1, 1]]]]*
            Gb[[3, B[[k, 1, 2]]]],
        {k, 1, Length[B]}], 0]];
B = SortCriticalPairs[B, vars, opts];
```

Nach der Initialisierung der Listen **Gb** und **B** wenden wir das erste Buchbergerkriterium auf die Liste der kritischen Paare **B** an, d. h. wir löschen alle Tripel $((i,j), s_{ij}, \tau_{ij})$, für die

$$\tau_{ij} - \tau_i \cdot \tau_j = 0$$

gilt, also für die die Leitmonome der Polynome f_i und f_j aus **Gb** relativ prim sind.
Durch den Aufruf der Funktion **SortCriticalPairs**, die im Anhang A.8 auf Seite 79 beschrieben ist, sortieren wir nun die kritischen Paare nach folgender Sortierreihenfolge:

1. nach dem Sugar, d. h. das Paar mit kleinstem Sugar zuerst.

2. falls nötig, differenzieren wir zusätzlich nach *normaler Auswahlstrategie*, d. h.

   (a) Leitmonome τ_{ij} nach Monomordnung sortieren,

   (b) falls nötig, bevorzugen wir das kleinere j.

```
While[Length[B] > 0,
  cp = B[[1]];
  B = Delete[B, 1];
  sred =
    PolynomialReduce[
        SPol[Gb[[1, cp[[1, 1]]]],
          Gb[[1, cp[[1, 2]]]], vars, opts],
        Gb[[1]], vars, opts][[2]];
```

Nun arbeiten wir die Liste **B** der kritischen Paare ab. Wir wählen aus **B** das Tripel $((i,j), s_{ij}, \tau_{ij})$ des kritischen Paares (i,j) mit kleinstem Sugar, speichern es in der Variablen **cp** (critical pair) ab und löschen es aus **B**. Dann speichern wir in **sred** den Rest des S-Polynoms $S(f_i, f_j)$ modulo aller bis dato in **Gb** vorhandenen Polynome.

```
If[sred =!= 0,
  t++;
  Gb = MapThread[Append,
      {Gb, {sred, cp[[2]],
         SplitTerm[LT[sred, vars, opts],
            vars][[2]]}}];
```

Ist der in **sred** abgespeicherte Rest Null, wenden wir uns dem nächsten kritischen Paar zu. Falls er von Null verschieden ist, wird er der Polynomliste **Gb** angehängt, zuzüglich des zugehörigen Sugars s_{ij} und Leitmonoms. Auch wird die Variable **t**, die die Anzahl der Polynome in **Gb** enthält, inkrementiert.

```
BNew = Table[{
      {k, t},
      Max[Gb[[2, k]] -
            TotalDegree[Gb[[3, k]], vars],
          Gb[[2, t]] - TotalDegree[
              Gb[[3, t]], vars]] +
        TotalDegree[
          τ = LCMMonomial[Gb[[3, k]],
              Gb[[3, t]], vars], vars],
      τ
    }, {k, 1, t - 1}];
```

Durch das neu hinzugewonnene Polynom haben wir auch weitere kritische Paare zu berücksichtigen. Diese speichern wir mit Sugar und Leitmonom in **BNew** ab.
Mit der folgenden Anweisung eliminieren wir kritische Paare aus **B**. Diesem Vorgehen liegt Buchbergers zweites Kriterium zu Grunde. Da wir nach der sloppy-Variante vorgehen, beziehen wir uns auf die Leitmonome der kleinsten gemeinsamen Vielfachen. Ist τ_t das Leitmonom des neuen Polynoms, so eliminieren wir die kritischen Paare (i, j)

aus **B**, für die τ_{ij} ein echtes Vielfaches von τ_{it} bzw. τ_{jt} ist.

```
B = Delete[B,
    Position[
      Table[
        StrictDivideQ[
            BNew[[B[[k, 1, 1]], 3]],
            B[[k, 3]], vars] &&
          StrictDivideQ[
            BNew[[B[[k, 1, 2]], 3]],
            B[[k, 3]], vars],
        {k, 1, Length[B]}], True]];
```

Nun wenden wir das erste Buchbergerkriterium auf **BNew**, die neuen kritischen Paare, an.

```
BNew = Delete[BNew,
    Position[
      Table[BNew[[k, 3]]-
          Gb[[3, k]] * Gb[[3, t]],
        {k, 1, t - 1}], 0]];
```

Dann werden die neuen kritischen Paare untereinander nach dem oben beschriebenen Verfahren sortiert und in die Liste **B** der alten kritischen Paare eingemischt. Die hierzu verwendete **Merge**-Funktion ist in Anhang A.9 auf Seite 81 beschrieben.

```
      BNew = SortCriticalPairs[BNew, vars,
          opts];
      B = Merge[B, BNew, vars, opts];
    ];
  ];
  ExpandPoly[Gb[[1]], vars]
]
```

Erst wenn die Liste **B** kein Element mehr enthält, d. h. wenn alle kritischen Paare berücksichtigt wurden, verlassen wir die **While**-Schleife. Dann bilden die Polynome in **Gb** eine Gröbnerbasis, die in ausmultiplizierter Form zurückgegeben wird.

Beispiel 2.11

Jetzt ist es an der Zeit zu überprüfen, ob der Buchbergeralgorithmus mit Sugar-Strategie wirklich schneller ist.

1. Nehmen wir uns das Erzeugendensystem $G \subset \mathbb{Q}[x, y, z, w]$ mit den Erzeugern

$$\begin{array}{lll} x^2w + y^2z, & -xzw + y^2z, & x^3w^3 + yz^3w^3, \\ x^2w + xzw, & -xyz^2w^3 - xz^3w^4, & xy^2z + y^2z^2, \\ -y^4z - y^2z^2w, & -y^6z^2 + y^3z^4w^2, & -yz^8w^4 - yz^7w^3 \end{array}$$

aus Beispiel 2.4.4 nochmals vor und schauen zunächst, wie lange der Buchbergeralgorithmus mit den beiden Buchbergerkriterien für die Berechnung einer Gröbnerbasis benötigt.

```
In[55]:= BuchbergerK2[G, {x, y, z, w},
           MonomialOrder → Lexicographic]; //
         Timing
Out[55]= {9.77 Second, Null}

In[56]:= BuchbergerK2[G, {x, y, z, w},
           MonomialOrder → DegreeLexicographic]; //
         Timing
Out[56]= {3.91 Second, Null}
```

Im Vergleich zum Buchbergeralgorithmus **Buchberger** ist **BuchbergerK2** unter *lex*-Ordnung etwa 11 Sekunden und unter *grlex*-Ordnung etwa 2 Sekunden schneller. Kann nun die Sugar-Strategie noch mehr Zeit herausholen?

```
In[57]:= BuchbergerSugar[G, {x, y, z, w},
           MonomialOrder → Lexicographic]; //
         Timing
Out[57]= {5.43 Second, Null}

In[58]:= BuchbergerSugar[G, {x, y, z, w},
           MonomialOrder → DegreeLexicographic]; //
         Timing
Out[58]= {4.72 Second, Null}
```

Wie wir sehen, konnten wir die Berechnungsdauer unter *lex*-Ordung auf fast die Hälfte reduzieren, während die Sugar-Strategie in diesem Beispiel unter *grlex*-Ordnung schlechter abschneidet. Der Zeitaufwand des Beispiels spielt sich aber noch im Sekundenbereich ab, so dass dieses Manko nicht so tragisch ist.

Schauen wir uns nun ein paar Erzeugendensysteme[2] an, bei denen es keine richtige Freude mehr macht vor dem Computer zu verharren, bis dieser eine Gröbnerbasis berechnet hat.

2. Dieses Beispiel wurde in Verbindung mit Integer-Programmierung studiert. Das Erzeugendensystem $G \subset \mathbb{Q}[x, y, z, t, u, v, w]$ bestehe aus folgenden Polynomen

$$\begin{aligned} &x^2yz^4 - t, \\ &x^5y^7 - z^2u, \\ &-x^3zv + y^2, \\ &-z^5w + xy^3. \end{aligned}$$

Für die Berechnungsdauer einer Gröbnerbasis unter *grlex*-Ordnung mit dem **BuchbergerK2**-Algorithmus ergibt sich folgender Wert:

[2] aus [GMN+91], Seite 52

```
In[59]:= BuchbergerK2[G, {x, y, z, t, u, v, w},
             MonomialOrder →
               DegreeReverseLexicographic]//Timing
Out[59]= {171.07 Second, Null}
```

Folglich warten wir ca. 3 Minuten auf eine Gröbnerbasis. Diese besteht aus 36 Polynomen, die wir hier nicht abdrucken wollen.

```
In[60]:= BuchbergerSugar[G, {x, y, z, t, u, v, w},
             MonomialOrder →
               DegreeReverseLexicographic]//Timing
Out[60]= {29.16 Second, Null}
```

Mit einer Berechnungsdauer von etwa einer halben Minute lohnt sich die Benutzung der Sugar-Strategie, zudem besteht diese Gröbnerbasis nur aus 21 Polynomen.

3. Das folgende Erzeugendensystem $G \subset \mathbb{Q}[x,y,z,t]$ stellt eine Parameterkurve dar und ist charakteristisch für eine Reihe von Beispielen des selben Typs.

$$\begin{aligned} &x^{31} - x^6 - x - y, \\ &x^8 - z, \\ &x^{10} - t. \end{aligned}$$

Vergleichen wir wieder die Algorithmen, indem wir einmal Gröbnerbasen bezüglich *lex*- und darauf bezüglich *grevlex*-Ordnung berechnen.

```
In[61]:= BuchbergerK2[G, {x, y, z, t}]//Timing
Out[61]= {1817.61 Second, Null}

In[62]:= BuchbergerSugar[G, {x, y, z, t}]//Timing
Out[62]= {259.19 Second, Null}


In[63]:= BuchbergerK2[G, {x, y, z, t},
             MonomialOrder →
               DegreeReverseLexicographic]//Timing
Out[63]= {241.03 Second, Null}

In[64]:= BuchbergerSugar[G, {x, y, z, t},
             MonomialOrder →
               DegreeReverseLexicographic]//Timing
Out[64]= {48.96 Second, Null}
```

Bei beiden Monomordnungen ist der Berechnungsvorteil, den wir mit der Sugar-Strategie haben, nicht zu übersehen.

Kapitel 3

Eine Anwendung von Gröbnerbasen

Im letzten Kapitel dieser Arbeit schauen wir uns eine mögliche Anwendung von Gröbnerbasen in der Geometrie an, nämlich wie sich mit Hilfe von Gröbnerbasen „neue“ geometrische Eigenschaften finden lassen. Die betrachteten Objekte werden dabei in erster Linie Dreiecke und Ellipsen in der euklidischen Ebene sein. Zuvor gehen wir jedoch noch näher auf das Lösen polynomieller Gleichungssysteme ein, das sich im Weiteren als wichtiges Werkzeug erweisen wird.

3.1 Lösen polynomieller Gleichungssysteme

In diesem Abschnitt beschäftigen wir uns mit den grundlegenden Verfahren zum Lösen polynomieller Gleichungssysteme in $\mathbb{C}$. Seien $f_1, \dots, f_s \in \mathbb{C}[x_1, \dots, x_n]$. Wir suchen die Punkte $(a_1, \dots, a_n) \in \mathbb{C}^n$ für die $f_i(a_1, \dots, a_n) = 0$ gilt, also Lösungen des polynomiellen Gleichungssystems

$$\begin{aligned} f_1(x_1, \dots, x_n) &= 0, \\ &\vdots \\ f_s(x_1, \dots, x_n) &= 0 \end{aligned} \tag{3.1}$$

sind. Solch ein Punkt $(a_1, \dots, a_n)$ ist auch Lösung von allen Polynomen aus dem Ideal $I = \langle f_1, \dots, f_s \rangle$ und die Menge aller Lösungen bildet die affine Varietät

$$\mathbf{V}(I) = \{(a_1, \dots, a_n) \in \mathbb{C}^n : f(a_1, \dots, a_n) = 0 \text{ für alle } f \in I\}$$

wie bereits in Kapitel 1 auf Seite 2 bzw. Seite 11 beschrieben. Ziel ist es nun die affine Varietät $\mathbf{V}(I)$, die Menge aller Lösungen des polynomiellen Gleichungssystems, zu bestimmen. Dabei stellen Gröbnerbasen ein hilfreiches Werkzeug dar. Der Lösungsprozess besteht im Wesentlichen aus zwei Schritten.

1. dem *Eliminationsschritt*:
   Mit Hilfe von Gröbnerbasen wird eine einfacher zu lösende Darstellung des Polynomsystems bestimmt.

2. dem *Erweiterungsschritt*:
   Berechnete Teillösungen von leichter zu lösenden Polynomgleichungen in weniger Variablen werden zu Lösungen des originalen Gleichungssystems erweitert.

Schauen wir uns zunächst den Eliminationsschritt an. Wie beim Lösen linearer Gleichungssysteme, bei dem das System in eine Dreiecksform mit fortlaufend weniger Variablen gebracht wird, versuchen wir auch bei polynomiellen Gleichungssystemen sukzessive Variablen zu eliminieren.
Hierzu definieren wir für $I = \langle f_1, \dots, f_s \rangle \subset k[x_1, \dots, x_n]$ das l-te *Eliminationsideal*

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

Es ist leicht zu sehen, dass I_l ein Ideal von $k[x_{l+1}, \dots, x_n]$ ist. Es besteht aus Polynomen von I, die nicht die Variablen $x_1, \dots, x_l$ enthalten. Mit einem systematischen Verfahren, welches Erzeuger von I_l bestimmt, können wir dann den Eliminationsschritt durchführen. Dieses Verfahren besteht aus der Berechnung einer Gröbnerbasis bezüglich einer geeigneten Ordnung, wie folgender Satz zeigt.

Satz 22 (Eliminationssatz) *Sei $I \subset k[x_1, \dots, x_n]$ ein Ideal und G eine Gröbnerbasis von I unter lex-Ordnung. Dann ist für $0 \leq l \leq n-1$ die Menge*

$$G_l := G \cap k[x_{l+1}, \dots, x_n]$$

eine Gröbnerbasis des l-ten Eliminationsideals I_l.

Beweis
Für $l \in \{0, \dots, n\}$ zeigen wir $\langle \mathrm{LT}(I_l) \rangle \subset \langle \mathrm{LT}(G_l) \rangle$, d. h. wir haben zu $f \in I_l$ ein $g \in G_l$ zu finden, dessen Leitterm $\mathrm{LT}(g)$ den Leitterm $\mathrm{LT}(f)$ teilt.
Da f auch ein Element von I ist, existiert ein $g \in G$ mit $\mathrm{LT}(g) \mid \mathrm{LT}(f)$. Wegen $f \in I_l$ enthält $\mathrm{LT}(g)$ höchstens die Variablen $x_{l+1}, \dots, x_n$ und da wir *lex*-Ordnung benutzen, sind alle Terme, die mindestens eine der Variablen $x_1, \dots, x_l$ enthalten, größer als $\mathrm{LT}(g)$. Das Polynom g besteht demnach nur aus Termen, die frei von den Variablen $x_1, \dots, x_l$ sind. $\mathrm{LT}(g) \in k[x_{l+1}, \dots, x_n]$ impliziert also unter lexikographischer Ordnung, dass auch g aus $k[x_{l+1}, \dots, x_n]$ ist. Somit enthält auch G_l das Polynom g. □

Beispiel 3.1

Wir betrachten das Gleichungssystem

$$\begin{aligned} x^2 + y^2 + z^2 - 3 &= 0, \\ xyz - 1 &= 0, \\ xz + y - 2 &= 0. \end{aligned}$$

Um den Eliminationsschritt durchzuführen, berechnen wir die reduzierte Gröbnerbasis bezüglich *lex*-Ordnung des Ideals

$$I = \langle x^2 + y^2 + z^2 - 3, xyz - 1, xz + y - 2 \rangle \subset \mathbb{C}[x, y, z].$$

```
In[65]:= G = {x^2 + y^2 + z^2 - 3, x y z - 1, x z + y - 2};
         BuchbergerRed[G, {x, y, z}]
Out[65]= {x - 7 z/2 + y z + 2 z^3 - z^5/2, 1 - 2 y + y^2,
           3/2 - y - 2 z^2 + y z^2 + z^4/2, -1 + 3 z^2 - 3 z^4 + z^6}
```

Die Eliminationsideale I_1 und I_2 lassen sich somit durch folgende Gröbnerbasen erzeugen:

$$\begin{aligned} I_1 &= I \cap \mathbb{C}[y,z] = \langle z^6 - 3z^4 + 3z^2 - 1, 2yz^2 - 2y + z^4 - 4z^2 + 3, y^2 - 2y + 1\rangle, \\ I_2 &= I \cap \mathbb{C}[z] = \langle z^6 - 3z^4 + 3z^2 - 1\rangle. \end{aligned}$$

Nun diskutieren wir den Erweiterungsschritt. Die grundlegende Idee, die hinter diesem Verfahren steht, lässt sich wieder mit dem Vorgehen bei linearen Gleichungssystemen vergleichen. Dort wird beim in Dreiecksform gebrachten Gleichungssystem die untere Gleichung gelöst, die als Teillösung in die darüberstehende Gleichung rücksubstituiert wird, um diese lösen zu können. Dieses Verfahren wird fortlaufend nach oben fortgesetzt, bis die oberste Gleichung erreicht ist und die Teillösungen zu einer Lösung des ganzen Gleichungssystems erweitert wurden.
Um nun die Punkte der zum polynomiellen Gleichungssystem (3.1) korrespondierenden affinen Varietät $\mathbf{V}(I)$ zu bestimmen, beginnen wir entsprechend beim Eliminationsideal I_{n-1} und bestimmen für dieses alle Teillösungen $a_n \in \mathbf{V}(I_{n-1}) \subset \mathbb{C}$. Dann erweitern wir diese Teillösungen durch eine weitere Koordinate, d. h. wir suchen alle a_{n-1}, für die (a_{n-1}, a_n) in der Varietät $\mathbf{V}(I_{n-2})$ liegt. Diesen Erweiterungsvorgang führen wir so lange fort, bis wir die Teillösungen $(a_2, \ldots, a_n)$ zu Lösungen $(a_1, \ldots, a_n) \in \mathbf{V}(I)$ erweitert haben.
Allgemeiner formuliert suchen wir bei jedem Erweiterungsschritt zu gegebenem Eliminationsideal $I_{l-1} = \langle g_1, \ldots, g_r\rangle \subset \mathbb{C}[x_l, \ldots, x_n]$ und schon berechneten Teillösungen $(a_{l+1}, \ldots, a_n) \in \mathbf{V}(I_l)$ die Lösungen $x_l = a_l$ der Gleichungen

$$g_1(x_l, a_{l+1}, \ldots, a_n) = \cdots = g_r(x_l, a_{l+1}, \ldots, a_n) = 0.$$

Diese Gleichungen setzen sich aus Polynomen in der einen Variablen x_l zusammen. Folglich bestehen die Lösungen aus den Nullstellen des größten gemeinsamen Teilers der r Polynome.
Ganz so reibungslos wie bei linearen Gleichungssystemen vollzieht sich aber diese Erweiterung nicht, wie folgendes Beispiel illustriert:

Beispiel 3.2

Wir betrachten das Gleichungssystem

$$\begin{aligned} xy &= 1, \\ xz &= 1. \end{aligned}$$

Mit dem dazu korrespondierenden Ideal $I = \langle xy-1, xz-1\rangle$ erhalten wir durch Anwendung des Eliminationssatzes folgende Gröbnerbasen der Eliminationsideale:

$$\begin{aligned} I_1 &= I \cap \mathbb{C}[y,z] = \langle y-z\rangle, \\ I_2 &= I \cap \mathbb{C}[z] = \{0\}. \end{aligned}$$

Das zweite Eliminationsideal I_2 lässt alle $a \in \mathbb{C}$ als Teillösungen zu, die sich mit I_1 zu Teillösungen $(a,a) \in \mathbb{C}^2$ erweitern lassen. Die nächste Erweiterung zu den vollständigen Lösungen $(\frac{1}{a}, a, a) \in \mathbb{C}^3$ zeigt jedoch, dass nicht alle Teillösungen (a,a) erweitert werden können, denn für $a = 0$ existiert keine Lösung.

Es stellt sich nun die Frage, unter welchen Umständen eine Teillösung erweitert werden kann. Eine hinreichende Antwort gibt uns der Erweiterungssatz. Wir formulieren ihn für den Fall, in dem die erste Variable x_1 eliminiert ist, d. h. wir fragen uns, wann eine uns bekannte Teillösung $(a_2, \dots, a_n) \in \mathbf{V}(I_1)$ zu einer Lösung $(a_1, \dots, a_n) \in \mathbf{V}(I)$ erweitert werden kann. Dieser Spezialfall lässt sich ohne Weiteres auf eine Teillösung $(a_l, \dots, a_n) \in \mathbf{V}(I_{l-1})$ verallgemeinern.

Satz 23 (Erweiterungssatz) *Sei $I = \langle f_1, \dots, f_s\rangle \subset \mathbb{C}[x_1 \dots, x_n]$ und sei I_1 das erste Eliminationsideal von I. Für $1 \le i \le s$ schreiben wir f_i in der Form*

$$f_i = g_i(x_2, \dots, x_n)\, x_1^{N_i} + \textit{Terme in } x_i \textit{ mit kleinerem Grad als } N_i,$$

mit $N_i \ge 0$ und von Null verschiedenen $g_i \in \mathbb{C}[x_1, \dots, x_n]$. Ist $(a_2, \dots, a_n) \in \mathbf{V}(I_1)$ eine Teillösung, dann existiert ein $a_1 \in \mathbb{C}$, so dass $(a_1, a_2, \dots, a_n) \in \mathbf{V}(I)$, falls

$$(a_2, \dots, a_n) \notin \mathbf{V}(g_1, \dots, g_s).$$

Im Beweis des Satzes werden Resultanten benutzt, die wir hier nicht behandeln werden. Wir verweisen hierzu auf [CLO98], Kapitel 3 §6 und insbesondere auf die Seiten 161-163.

Im Erweiterungssatz können wir die g_i's als Leitkoeffizienten der f_i's bezüglich x_1 auffassen. Folglich bedeutet die Bedingung $(a_2, \dots, a_n) \in \mathbf{V}(I_1)$, dass nicht alle Leitkoeffizienten gleichzeitig an der Teillösung verschwinden dürfen.
Der Erweiterungssatz sagt uns demnach, dass der Erweiterungsschritt nur dann scheitern kann, wenn alle Leitkoeffizienten verschieden sind.
Bei obigem Beispiel 3.2 lassen sich nach diesem Satz folglich alle Teillösungen $(y,z) = (a,a)$ erweitern, für die $a \neq 0$ gilt, da nur $(0,0)$ an den Leitkoeffizienten $g_1 = y$ und $g_2 = z$ verschwindet. Dass sich $(0,0)$ tatsächlich nicht erweitern lässt, haben wir schon im Beispiel gesehen.

Beispiel 3.3

Lösen wir nun das in Beispiel 3.1 dargestellte polynomielle Gleichungssystem. In diesem Beispiel haben wir schon Gröbnerbasen für die Eliminationsideale bestimmt:

$$\begin{aligned} I_1 &= I \cap \mathbb{C}[y,z] = \langle 2yz^2 - 2y + z^4 - 4z^2 + 3, y^2 - 2y + 1, z^6 - 3z^4 + 3z^2 - 1\rangle, \\ I_2 &= I \cap \mathbb{C}[z] = \langle z^6 - 3z^4 + 3z^2 - 1\rangle. \end{aligned}$$

Wir beginnen mit dem Erzeuger von I_2 und faktorisieren ihn.

```
In[66]:= Factor[z^6 - 3 z^4 + 3 z^2 - 1]
Out[66]= (-1 + z)^3 (1 + z)^3
```

Folglich besteht $\mathbf{V}(I_2)$ aus der Menge $\{-1, 1\}$. Die Vielfachheit dieser Teillösungen lassen wir hier unbeachtet. Um zu überprüfen, welche dieser Teillösungen wir auf alle Fälle erweitern können, wenden wir den Erweiterungssatz an. Zu I_2 haben wir bezüglich y die Leitkoeffizienten $g_1 = 2z^2 - 2, g_2 = 1$ und $g_3 = z^6 - 3z^4 + 3z^2 - 1$. Den Leitkoeffizienten g_3 können wir ignorieren, da er uns keine Information vermittelt. Dagegen sichert uns der konstante Leitkoeffizient g_2 zu, dass $V(g_1, g_2) = \varnothing$ gilt, also jede Teillösung erweitert werden kann. Faktorisieren wir nun die Erzeuger von I_1, die für uns von Bedeutung sind, um die Teillösungen zu erweitern.

```
In[67]:= Factor[2 y z^2 - 2 y + z^4 - 4 z^2 + 3]
         Factor[ - y^2 + 2 y - 1]
Out[67]= (-1 + z) (1 + z) ( - 3 + 2 y + z^2)
Out[67]= - (-1 + y)^2
```

Wegen der Linearfaktoren $(z + 1)$ und $(z - 1)$ im ersten Polynom erhalten wir für die Teillösungen $z = -1$ und $z = 1$ keine weitere Einschränkung. Erst das zweite Polynom schränkt für beide Teillösungen die Variable y auf den Wert 1 ein. Somit erhalten wir die erweiterten Teillösungen $(1, -1)$ und $(1, 1)$ für (y, z).
In der berechneten Gröbnerbasis ist $2x + 2yz - z^5 + 4z^3 - 7z$ das einzige Polynom, das die Variable x enthält. Da wir einen konstanten Koeffizienten bezüglich x vorfinden, sichert uns der Erweiterungssatz abermals die Erweiterung sämtlicher Teillösungen zu. Diese lassen sich, wie leicht nachzurechnen ist, zu den Lösungen $(-1, 1, -1)$ und $(1, 1, 1)$ erweitern.

3.2 Die Schwerpunktellipse eines Dreiecks

Im letzten Abschnitt dieser Arbeit schauen wir uns an einem Beispiel an, wie wir mit Hilfe von Gröbnerbasen geometrische Eigenschaften herleiten können. In diesem Beispiel betrachten wir ein ebenes Dreieck $\triangle ABC$.
Bei ebenen Dreiecken schneiden sich die Mittelsenkrechten in einem Punkt, dem Mittelpunkt des Umkreises. Auch die Winkelhalbierenden besitzen einen gemeinsamen Schnittpunkt, den Inkreismittelpunkt. Weitere charakteristische Strecken im Dreieck sind die Seitenhalbierenden. Auch diese schneiden sich in einem Punkt, dem Schwerpunkt des Dreiecks. Dieser Schwerpunkt ist jedoch kein Mittelpunkt eines für das Dreieck charakteristischen Kreises. Wie wir aber in diesem Abschnitt sehen werden, lässt sich zu jedem Dreieck eine Ellipse finden, deren Mittelpunkt der Schwerpunkt des Dreiecks ist und die durch dessen Eckpunkte verläuft - sozusagen eine „Umellipse“. Diese nennen wir *Schwerpunktellipse* des Dreiecks, um zu betonen, dass ihr Mittelpunkt der Dreiecksschwerpunkt ist.
Eines unserer Ziele in diesem Abschnitt ist die Aufstellung gewisser Formeln, mit denen wir zu gegebenen Dreiecksseitenlängen die charakteristischen Größen der zugehörigen Schwerpunktellipse bestimmen können.

Dazu drehen wir zunächst die Aufgabenstellung um und bestimmen zu gegebener Ellipse ein Dreieck, zu dem die Ellipse eine Schwerpunktellipse ist. Dieses Dreieck nennen wir *Ellipsendreieck*. So können wir für die Berechnungen eine für uns geeignet transformierte Ellipsengleichung verwenden. Doch erst einmal wollen wir zeigen, dass das zu bestimmende Dreieck eindeutig ist. Für den Beweis benötigen wir noch folgende Definition:

Definition 24 (senkrecht-achsenaffine Abbildung) *Eine senkrecht-achsenaffine Abbildung $A(f,s) : \mathbb{R}^2 \to \mathbb{R}^2$ ist eine Abbildung der Ebene in sich mit folgenden Eigenschaften:*

1. *Es existiert eine Fixpunktgerade f.*
2. *Für jeden Punkt $P \notin f$ und sein Bild $P'(\neq P)$ gilt:*
   (a) *$g(P;P') \perp f$ (Schnittpunkt sei (P)),*
   (b) *$|P'(P)| = |s| \cdot |P(P)|$ mit konstantem $s \in \mathbb{R}\backslash\{0\}$ (**Affinfaktor**),*
   (c) *P und P' liegen auf derselben Seite von f, falls $s > 0$, ansonsten auf verschiedenen.*

Sind f, g zwei Geraden in der Ebene, die sich im Punkt P schneiden und ist $g' = A(f,s)(g)$, so ist leicht einzusehen, dass g' die Geraden f und g auch in P schneidet (siehe etwa die Strecken $\overline{A'C'}$ und $\overline{AC}$ in Abbildung 3.1). Nun zu dem Eindeutigkeitssatz:

Satz 25 *Sei E die Menge aller Punkte in der euklidischen Ebene, die eine Ellipse beschreiben, und sei $C \in E$. Dann existiert genau ein Ellipsendreieck $\triangle ABC$ zu E.*

Beweis
Sei E eine Ellipse mit den Halbachsen g bzw. h und Mittelpunkt M. Sei G einer der Hauptachsenscheitelpunkte von E an der Halbachse g, f die Gerade durch die Punkte M und G und C ein beliebiger Punkt auf E (siehe Abbildung 3.1). Durch die senkrecht-achsenaffine Abbildung $A(f, \frac{g}{h})$ lässt sich E bijektiv auf einen Kreis K mit Radius g abbilden. Sei $C' := A(f, \frac{g}{h})(C)$. Da M auf f liegt, gilt $A(f, \frac{g}{h})(M) = M$. Für den Kreis lässt sich das Problem wie folgt lösen:
Ein Dreieck $\triangle A'B'C'$, dessen Ecken auf K liegen, hat K als Umkreis. Soll der Schwerpunkt des Dreiecks mit dem Umkreismittelpunkt, also mit M, übereinstimmen, ist das Dreieck gleichseitig, denn nur beim gleichseitigen Dreieck sind Schwerpunkt und Umkreismittelpunkt identisch[1]. Folglich gibt

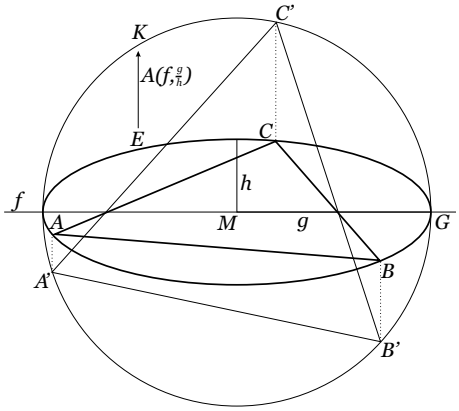


Abbildung 3.1:

[1] Sind in einem Dreieck Schwerpukt und Umkreismittelpunkt identisch, so liegen die Seitenhalbierenden auf den Mittelsenkrechten. Dies ist nur bei einem gleichseitigen Dreieck der Fall.

es zu dem Kreis K und dem Punkt C' auf K genau ein Dreieck $\triangle A'B'C'$ mit $A', B' \in K$, dessen Schwerpunkt mit dem Kreismittelpunkt übereinstimmt.
Die Punkte $A := A(f, \frac{g}{h})^{-1}(A')$ und $B := A(f, \frac{g}{h})^{-1}(B')$ liegen auf der Ellipse E. Die Seitenhalbierenden des Dreiecks $\triangle A'B'C'$ werden durch $A(f, \frac{g}{h})^{-1}$ auf Seitenhalbierenden des Dreiecks $\triangle ABC$ abgebildet. Der Schwerpunkt von $\triangle ABC$ liegt damit auch auf f und stimmt, da f die Fixpunktgerade ist, mit M überein. □

Dieser Beweis ist konstruktiv. Die Konstruktion lässt sich zudem mit Zirkel und Lineal durchführen, was bedeutet, dass die aus den geometrischen Eigenschaften resultierenden Polynome höchstens quadratisch sind.

Eine Ellipse E lässt sich eindeutig über die Längen ihrer beiden Halbachsen $g, h \in \mathbb{R}^+$ beschreiben. Dabei wollen wir der Einfachheit halber nicht zwischen einer Strecke und ihrer Länge unterscheiden. Die Lage des Dreiecks $\triangle ABC$ in E soll durch den Abstand l des Dreieckpunktes C von der Halbachse h bestimmt sein (siehe Abbildung 3.2). Wir bestimmen nun Formeln, mit denen sich zu diesen Größen g, h und l die Seitenlängen $a, b, c \in \mathbb{R}^+$ des Ellipsendreiecks berechnen lassen.

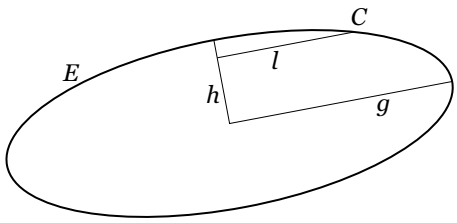


Abbildung 3.2:

Den Großteil der Arbeit werden wir dabei mit Hilfe von Gröbnerbasen verrichten. Wir gehen wie folgt vor:
Die Ellipse E legen wir in das kartesische Koordinatensystem mit Mittelpunkt M im Ursprung und den Halbachsen g und h jeweils auf der ersten und zweiten Koordinatenachse. Wegen der Ellipsensymmetrie sei o.B.d.A. der Punkt C im ersten Quadranten des Koordinatensystems. So lassen sich die Eckpunkte des Ellipsendreiecks über Koordinaten beschreiben, mit denen die Längen der Dreiecksseiten berechnet werden können. Zur Bestimmung der Punktkoordinaten

$$A = (a_x, a_y), B = (b_x, b_y), C = (c_x, c_y)$$

stellen wir Polynomgleichungen auf, die die Beziehung zwischen der Ellipse und dem Ellipsendreieck herstellen. Von dem durch diese Polynome erzeugten Ideal berechnen wir eine Gröbnerbasis, mit der wir mittels des im letzten Abschnitt behandelten Lösungsverfahrens polynomieller Gleichungssysteme die Koordinaten bestimmen.
In Anlehnung an den Beweis des obigen Satzes gehen wir zunächst von dem Kreis K aus, der das Bild der Ellipse unter der Abbildung $A(f, s)$ mit Streckungsfaktor $s = \frac{g}{h}$ und der durch g laufenden Fixpunktgeraden f darstellt. Ist $C' = (c'_x, c'_y) := A(f, s)(C)$ das Bild von C, dann bildet C' mit den Punkten $A' = (a'_x, a'_y) := A(f, s)(A)$ und $B' = (b'_x, b'_y) := A(f, s)(B)$ ein gleichseitiges Dreieck, dessen Eckpunkte auf K liegen. Die Koordinaten dieser Punkte können wir wie folgt berechnen:

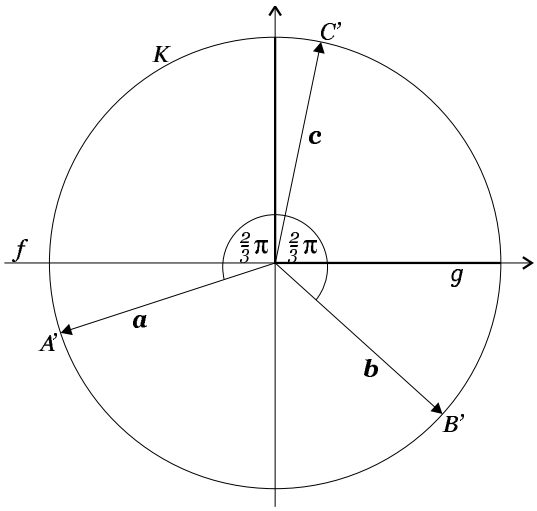


Abbildung 3.3:

Seien $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^2$ die Vektoren zu den Punkten A', B' und C' (siehe Abbildung 3.3). Zu gegebenem Vektor $\mathbf{c}$ suchen wir nun die Vektoren $\mathbf{a}$ und $\mathbf{b}$. Diese sind Lösungen der Gleichung

$$\frac{\langle \mathbf{c}, \mathbf{v} \rangle}{\|\mathbf{c}\| \cdot \|\mathbf{v}\|} = \cos\left(\frac{2}{3}\pi\right) \tag{3.1}$$

mit der Variablen $\mathbf{v} \in \mathbb{R}^2$. Dabei ist bei dieser aus der linearen Algebra bekannten Gleichung $\langle \mathbf{c}, \mathbf{v} \rangle = c_x \cdot v_x + c'_y \cdot v'_y$ das Skalarprodukt der Vektoren $\mathbf{c} = (c_x, c'_y)$ und $\mathbf{v} = (v_x, v'_y)$, $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$ die euklidische Norm und $\frac{2}{3}\pi$ der Winkel zwischen den Vektoren $\mathbf{c}$ und $\mathbf{v}$.
Umformung von (3.1) liefert eine Polynomgleichung. Um die Wurzel zu beseitigen, quadrieren wir diese Gleichung und nehmen damit zusätzliche Lösungen in Kauf:

$$\langle \mathbf{c}, \mathbf{v} \rangle^2 = \frac{1}{4}\|\mathbf{c}\|^2 \cdot \|\mathbf{v}\|^2,$$

also

$$4(c_x \cdot v_x + c'_y \cdot v'_y)^2 = (c_x^2 + {c'_y}^2)(v_x^2 + {v'_y}^2). \tag{3.2}$$

Da die Fixpunktgerade f mit der ersten Koordinatenachse übereinstimmt, sind die x-Koordinaten der Dreieckspunkte $A(f, s)$-invariant, was insbesondere auch $c'_x = c_x = l$ bedeutet.
Gleichung (3.2) legt nur die Richtung des Vektors $\mathbf{v}$ fest. Da aber auch die Längen der Vektoren $\mathbf{c}$ und $\mathbf{v}$ mit dem Radius von K übereinstimmen sollen, d. h. $\|\mathbf{v}\| = \|\mathbf{c}\| = g$, erhalten wir die Gleichungen

$$c_x^2 + {c'_y}^2 = g^2, \tag{3.3}$$

$$v_x^2 + {v'_y}^2 = g^2. \tag{3.4}$$

Hieraus würden sich nun die Koordinaten der Dreieckspunkte auf dem Kreis bestimmen lassen. Uns interessieren aber eigentlich die Punkte auf der Ellipse, also die Urbilder der Punkte A', B' und C' unter der Abbildung $A(f, s)$. Wie oben schon erwähnt,

bleiben die x-Koordinaten unter $A(f,s)$ unverändert, d. h.

$$a'_x = a_x,\ b'_x = b_x,\ c'_x = c_x.$$

Diese Gleichungen werden wir nicht als Polynome in unser Erzeugendensystem aufnehmen und verwenden nur die Namen ohne Strich, wie in den obigen Gleichungen bereits geschehen. Die y-Koordinaten der Dreieckspunkte ändern sich unter $A(f,s)$ dagegen sehr wohl. Mit dem Streckungsfaktor $s = \frac{g}{h}$ erhalten wir folgende Gleichungen:

$$h \cdot s = g, \tag{3.5}$$
$$c_y \cdot s = c'_y, \tag{3.6}$$
$$v_y \cdot s = v'_y, \tag{3.7}$$

wobei sich für v_y die y-Koordinaten der Punkte A und B auf E ergeben werden.
In Gleichung (3.1) dividieren wir durch die Länge der Vektoren $\mathbf{c}$ und $\mathbf{v}$, die mit der Länge der Ellipsenhalbachse g übereinstimmt. Mit w als weiterer Variable fordern wir durch die Gleichung

$$g \cdot w = 1, \tag{3.8}$$

dass g und damit $\|\mathbf{c}\|$ bzw. $\|\mathbf{v}\|$ von Null verschieden sind. Die Berücksichtigung eines hieraus resultierenden Polynoms als Erzeugerpolynom ist üblich bei der rationalen Implizitierung (siehe [CLO98] Kapitel 3 §3, insbesondere Theorem 2 auf Seite 130). Uns wird Gleichung (3.8) vor allem die weiteren Berechnungen vereinfachen.

Wir erhalten gemäß den Gleichungen (3.2) bis (3.8) folgende Erzeugerpolynome für unser Ideal I. Dabei steht der Buchstabe `k` in einigen Variablen für „Kreis“ und soll dem Strich in obiger Namensgebung entsprechen, also `cyk` $= c'_y$.

```
In[68]:= p1 = 4 (cx * vx + cyk * vyk)^2 - (cx^2 + cyk^2) (vx^2 + vyk^2);
         p2 = cx^2 + cyk^2 - g^2;
         p3 = vx^2 + vyk^2 - g^2;
         p4 = g - h * s;
         p5 = cyk - cy * s;
         p6 = vyk - vy * s;
         p7 = g * w - 1;

In[69]:= G = {p1, p2, p3, p4, p5, p6, p7};
```

Nun bestimmen wir die reduzierte Gröbnerbasis von I bezüglich *lex*-Ordnung mit der Variablenreihenfolge `w` > `s` > `vyk` > `cyk` > `vy` > `cy`. Die Variablen `g`, `h` und `cx` $= l$ betrachten wir als Parameter. Da die Berechnung einer Gröbnerbasis von 7 Erzeugerpolynomen in 7 Variablen etwas aufwendiger sein wird, benutzen wir dafür die *Mathematica* eigene Funktion `GroebnerBasis`. Diese ist im Kern von *Mathematica* implementiert und deshalb um einiges schneller als unsere in Kapitel 2 implementierten Algorithmen. Der Hauptgeschwindigkeitsvorteil resultiert aber aus dem integrierten *GröbnerWalk*-Algorithmus, der, wie bereits in Beispiel 2.5.1 auf Seite 31 erwähnt, unter *lex*-Ordnung wesentlich effektiver arbeitet (siehe auch [Tra00]).

```
In[70]:= gb = GroebnerBasis[G,
              {w, s, vyk, cyk, vy, vx, cy}]; //Timing
Out[70]= {27.13 Second, Null}
```

```
In[71]:= Length[gb]
Out[71]= 58
```

Wie wir sehen, benötigt selbst die *Mathematica*-Funktion etwa eine halbe Minute zur Berechnung der reduzierten Gröbnerbasis.[2] Diese besteht aus 58 Polynomen, die wir aus Platzgründen hier unterschlagen haben.
Suchen wir uns nun aus dieser Gröbnerbasis die Polynome heraus, die nur die Variable **cy** und natürlich die Parameter **g**, **h** und **cx** enthalten, die also nach dem Eliminationssatz (Satz 22 auf Seite 54) eine Gröbnerbasis des Ideals $I_6 = I \cap \mathbb{C}[\mathbf{cy}]$ bilden.

```
In[72]:= G6 = Select[gb,
             FreeQ[#, Alternatives[vx, vy, cyk,
                  vyk, s, w]]&]
Out[72]= {-cy^2 g^2 - cx^2 h^2 + g^2 h^2}
```

Das Eliminationsideal I_6 wird nur von einem Polynom erzeugt. Dieses setzen wir gleich Null und lösen die resultierende Gleichung nach **cy** auf.

```
In[73]:= Reduce[{G6[[1]] == 0}, cy]
```

$$\text{Out[73]= } cx == 0 \&\& g == 0 \,||\, cy == -\sqrt{1 - \frac{cx^2}{g^2}}\, h \&\& g \neq 0 \,||\, cy == \sqrt{1 - \frac{cx^2}{g^2}}\, h \&\& g \neq 0 \,||\, g == 0 \&\& h == 0$$

Wir erhalten zwei Lösungen für **cy**, wobei wir nur die positive berücksichtigen. Durch die Forderung $g \neq 0$ verlassen wir das Ideal und die Polynomarithmetik. Für das weitere Vorgehen stellt das aber keine Einschränkung dar, da für unsere geometrischen Betrachtungen die Länge der Halbachse g positiv sein soll. Speichern wir zunächst dieses Ergebnis in einer leicht modifizierten Form ab.

```
In[74]:= CY = (h/g) Sqrt[g^2 - cx^2];
```

In dieser Formel finden wir die Normaldarstellung der Ellipsengleichung wieder, denn umgeformt ergibt sich aus ihr mit $x := c_x$ und $y := c_y$

$$\frac{x^2}{g^2} + \frac{y^2}{h^2} = 1.$$

Weiter geht es mit der Bestimmung der x-Koordinaten der Punkte A und B. Suchen wir uns also die Gröbnerbasis von $I_5 = I \cap \mathbb{C}[\mathbf{vx}, \mathbf{cy}]$.

```
In[75]:= G5 = Select[gb,
             FreeQ[#, Alternatives[vy, cyk, vyk,
                  s, w]]&]
Out[75]= {-cy^2 g^2 - cx^2 h^2 + g^2 h^2, 16 cx^4 - 24 cx^2 g^2 +
          9 g^4 + 16 cx^2 vx^2 - 24 g^2 vx^2 + 16 vx^4}

In[76]:= pol5 = Factor[G5[[2]]]
```

[2] Der Algorithmus **BuchbergerSugar** würde mehrere Stunden für die Berechnung benötigen, da er ohne „GröbnerWalk“ arbeitet.

```
Out[76]= (4 cx^2 - 3 g^2 - 4 cx vx + 4 vx^2)
           (4 cx^2 - 3 g^2 + 4 cx vx + 4 vx^2)
```

Abgesehen von dem I_6 erzeugenden Polynom besteht die Gröbnerbasis von I_5 nur aus einem weiteren Polynom. Dieses lässt sich faktorisieren, was auf die Quadrierung der Gleichung (3.2) zurückzuführen ist. Der eine Faktor bezieht sich auf einen Punkt C im ersten Quadranten des Koordinatensystems und der andere auf einen Punkt C im dritten Quadranten als Bild einer 180°-Drehung der Ellipse. Es wird sich später zeigen, dass der zweite Faktor der für uns interessante ist, der sich also auf C im ersten Quadranten bezieht.
Erwähnenswert ist noch, dass das Polynom frei von der Variablen **cx** ist. Folglich brauchen wir uns auch keine Gedanken darüber zu machen, ob der Erweiterungsschritt scheitern wird. Setzen wir also den zweiten Faktor Null und lösen nach **vx** auf.

```
In[77]:= ls5 = Reduce[pol5[[2]] == 0, vx]
Out[77]= vx == 1/2 (-cx - Sqrt[3] Sqrt[-cx^2 + g^2]) ||
           vx == 1/2 (-cx + Sqrt[3] Sqrt[-cx^2 + g^2])
```

Wieder erhalten wir zwei Lösungen, die zu a_x bzw. b_x korrespondieren. Damit das Ellipsendreieck $\triangle ABC$ die richtige Orientierung erhält (Beschriftung der Eckpunkte entgegen dem Uhrzeigersinn) sollte $a_x < b_x$ sein.

```
In[78]:= {AX, BX} =
           vx /. List@@SequenceHold[ToRules[ls5]]
Out[78]= {1/2 (-cx - Sqrt[3] Sqrt[-cx^2 + g^2]),
           1/2 (-cx + Sqrt[3] Sqrt[-cx^2 + g^2])}
```

Zur Bestimmung der y-Koordinaten der Punkte A und B suchen wir uns nun die Gröbnerbasis des Eliminationsideals $I_4 = I \cap \mathbb{C}[\mathbf{vy}, \mathbf{vx}, \mathbf{cy}]$ heraus. Dabei wollen wir gleich die Erzeuger ohne enthaltene Variable **vy** außer Acht lassen.

```
In[79]:= G4 = Drop[Select[gb,
                FreeQ[#, Alternatives[cyk, vyk, s,
                    w]]&], 2];
          Length[G4]
Out[79]= 17

In[80]:= Collect[G4[[1]], vy]
Out[80]= 8 cx^4 cy vx - 6 cx^2 cy g^2 vx - 3 cy g^4 vx +
           4 cy g^2 vx^3 + (-8 cx^5 + 14 cx^3 g^2 - 6 cx g^4) vy
```

Diesmal erhalten wir ganze 17 Polynome, von denen wir uns zunächst das erste für die weiteren Berechnungen hernehmen. Wir bestimmen mit diesem die y-Koordinate des Punktes A, indem wir die schon berechneten Variablen $c_y \hat{=}$ **CY** bzw. $a_x \hat{=}$ **AX** im Polynom substituieren, dies Null setzen und nach **vy** auflösen. Auch hier sichert uns der Erweiterungssatz die Existenz von Lösungen zu, da der Leitkoeffizient $-8c_x^5 + 14c_x^3 g^3 - 6c_x g^4$ frei von den Varablen **vx** und **cy** ist.

```
In[81]:= ls4a = Reduce[
              {(G4[[1]] /. {vx → AX, cy → CY}) == 0}, vy]
```

```
Out[81]= cx == 0&&g ≠ 0||cx == -g&&g ≠ 0||
          cx == g&&g ≠ 0||2 cx == -√3 g&&g ≠ 0||
          2 cx == √3 g&&g ≠ 0||
          vy == (√3 cx h - √(-cx^2 + g^2) h)/(2 g)&&g ≠ 0
```

```
In[82]:= AY = vy/.ToRules[(List@@ls4a)[[-1]]]
```

$$\mathit{Out[82]}= \frac{\sqrt{3}\,cx\,h-\sqrt{-cx^2+g^2}\,h}{2\,g}$$

Nachdem wir das Ergebnis wieder abgespeichert haben, verfahren wir analog bei der Bestimmung von b_y. Diesmal substituieren wir jedoch **vx** mit dem berechneten $b_x \hat{=}$ **BX**.

```
In[83]:= ls4b = Reduce[
             {(G4[[1]]/.{vx → BX, cy → CY}) == 0}, vy]
Out[83]= cx == 0&&g ≠ 0||cx == -g&&g ≠ 0||
          cx == g&&g ≠ 0||2 cx == -√3 g&&g ≠ 0||
          2 cx == √3 g&&g ≠ 0||
          vy == (-√3 cx h - √(-cx^2 + g^2) h)/(2 g)&&g ≠ 0
```

```
In[84]:= BY = vy/.ToRules[(List@@ls4b)[[-1]]]
```

$$\mathit{Out[84]}= \frac{-\sqrt{3}\,cx\,h-\sqrt{-cx^2+g^2}\,h}{2\,g}$$

Prüfen wir nun, ob die berechneten Ergebnisse auch mit den anderen oben herausgesuchten Polynomen vereinbar sind.

```
In[85]:= Together[G4/.{vy → AY, vx → AX, cy → CY}]
         Together[G4/.{vy → BY, vx → BX, cy → CY}]
Out[85]= {0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
Out[85]= {0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Sowohl (**AY**, **AX**, **CY**) als auch (**BY**, **BX**, **CY**) sind Nullstellen sämtlicher Polynome aus I_4. Da wir die Koordinaten der Dreieckspunkte über die Längen g, h und l = **cx** ausgedrückt haben und uns die Variablen **cyk**, **vyk**, **s** und **w** nicht weiter interessieren, bestimmen wir nun die Längen der Dreiecksseiten. Schreiben wir uns hierzu eine Funktion.

```
In[86]:= TrianglePoints[g_, h_, cx_] :=
           Evaluate[{{AX, AY}, {BX, BY}, {cx, CY}}]
```

```
In[87]:= EllipsesTriangle[g_, h_, l_] :=
           Module[{pA, pB, pC},
             {pA, pB, pC} = TrianglePoints[g, h, l];
             Sqrt[{(pB - pC).(pB - pC),
                 (pA - pC).(pA - pC), (pA - pB).(pA - pB)}]
           ]/; And@@Positive[{g, h}]&&0 ≤ l ≤ g||
             Not[And@@NumericQ/@{g, h, l}]
```

Die Funktion **EllipsesTriangle** berechnet zu gegebenen Ellipsenhalbachsen g und h und dem Abstand l des Dreieckspunktes C von h die Seitenlängen des gesuchten

Ellipsendreiecks $\triangle ABC$. Dazu ruft sie die Funktion **TrianglePoints** auf, die die Koordinaten der Dreieckspunkte mit Hilfe unserer berechneten Formeln bestimmt und berechnet den Abstand dieser Punkte.

```
In[88]:= Together[EllipsesTriangle[g, h, l]]
```

$$\begin{aligned}\text{Out[88]}= \Big\{ &\frac{1}{2}\sqrt{3}\sqrt{\Big(-\frac{1}{g^2}\big(-g^4-3g^2h^2-2g^2l^2+2h^2l^2+\\ &\qquad 2\sqrt{3}g^2l\sqrt{g^2-l^2}-\\ &\qquad 2\sqrt{3}h^2l\sqrt{g^2-l^2}\big)\Big)},\\ &\frac{1}{2}\sqrt{3}\sqrt{\Big(\frac{1}{g^2}\big(g^4+3g^2h^2+2g^2l^2-\\ &\qquad 2h^2l^2+2\sqrt{3}g^2l\sqrt{g^2-l^2}-\\ &\qquad 2\sqrt{3}h^2l\sqrt{g^2-l^2}\big)\Big)},\\ &\sqrt{3}\sqrt{-\frac{-g^4+g^2l^2-h^2l^2}{g^2}}\Big\}\end{aligned}$$

Dies sind nun die Formeln, mit denen sich die Seitenlängen des Ellipsendreiecks zu gegebenen Ellipsenhalbachsenlängen und dem Abstand des Dreieckspunktes C von einer Halbachse bestimmen lassen. Da der *Mathematica*-Output hier nicht sehr übersichtlich ist und sich die Formeln noch ein wenig umformen lassen, fassen wir sie nochmals zusammen.

$$a = \frac{\sqrt{3}}{2g}\sqrt{g^2(g^2+3h^2)+2l(g^2-h^2)(l-\sqrt{3}\sqrt{g^2-l^2})}, \tag{3.9}$$

$$b = \frac{\sqrt{3}}{2g}\sqrt{g^2(g^2+3h^2)+2l(g^2-h^2)(l+\sqrt{3}\sqrt{g^2-l^2})}, \tag{3.10}$$

$$c = \frac{\sqrt{3}}{g}\sqrt{g^2(g^2-l^2)+h^2l^2}. \tag{3.11}$$

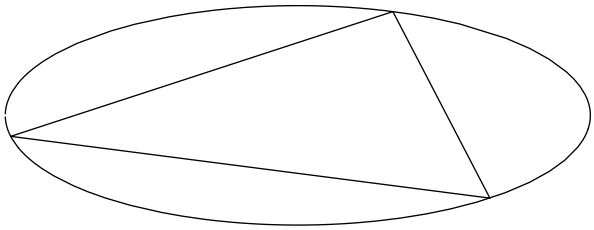
Beispiel 3.4

Jetzt wenden wir unsere Formeln an.

```
In[89]:= EllipsesTriangle[4, 1.5, 1.3]
Out[89]= {2.87422, 5.49735, 6.60628}
```

Einer Ellipse mit den Halbachsenlängen $g = 4$ cm und $h = 1.5$ cm sowie Abstand $l = 1.3$ cm des Dreieckspunktes C von h ist also ein Ellipsendreieck mit den Seitenlängen $a = 2.9$ cm, $b = 5.5$ cm und $c = 6.6$ cm einbeschrieben. Mit der im Anhang A.11 beschriebenen Funktion können wir dies auch visualisieren:

```
In[90]:= ShowEllipsesTriangle[4, 1.5, 1.3,
           Axes → False];
```



Zu Beginn dieses Abschnittes haben wir als ein Ziel die Aufstellung von Formeln formuliert, mit denen wir zu gegebenen Seitenlängen eines Dreiecks die charakteristischen Größen der zugehörigen Schwerpunktellipse bestimmen können. Diese Größen sind die Längen der Ellipsenhalbachsen g und h. Um auch das Dreieck in die Ellipse einzeichnen zu können, ist eine weitere charakteristische Größe der Abstand l des Dreieckspunktes C von h.
Aus den Gleichungen (3.9) bis (3.11) lassen sich nun durch Umformung leicht unsere gesuchten Formeln bestimmen.

```
In[91]:= gl = MapThread[Equal, {{a, b, c},
             Together[EllipsesTriangle[
                g, h, l]]}];
         sol = Solve[gl, {g, h, l}];
         Length[sol]
Out[91]= 16
```

Der **Solve**-Befehl zum Lösen von Gleichungen berechnete 16 verschiedene Lösungen, die wir aus Platzgründen hier nicht auflisten wollen. Da *Mathematica* nicht weiß, dass die Größen a, b und c positiv sind, erhalten wir bereits $2^3 = 8$ verschiedene Lösungen, jeweils mit positiven bzw. negativen Längen, die betragsmäßig übereinstimmen. Zudem wird *Mathematica* beim Lösen dieser Wurzelgleichungen quadrieren, was u. A. die Vertauschung der Halbachsen g und h zur Folge hat, so dass l einmal als Abstand von C zu h und ein weiteres Mal als Abstand von C zu g angesehen wird. Hieraus resultieren die 16 Lösungen, von denen uns aber nur eine wirklich interessiert (positive a, b, c und l als Abstand zwischen C und h). Diese eine von *Mathematica* berechnete Lösung ist immer noch so komplex, dass eine Auflistung dieser nicht lohnenswert wäre. Aber mit ein paar Umformungen ergeben sich schließlich folgende Formeln:

$$g = \frac{1}{3}\sqrt{a^2 + b^3 + c^2 + 2D}, \tag{3.12}$$

$$h = \frac{1}{3}\sqrt{a^2 + b^3 + c^2 - 2D}, \tag{3.13}$$

$$l = \frac{1}{D}\sqrt{2(2a^2 + 2b^2 - c^2)D^2 + (5a^4 + 5b^4 + 2c^4 - 2a^2b^2 - 5a^2c^2 - 5b^2c^2)D}, \tag{3.14}$$

wobei $D = \sqrt{a^4 + b^4 + c^4 - a^2b^2 - a^2c^2 - b^2c^2}$ ist. Schreiben wir uns wieder eine Funktion, die aus den Längen der Dreiecksseiten die Größen der Ellipse berechnet.

```
In[92]:= CentroidEllipse[a_, b_, c_] :=
           Module[{DD},
             DD = Sqrt[a^4 - a^2 b^2 + b^4 - a^2 c^2 - b^2 c^2 + c^4];
             {1/3 Sqrt[a^2 + b^2 + c^2 + 2DD], 1/3 Sqrt[a^2 + b^2 + c^2 - 2DD],
               1/(6 DD) Sqrt[(2 (2a^2 + 2 b^2 - c^2) DD^2 +
                   (5a^4 + 5b^4 + 2c^4 - 2a^2 b^2 - 5a^2 c^2 - 5b^2 c^2) DD)]
             }]/; And @@ Positive[{a, b, c}] &&
                  a < b + c && b < a + c && c < a + b
```

Beispiel 3.5

Schauen wir nun, ob die beiden Funktionen **EllipsesTriangle** und **CentroidEllipse** miteinander verträglich sind. Wir berechnen zunächst zu einer Ellipse mit den Ellipsenhalbachsen $g = 4$ cm, $h = 2$ cm und einem Abstand $l = 1$ cm von Punkt C zu h die Seitenlängen des zugehörigen Ellipsendreiecks.

```
In[93]:= z = EllipsesTriangle[4, 2, 1] // RootReduce
Out[93]= {Root[6921 - 708 #1^2 + 16 #1^4 &, 3],
           Root[6921 - 708 #1^2 + 16 #1^4 &, 4], Sqrt[183]/2}
```

Die **Root**-Ausdrücke entsprechen dabei folgenden algebraischen Zahlen und Dezimalbrüchen:

```
In[94]:= ToRadicals[z]
Out[94]= {Sqrt[177/8 - 27 Sqrt[5]/8], Sqrt[177/8 + 27 Sqrt[5]/8], Sqrt[183]/2}
In[95]:= N[z]
Out[95]= {3.81815, 5.44718, 6.76387}
```

Aus diesen Seitenlängen berechnen wir nun die Ellipsenhalbachsen g und h, sowie den Abstand l zwischen h und C der zugehörigen Schwerpunktellipse.

```
In[96]:= CentroidEllipse[z[[1]], z[[2]],
           z[[3]]] // RootReduce
Out[96]= {4, 2, 1}
```

Wir erhalten wieder die ursprünglichen Werte $g = 4$ cm, $h = 2$ cm und $l = 1$ cm und können somit problemlos zwischen den Seitenlängen eines Dreiecks und den Halbachsenlängen der Schwerpunktellipse hin- und herrechnen.

Jetzt wollen wir zu gegebenen Halbachsenseiten g und h einer Ellipse den Abstand des Dreieckspunktes C von h berechnen, so dass das Ellipsendreieck rechtwinklig ist, sofern dies möglich ist.
Betten wir wie oben die Ellipse wieder in das kartesische Koordinatensystem ein. Wie gehabt können wir wegen der Ellipsensymmetrie o.B.d.A. annehmen, dass der Dreieckspunkt C im ersten Quadranten liegt. Dann suchen wir uns wieder Polynome, die geometrische Eigenschaften ausdrücken und bestimmen die reduzierte Gröbnerbasis

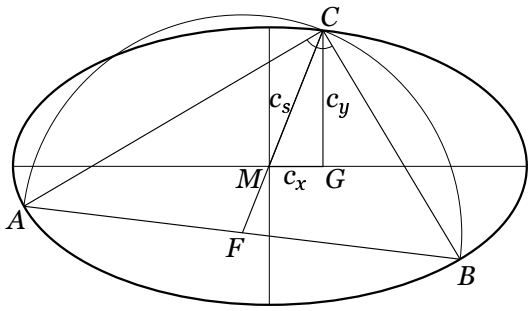


Abbildung 3.4:

des aus diesen Polynomen erzeugten Ideals, mit der wir die gesuchte Formel herleiten können.
Intuitiv würden wir die drei Wurzelgleichungen (3.9) bis (3.11) in quadrierter Form und die Pythagorasgleichung ($p = a^2 + b^2 - c^2$) als Erzeugerpolynome des Ideals verwenden. Jedoch sind die aus (3.9) und (3.10) resultierenden Polynome, die die Beziehungen zu den Dreiecksseitenlängen a und b ausdrücken, identisch, so dass diese Wahl der Polynome nicht zum Ziel führen wird. Beschreiten wir also einen anderen Weg.
Umformung von (3.11) ergibt

$$c^2g^2 = -3c_x^2g^2 + 3g^4 + 3c_xh^2 \tag{3.15}$$

und dies können wir verwenden, um eine Beziehung zwischen den Ellipsenhalbachsen g und h, dem Abstand $l = c_x$ zwischen C und h und der Dreiecksseite c herzustellen. Die Beziehung zu c_y erhalten wir durch das erste Gröbnerbasenpolynom

$$-c_y^2g^2 - c_x^2h^2 + g^2h^2 \tag{3.16}$$

unserer oben berechneten Gröbnerbasis **gb** (siehe `Out[72]`, Seite 62). Als weitere Größe betrachten wir die Seitenhalbierende c_s zur Dreiecksseite c. Diese wird vom Schwerpunkt des Dreiecks im Verhältnis 2:1 geteilt. So erhalten wir über das rechtwinklige Dreieck $\triangle MGC$ (siehe Abbildung 3.4) mittels Pythagoras die Gleichung

$$c_x^2 + c_y^2 = \left(\frac{2}{3}c_s\right)^2. \tag{3.17}$$

Jetzt fehlt nur noch die Eigenschaft, dass das Dreieck $\triangle ABC$ im Punkt C einen rechten Winkel hat. Die entsprechende Gleichung lässt sich leicht über den Thaleskreis aufstellen. Betrachten wir den Kreisbogen um den Fußpunkt F von c_s auf c mit Radius $\frac{c}{2}$, der durch die Punkte A und B verläuft (siehe Abbildung 3.4). Ein rechter Winkel im Punkt C liegt genau dann vor, wenn auch C auf dem Kreisbogen liegt. Dies ist der Fall, wenn die Seitenhalbierende c_s die Länge des Kreisradius hat, wenn also

$$c_s = \frac{1}{2}c \tag{3.18}$$

gilt.
Nehmen wir uns nun die sich aus (3.15) bis (3.18) ergebenden Polynome her und

berechnen von ihnen die reduzierte Gröbnerbasis bezüglich *lex*-Ordnung mit der Variablenreihenfolge $c_s > c_y > c > c_x$ und den Parametern g und h.

```
In[97]:= q1 = -c^2 g^2 - 3 cx^2 g^2 + 3 g^4 + 3 cx^2 h^2;
         q2 = gb[[1]];
         q3 = cx^2 + cy^2 - (2/3 cs)^2;
         q4 = c - 2cs;

In[98]:= gb2 = GroebnerBasis[{q1, q2, q3, q4},
           {cs, cy, c, cx}]
Out[98]= {-4 cx^2 g^2 + g^4 + 4 cx^2 h^2 - 3 g^2 h^2,
          4 c^2 g^2 - 9 g^4 - 9 g^2 h^2,
          16 c^2 cx^2 h^2 - 9 g^4 h^2 - 72 cx^2 h^4 + 27 g^2 h^4,
          -c^2 + 9 cx^2 + 9 cy^2, c - 2 cs}
```

Nur das erste Polynom ist für uns interessant, da es als einziges nur die Variablen g, h und c_x enthält. Setzen wir es gleich Null und lösen die daraus resultierende Gleichung nach c_x auf.

```
In[99]:= Reduce[gb2[[1]] == 0, cx]
Out[99]= g == 0&&h == 0||
           cx == -1/2 g Sqrt[3 - 8 g^2/(4 g^2 - 4 h^2)]&&g - h != 0&&
             g + h != 0||cx == 1/2 g Sqrt[3 - 8 g^2/(4 g^2 - 4 h^2)]&&
             g - h != 0&&g + h != 0
```

Da $c_x = l$ positiv sein soll, erhalten wir eine eindeutige Lösung, mit der wir auch gleichzeitig eine Formel für l haben:

$$l = \frac{1}{2} g \sqrt{3 - \frac{2g^2}{g^2 - h^2}}. \tag{3.19}$$

Mathematica fügt dieser Lösung noch zwei Bedingungen an. Die Bedingung $g + h \neq 0$ ist ohnehin erfüllt, da g und h positiv sind. Die Bedingung $g - h \neq 0$ bedeutet, dass die Ellipse kein Kreis sein darf. In diesem Fall wäre unser Ellipsendreieck gleichseitig, hätte also keinen rechten Winkel. Diese Bedingung können wir erweitern, so dass mit Sicherheit ein rechtwinkliges Ellipsendreieck existiert. Das ist genau dann der Fall, wenn der Ausdruck unter der Wurzel in (3.19) nicht negativ ist, woraus sich die Bedingung

$$h \leq \frac{g}{\sqrt{3}}$$

an die Halbachsen ergibt. Damit ist h als die kleinere Halbachse festgesetzt, was aber wegen der Ellipsensymmetrie keine Einschränkung darstellt.
Aus (3.19) ergibt sich auch noch eine Eigenschaft des Abstandes l:

$$0 \leq l < \frac{1}{2} g.$$

Abrunden wollen wir diesen Abschnitt mit der Bestimmung unserer hergeleiteten Formeln mit den Möglichkeiten, die *Mathematica* uns bietet, denn natürlich besitzt auch dieses Computeralgebrasystem eine Funktion zum Lösen polynomieller Gleichungssysteme. Dieser Funktion mit Namen **Solve** übergeben wir nun die anfangs aufgestellten Polynome, die unter *In[69]* auf Seite 61 in der Variablen **G** als Polynomliste zusammengefasst sind.

```
In[100]:= (sol = Solve[G == 0, {cy, vx, vy},
                {cyk, vyk, s, w}]);//Timing
          sol//Length
Out[100]= {3.89 Second, Null}
Out[100]= 8
```

Wir erhalten acht verschiedene Lösungen, die wesentlich komplexer ausfallen als unsere hergeleiteten. Lassen wir die Vorzeichen unbeachtet, haben alle etwa folgende Gestalt:

```
In[101]:= la = {vx/.sol[[5]], vy/.sol[[5]],
               cy/.sol[[5]]}//Together
```

$$
\begin{aligned}
\text{Out[101]}= \Big\{ & \frac{1}{2}\left(-cx+\sqrt{3}\sqrt{-cx^2+g^2}\right), \\
& -\frac{1}{g}\Big(\sqrt{\Big(-\frac{1}{4}cx^2h^2+g^2h^2+} \\
& \qquad \frac{1}{2}\sqrt{3}\,cx\sqrt{-cx^2+g^2}\,h^2- \\
& \qquad \frac{3}{4}\left(-cx^2+g^2\right)h^2\Big)\Big), \frac{1}{g\,(4cx^2-g^2)} \\
& \Big(2\Big(cx^2\sqrt{\Big(-\frac{1}{4}cx^2h^2+g^2h^2+\frac{3}{4}\left(cx^2-g^2\right)} \\
& \qquad h^2+\frac{1}{2}\sqrt{3}\,cx\sqrt{-cx^2+g^2}\,h^2\Big)- \\
& g^2\sqrt{\Big(-\frac{1}{4}cx^2h^2+g^2h^2+\frac{3}{4}\left(cx^2-g^2\right)} \\
& \qquad h^2+\frac{1}{2}\sqrt{3}\,cx\sqrt{-cx^2+g^2}\,h^2\Big)+ \\
& \sqrt{3}\,cx\sqrt{-cx^2+g^2} \\
& \quad \sqrt{\Big(-\frac{1}{4}cx^2h^2+g^2h^2+\frac{3}{4}\left(cx^2-g^2\right)h^2+} \\
& \qquad \frac{1}{2}\sqrt{3}\,cx\sqrt{-cx^2+g^2}\,h^2\Big)\Big)\Big)\Big\}
\end{aligned}
$$

Die Variable **vx** steht für die x-Koordinaten der Punkte A und B. Folglich gehören immer zwei Lösungen zusammen. Wir haben also zwei Lösungen herauszusuchen, die mit unseren hergeleiteten Formeln korrespondieren, und sie wenn möglich noch auf deren Gestalt zu bringen.
Nach eingehender Betrachtung stellt sich jedoch heraus, dass wir mehr Lösungen benötigen. In dem **cy** entsprechenden Ausdruck von *Out[101]* (drittes Listenelement von **la**) wird durch $4c_x^2 - g^2$ dividiert. Ist also $c_x = \frac{1}{2}g$, existiert nach den **Solve**-Ergebnissen kein Ellipsendreieck, da durch Null geteilt wird. Es stellt sich nun folgendes heraus:

- Für $c_x \in [0, \frac{1}{2}g)$ sind die dritte und die fünfte **Solve**-Lösung die gewünschten.

- Für $c_x \in (\frac{1}{2}g, g]$ sind die zweite (entspricht der an der x-Achse gespiegelten dritten Lösung) und die fünfte **Solve**-Lösung die gewünschten.
- Für $c_x = \frac{1}{2}g$ wäre noch eine zusätzliche Lösungsformel zu konstruieren, etwa $A = (0, -g)$, $B = \left(c_x, -\frac{h}{g}\sqrt{g^2 - c_x^2}\right)$ und $C = \left(c_x, \frac{h}{g}\sqrt{g^2 - c_x^2}\right)$.

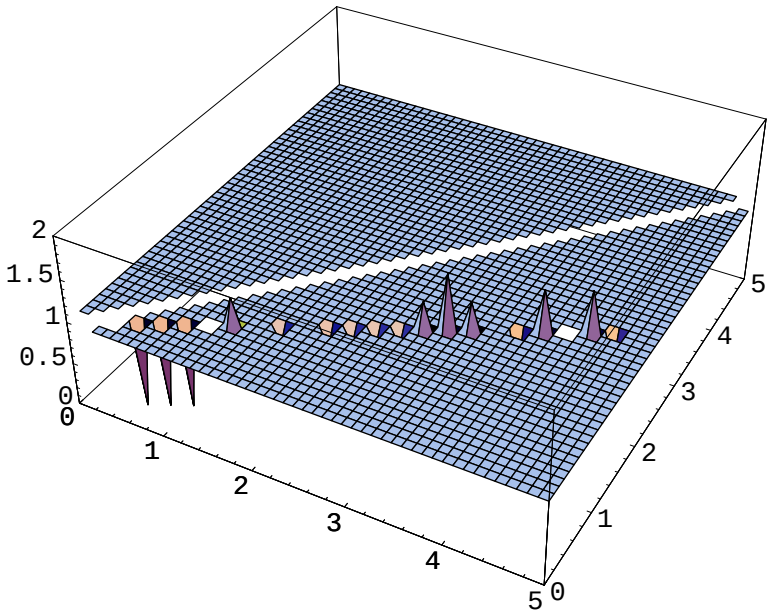
Dies ist natürlich nicht das Ergebnis, das wir uns vorgestellt haben. Zudem hat *Mathematica* Schwierigkeiten damit, die Formeln zu vereinfachen. Dividieren wir etwa die fünfte **Solve**-Lösung von **cy** durch unsere entsprechende Formel, so sollte sich dieser Quotient mit den entsprechenden Einschränkungen an die Variablen zu 1 reduzieren. Doch selbst *Mathematica*s **FullSimplify**-Funktion scheitert daran.

```
In[102]:= div = FullSimplify[la[[3]]/CY,
                g > 0&&h > 0&&g/2 > cx ≥ 0]
```

$$\text{Out[102]}= \frac{cx^2 - g^2 + \sqrt{3}\,cx\,\sqrt{-cx^2 + g^2}}{(4\,cx^2 - g^2)\,\sqrt{\frac{-cx^2+g^2}{g^2+2\,cx\left(cx+\sqrt{3}\sqrt{-cx^2+g^2}\right)}}}$$

Immerhin kann *Mathematica* die Variable h eliminieren, so dass wir uns zumindest mittels einer 3D-Visualisierung davon überzeugen können, dass für $c_x \in [0, \frac{1}{2}g)$ der Quotient gleich 1 ist. Fassen wir den Quotienten als Funktion in den Variablen c_x und g auf. Die wegen der Nulldivision auftretenden Fehlermeldungen unterschlagen wir hier.

```
In[103]:= Plot3D[div, {g, 0, 5}, {cx, 0, 5},
            PlotPoints → 51, PlotRange → {0, 2}];
```



Auf der vorderen Achse ist g abgetragen und auf der nach hinten verlaufenden Achse die Variable c_x. Für uns ist nur der Graph für $c_x < g$ interessant, also die vordere

Fläche mit den „Hütchen“. Diese Hütchen spiegeln die Singularität an den Stellen wider, für die $c_x = \frac{1}{2}g$ gilt. Abgesehen davon haben wir für $c_x \in [0, \frac{1}{2}g) \cup (\frac{1}{2}g, g]$ den erwarteten Funktionswert 1. Entsprechendes lässt sich auch für die zweite und dritte **`Solve`**-Lösung zeigen.

Zusammenfassend ist zu sagen, dass wir mit *Mathematica* zu keinen befriedigenden Formeln kommen, wenn wir das Problem direkt angehen. Erst mit dem schrittweisen Vorgehen, sinnvollen Restriktionen und der Verwendung von Gröbnerbasen erhalten wir ansprechende und geschlossene Formeln.

Anhang A

Hilfsfunktionen für die Algorithmen

Im Folgenden sind die in *Mathematica* programmierten Hilfsfunktionen aufgelistet, auf die die in Kapitel 2 beschriebenen Algorithmen zugreifen. Sämtliche Funktionen prüfen ihre Argumente nicht auf zulässige Werte. Erwartet etwa eine Funktion als Argument ein Polynom, so wird davon ausgegangen, dass sie ein solches von Null verschiedenes Polynom übergeben bekommt.

A.1 MonomialGreater

```
In[104]:= Options[MonomialGreater] =
            {MonomialOrder → Lexicographic};

          MonomialGreater[mon1_, mon2_, vars_,
             opts___?OptionQ] :=
            Module[{ord},
              ord = MonomialOrder/.{opts}/.
                  Options[MonomialGreater];
              Switch[ord,
                Lexicographic,
                LexGreater[mon1, mon2, vars],
                DegreeLexicographic,
                GrlexGreater[mon1, mon2, vars],
                DegreeReverseLexicographic,
                GrevlexGreater[mon1, mon2, vars]
              ]
            ]
```

Die Funktion `MonomialGreater` definiert eine Größenrelation auf Termen:

$$\texttt{MonomialGreater}[\texttt{m}_1, \texttt{m}_2, \texttt{vars}, \texttt{MonomialOrder} \to \texttt{ord}] = \begin{cases} \texttt{True} & : \quad \texttt{m}_1 >_{ord} \texttt{m}_2 \\ \texttt{Equal} & : \quad \texttt{m}_1 =_{ord} \texttt{m}_2 \\ \texttt{False} & : \quad \texttt{m}_1 <_{ord} \texttt{m}_2 \end{cases}$$

Dabei prüft sie nur, welche Monomordnung über die Option **MonomialOrder** festgelegt wurde und ruft für die Entscheidung die entsprechende Ordnungsfunktion auf. Unterstützt werden folgende Monomordnungen:

MonomialOrder → Lexicographic	*lex*-Ordnung,
MonomialOrder → DegreeLexicographic	*grlex*-Ordnung,
MonomialOrder → DegreeReverseLexicographic	*grevlex*-Ordnung.

Fehlt die Angabe einer Monomordnung, wird standardmäßig mit der *lex*-Ordnung gearbeitet. **LexGreater**, **GrlexGreater** und **GrevlexGreater** bilden den eigentlichen Kern von **MonomialGreater**. Diese Funktionen vergleichen die übergebenen Terme bezüglich der in Abschnitt 1.4 eingeführten Monomordnungen.

```
In[105]:= LexGreater[mon1_, mon2_, vars_] :=
            Module[{sel},
              sel = Select[Exponent[mon1, vars] -
                    Exponent[mon2, vars], # ≠ 0&];
              If[Length[sel] == 0, Equal, sel[[1]] > 0]
            ];

In[106]:= GrlexGreater[mon1_, mon2_, vars_] :=
            Module[{exp1, exp2, deg1, deg2, sel},
              exp1 = Exponent[mon1, vars];
              exp2 = Exponent[mon2, vars];
              deg1 = Plus@@exp1;
              deg2 = Plus@@exp2;
              If[deg1 == deg2,
                sel = Select[exp1 - exp2, # ≠ 0&];
                If[Length[sel] == 0, Equal,
                  sel[[1]] > 0],
                deg1 > deg2
              ]
            ];

In[107]:= GrevlexGreater[mon1_, mon2_, vars_] :=
            Module[{exp1, exp2, deg1, deg2, sel},
              exp1 = Exponent[mon1, vars];
              exp2 = Exponent[mon2, vars];
              deg1 = Plus@@exp1;
              deg2 = Plus@@exp2;
              If[deg1 == deg2,
                sel = Select[exp1 - exp2, # ≠ 0&];
                If[Length[sel] == 0, Equal,
                  sel[[-1]] < 0],
                deg1 > deg2
              ]
            ];
```

Beispiel A.1

```
In[108]:= MonomialGreater[7x^2y^4, 2x^3y^2, {x, y}]
Out[108]= False

In[109]:= MonomialGreater[7x^2y^4, 2x^3y^2, {x, y},
            MonomialOrder → DegreeLexicographic]
Out[109]= True
```

A.2 ExpandPoly

```
In[110]:= ExpandPoly[poly_, vars_] :=
            Fold[Expand, poly, vars];
```

Diese kleine Funktion multipliziert Polynome bezüglich ihrer Variablen aus.

Beispiel A.2

```
In[111]:= p = (a + 1)^10 (-x y^3 + c x^2)^2;

In[112]:= ExpandPoly[p, {x, y}]
Out[112]= (1 + a)^10 c^2 x^4 -
            2 (1 + a)^10 c x^3 y^3 + (1 + a)^10 x^2 y^6
```

A.3 LT

```
In[113]:= LT[poly_, vars_, opts___?OptionQ] :=
            Module[{p, n, lt},
              p = ExpandPoly[poly, vars];
              If[Head[p] =!= Plus, Return[p]];
              n = Length[p];
              lt = p[[1]];
              Do[
                If[MonomialGreater[p[[i]], lt, vars,
                    opts],
                  lt = p[[i]], , lt+= p[[i]]
                ],
                {i, 2, n}];
              Collect[lt, vars]
            ];
```

LT berechnet den Leitterm des an **poly** übergebenen Polynoms. **vars** und **opts** entsprechen dabei den Parametern von **MonomialGreater**.

Beispiel A.3

```
In[114]:= LT[p, {x, y}]
Out[114]= (1 + a)^10 c^2 x^4

In[115]:= LT[p, {x, y},
            MonomialOrder →
              DegreeReverseLexicographic]
Out[115]= (1 + a)^10 x^2 y^6

In[116]:= LT[a(x + y)^2 - b x^2, {x, y},
            MonomialOrder → DegreeLexicographic]
Out[116]= (a - b) x^2
```

A.4 LCMMonomial

```
In[117]:= LCMMonomial[mon1_, mon2_, vars_] :=
            Times@@
              (vars^
                 Max/@Transpose[
                     {Exponent[mon1, vars],
                      Exponent[mon2, vars]}]);
```

Die Hilfsfunktion **LCMMonomial** berechnet das kleinste gemeinsame Vielfache der Terme **mon1** $= a\,x_1^{\alpha_1}\dots x_n^{\alpha_n}$ und **mon2** $= b\,x_1^{\beta_1}\dots x_n^{\beta_n}$. Hier ist die Vorgehensweise der Funktion nicht ganz so offensichtlich.
Zuerst werden die Exponententupel beider Terme in einer Liste gespeichert,

$$\{\{\alpha_1,\dots,\alpha_n\},\{\beta_1,\dots,\beta_n\}\}.$$

Diese wird transponiert,

$$\{\{\alpha_1,\beta_1\},\dots,\{\alpha_n,\beta_n\}\}$$

und von jedem Tupel das Maximum bestimmt,

$$\{\gamma_1,\dots,\gamma_n\} \quad \text{wobei} \quad \gamma_i = \max\{\alpha_i,\beta_i\}.$$

Jetzt wird die Variablenliste mit dieser Liste potenziert,

$$\{x_1^{\gamma_1},\dots,x_n^{\gamma_n}\}$$

und der Kopf durch **Times** ersetzt, um schließlich aus der Liste ein Monom zu formen,

$$x_1^{\gamma_1}\dots x_n^{\gamma_n}.$$

Beispiel A.4

```
In[118]:= LCMMonomial[4a^2x^3y^10z^7, -c y^3x^5, {x, y, z}]
Out[118]= x^5 y^10 z^7
```

A.5 SplitTerm

```
In[119]:= SplitTerm[mon_, vars_] :=
            Module[{ko},
              ko = mon;
              Do[
                ko /= vars[[i]]^Exponent[ko, vars[[i]]],
                {i, 1, Length[vars]}];
              {ko, mon/ko}
            ]
```

SplitTerm teilt den übergebenen Term **mon** in Koeffizienten und Monom auf. Diese Werte werden als Paar in einer Liste zurückgegeben. Es gilt für $f \in k[x_1, \ldots, x_n]$:

$$\{\mathrm{LC}(f), \mathrm{LM}(f)\} = \texttt{SplitTerm[}\mathrm{LT}(f), \{x_1, \ldots, x_n\}\texttt{]}.$$

Beispiel A.5

```
In[120]:= SplitTerm[c (3 - a)^2 x^2 c y^3 z^2, {x, y, z}]
Out[120]= {(3 - a)^2 c^2, x^2 y^3 z^2}
```

In **BuchbergerRed**, siehe Seite 30, wird diese Funktion auf eine Liste von Monomen angewendet. Transponiert erhalten wir dann jeweils eine Koeffizienten- und eine Monomliste.

```
In[121]:= l = {a x, b, -x^2 y^3, (a + b)^2 y x, 2 b^4 y^2};

In[122]:= {lc, lm} =
            Transpose[
              Map[SplitTerm[#1, {x, y, z, w}]&, l]];

In[123]:= lc
Out[123]= {a, b, -1, (a + b)^2, 2 b^4}

In[124]:= lm
Out[124]= {x, 1, x^2 y^3, x y, y^2}
```

A.6 DivideQ und StrictDivideQ

```
In[125]:= DivideQ[mon1_, mon2_, vars_] :=
            Module[{e1, e2, i = 1},
              e1 = Exponent[mon1, vars];
              e2 = Exponent[mon2, vars];
              While[i ≤ Length[vars],
                If[e1[[i]] > e2[[i]], Return[False]];
                i++
              ];
              True
            ];
```

DivideQ testet, ob der Term **mon1** den Term **mon2** teilt. **StrictDivideQ** dagegen prüft, ob **mon1** ein echter Teiler von **mon2** ist.

```
In[126]:= StrictDivideQ[mon1_, mon2_, vars_] :=
            Module[{e1, e2, i = 1},
              e1 = Exponent[mon1, vars];
              e2 = Exponent[mon2, vars];
              While[i ≤ Length[vars],
                If[e1[[i]] > e2[[i]], Return[False]];
                i++
              ];
              If[e1 === e2, Return[False]];
              True
            ];
```

Beispiel A.6

```
In[127]:= DivideQ[√2a x^2 y^3 z^11, b^n x^11 y^3 z^20, {x, y, z}]
          DivideQ[x y z^2, x y z^2, {x, y, z}]
          StrictDivideQ[x y z^2, x y z^2, {x, y, z}]
Out[127]= True
Out[127]= True
Out[127]= False
```

A.7 TotalDegree

```
In[128]:= TotalDegree[poly_Plus, vars_] :=
            Max[
              Plus@@@
                (Exponent[#, vars]&/@
                   (List@@ExpandPoly[poly, vars]))];

          TotalDegree[mon_, vars_] :=
            Plus@@Exponent[mon, vars]
```

TotalDegree berechnet den totalen Grad des Polynoms **poly**, also das Maximum über die aufsummierten Exponenten. Besteht das übergebene Polynom nur aus einem Term, wird die vereinfachte Version der Funktion benutzt, die lediglich die Summe über die Exponenten der Variablen bestimmt.

Beispiel A.7

```
In[129]:= TotalDegree[3a^3 x^4 y^7 + x^2 z^10, {x, y, z}]
Out[129]= 12
```

A.8 SortCriticalPairs

```
In[130]:= SortCriticalPairs[B_, vars_,
            opts___?OptionQ] :=
          Flatten[
           Map[Sort[#, #1[[1, 2]] < #2[[1, 2]]&]&,
            Level[
             Map[
              Split[#,
                  MonomialGreater[#2[[3]], #1[[3]],
                       vars, opts] == Equal&]&,
              Map[
               Sort[#, MonomialGreater[#2[[3]],
                      #1[[3]], vars, opts]&]&,
               Split[Sort[B, #1[[2]] < #2[[2]]&],
                #1[[2]] == #2[[2]]&]]], {2}]], 1]
```

Für den Buchbergeralgorithmus mit Sugar-Strategie benötigen wir eine bestimmte Ordnung auf der Liste der kritischen Paare. Diese stellen wir mit der Sortierfunktion **SortCriticalPairs** her. Sortiert wird wie folgt:

1. nach der Größe des Sugars,
2. nach der Ordnung der kleinsten gemeinsamen Vielfachen,
3. nach der Größe des zweiten Indicis.

Wegen der Unübersichtlichkeit des Programmcodes gehen wir an einem Beispiel die wesentlichen Berechnungsschritte einzeln durch.

Beispiel A.8

Sei **G** ein System von Polynomen.

```
In[131]:= G = {x^2 y^3 z^4 - x^2, x^2 y^2 - a x, x^4 - y^2, z^4 - a z};
```

BuchbergerSugar würde unter *lex*-Ordnung hierzu intern folgende Liste von kritischen Paaren berechnen.

```
In[132]:= B = {{{1, 2}, 9, x^2 y^3 z^4}, {{1, 3}, 11, x^4 y^3 z^4},
              {{2, 3}, 6, x^4 y^2}, {{1, 4}, 9, x^2 y^3 z^4},
              {{2, 4}, 8, x^2 y^2 z^4}, {{3, 4}, 8, x^4 z^4}};
```

Zuerst wird nach der Größe des Sugars sortiert.

```
In[133]:= B1 = Sort[B, #1[[2]] < #2[[2]]&]
Out[133]= {{{2, 3}, 6, x^4 y^2}, {{3, 4}, 8, x^4 z^4},
           {{2, 4}, 8, x^2 y^2 z^4}, {{1, 4}, 9, x^2 y^3 z^4},
           {{1, 2}, 9, x^2 y^3 z^4}, {{1, 3}, 11, x^4 y^3 z^4}}
```

Dann wird die Liste partitioniert. Kritische Paare mit gleichem Sugar kommen in die gleiche Partition.

```
In[134]:= B2 = Split[B1, #1[[2]] == #2[[2]]&]
Out[134]= {{{{2, 3}, 6, x^4 y^2}},
           {{{3, 4}, 8, x^4 z^4}, {{2, 4}, 8, x^2 y^2 z^4}},
           {{{1, 4}, 9, x^2 y^3 z^4}, {{1, 2}, 9, x^2 y^3 z^4}},
           {{{1, 3}, 11, x^4 y^3 z^4}}}
```

In jeder Partition werden nun die Leitterme nach der zugrundeliegenden Monomordnung sortiert. Wir wählen hier die *lex*-Ordnung.

```
In[135]:= B3 = Map[Sort[#,
               MonomialGreater[#2[[3]], #1[[3]], {x, y, z},
                 MonomialOrder → Lexicographic]&]&, B2]
Out[135]= {{{{2, 3}, 6, x^4 y^2}},
           {{{2, 4}, 8, x^2 y^2 z^4}, {{3, 4}, 8, x^4 z^4}},
           {{{1, 4}, 9, x^2 y^3 z^4}, {{1, 2}, 9, x^2 y^3 z^4}},
           {{{1, 3}, 11, x^4 y^3 z^4}}}
```

Die kritischen Paare mit Sugar 8 wurden sortiert. Diejenigen mit Sugar 9 haben identische Leitterme. Mit einer neuen Partitionierung fassen wir die kritischen Paare zusammen, die sich bei beiden vorhergehenden Sortierkriterien als identisch herausgestellt haben.

```
In[136]:= B4 = Level[Map[Split[#,
               MonomialGreater[#2[[3]], #1[[3]], {x, y, z},
                 MonomialOrder → Lexicographic] == Equal&]&, B3],
               B3], {2}]
Out[136]= {{{{2, 3}, 6, x^4 y^2}}, {{{2, 4}, 8, x^2 y^2 z^4}},
           {{{3, 4}, 8, x^4 z^4}},
           {{{1, 4}, 9, x^2 y^3 z^4}, {{1, 2}, 9, x^2 y^3 z^4}},
           {{{1, 3}, 11, x^4 y^3 z^4}}}
```

Als drittes und letztes Sortierkriterium geben wir den kritischen Paaren $\{i, j\}$ mit kleinerem j Vorrang.

```
In[137]:= B5 = Map[Sort[#, #1[[1, 2]] < #2[[1, 2]]&]&, B4]
Out[137]= {{{{2, 3}, 6, x^4 y^2}}, {{{2, 4}, 8, x^2 y^2 z^4}},
           {{{3, 4}, 8, x^4 z^4}},
           {{{1, 2}, 9, x^2 y^3 z^4}, {{1, 4}, 9, x^2 y^3 z^4}},
           {{{1, 3}, 11, x^4 y^3 z^4}}}
```

In unserem Beispiel ist nun die Liste durch die Sortierung eindeutig bestimmt. Würde in einem anderen Beispiel nach diesen drei Sortierkriterien noch keine eindeutige Liste entstehen, wäre das für die Sugarstrategie nicht weiter tragisch.
Schließlich wird die Partitionierung wieder aufgehoben.

```
In[138]:= Flatten[B5, 1]
```

```
Out[138]= {{{2, 3}, 6, x^4 y^2}, {{2, 4}, 8, x^2 y^2 z^4},
           {{3, 4}, 8, x^4 z^4}, {{1, 2}, 9, x^2 y^3 z^4},
           {{1, 4}, 9, x^2 y^3 z^4}, {{1, 3}, 11, x^4 y^3 z^4}}
```

A.9 Merge

```
In[139]:= Merge[B1_, B2_, vars_, opts___?OptionQ] :=
           Module[{B = {}, i = 1, j = 1, lB1 = Length[B1],
               lB2 = Length[B2]},
             While[i ≤ lB1&&j ≤ lB2,
               Switch[Order[B1[[i, 2]], B2[[j, 2]]],
                 1, AppendTo[B, B1[[i++]]],
                 -1, AppendTo[B, B2[[j++]]],
                 0, If[MonomialGreater[B2[[j, 3]],
                     B1[[i, 3]], vars, opts],
                   AppendTo[B, B1[[i++]]],
                   AppendTo[B, B2[[j++]]],
                   If[B1[[i, 1, 2]] < B2[[j, 1, 2]],
                     AppendTo[B, B1[[i++]]],
                     AppendTo[B, B2[[j++]]],
                   ]]]];
             If[i ≤ lB1, B = Join[B, Drop[B1, i - 1]]];
             If[j ≤ lB2, B = Join[B, Drop[B2, j - 1]]];
             B
           ]
```

Zur Berechnung von Gröbnerbasen mit Sugar-Strategie benötigen wir auch eine Funktion, die zwei sortierte Listen mit kritischen Paaren ineinander mischt. Dies ist effizienter als die Listen zu verketten und mit **SortCriticalPairs** erneut zu sortieren. Das Ineinandermischen nach der schon bei **SortCriticalPairs** benutzten Sortierung leistet die Funktion **Merge**, bei der es sich um einen typischen Mischalgorithmus handelt.

Beispiel A.9

Nehmen wir die aus Beispiel A.8 sortierte Liste mit kritischen Paaren und mischen sie mit der sortierten Liste neuer kritischer Paare, die **BuchbergerSugar** als nächstes berechnen würde. Die kritischen Paare, die wegen der Buchbergerkriterien wegfallen würden, sind hier nicht eliminiert.

```
In[140]:= B1 = {{{2, 3}, 6, x^4 y^2}, {{2, 4}, 8, x^2 y^2 z^4},
                {{3, 4}, 8, x^4 z^4}, {{1, 2}, 9, x^2 y^3 z^4},
                {{1, 4}, 9, x^2 y^3 z^4},
                {{1, 3}, 11, x^4 y^3 z^4}};

In[141]:= B2 = {{{3, 5}, 7, x^4}, {{2, 5}, 8, x^3 y^2},
                {{4, 5}, 10, x^3 z^4}, {{1, 5}, 13, x^3 y^3 z^4}};
```

```
In[142]:= Merge[B1, B2, {x, y, z},
            MonomialOrder → Lexicographic]

Out[142]= {{{2, 3}, 6, x^4 y^2},
            {{3, 5}, 7, x^4}, {{2, 4}, 8, x^2 y^2 z^4},
            {{2, 5}, 8, x^3 y^2}, {{3, 4}, 8, x^4 z^4},
            {{1, 2}, 9, x^2 y^3 z^4}, {{1, 4}, 9, x^2 y^3 z^4},
            {{4, 5}, 10, x^3 z^4}, {{1, 3}, 11, x^4 y^3 z^4},
            {{1, 5}, 13, x^3 y^3 z^4}}
```

Nach dem Mischen erhalten wir eine neue sortierte Liste mit allen kritischen Paaren.

A.10 BuchbergerSugar

```
In[143]:= BuchbergerSugar[G_, vars_,
              opts___?OptionQ] :=
            Module[{Gb, t = Length[G], B, BNew, τ,
              cp, sred},
              Gb = {G,
                Table[TotalDegree[G[[i]], vars],
                  {i, 1, t}],
                Table[
                  SplitTerm[LT[G[[i]], vars, opts],
                    vars][[2]], {i, 1, t}]
              };
              B = Flatten[Table[{
                    {i, j},
                    Max[Gb[[2, i]]-
                          TotalDegree[Gb[[3, i]], vars],
                        Gb[[2, j]] - TotalDegree[
                            Gb[[3, j]], vars]]+
                      TotalDegree[
                        τ = LCMMonomial[Gb[[3, i]],
                            Gb[[3, j]], vars], vars],
                    τ
                  }, {j, 1, t}, {i, 1, j - 1}], 1];
              B = Delete[B,
                Position[
                  Table[B[[k, 3]]-
                      Gb[[3, B[[k, 1, 1]]]]*
                        Gb[[3, B[[k, 1, 2]]]],
                    {k, 1, Length[B]}], 0]];
              B = SortCriticalPairs[B, vars, opts];
              While[Length[B] > 0,
                cp = B[[1]];
                B = Delete[B, 1];
```

```
    sred =
      PolynomialReduce[
          SPol[Gb[[1, cp[[1, 1]]]],
            Gb[[1, cp[[1, 2]]]], vars, opts],
          Gb[[1]], vars, opts][[2]];
      If[sred =!= 0,
        t++;
        Gb = MapThread[Append,
            {Gb, {sred, cp[[2]],
               SplitTerm[LT[sred, vars, opts],
                  vars][[2]]}}];
        BNew = Table[{
             {k, t},
             Max[Gb[[2, k]]-
                  TotalDegree[Gb[[3, k]], vars],
               Gb[[2, t]] - TotalDegree[
                  Gb[[3, t]], vars]]+
              TotalDegree[
               τ = LCMMonomial[Gb[[3, k]],
                  Gb[[3, t]], vars], vars],
             τ
            }, {k, 1, t - 1}];
        B = Delete[B,
            Position[
             Table[
              StrictDivideQ[
                  BNew[[B[[k, 1, 1]], 3]],
                  B[[k, 3]], vars] &&
                StrictDivideQ[
                  BNew[[B[[k, 1, 2]], 3]],
                  B[[k, 3]], vars],
                {k, 1, Length[B]}], True]];
        BNew = Delete[BNew,
            Position[
             Table[BNew[[k, 3]]-
                Gb[[3, k]] * Gb[[3, t]],
               {k, 1, t - 1}], 0]];
        BNew = SortCriticalPairs[BNew, vars,
            opts];
        B = Merge[B, BNew, vars, opts];
       ];
      ];
      ExpandPoly[Gb[[1]], vars]
    ]
```

Erläuterungen siehe Abschnitt 2.4.

A.11 ShowEllipsesTriangle

```
In[144]:= ShowEllipsesTriangle[g_, h_, l_,
            opts___?OptionQ] :=
           Module[{pA, pB, pC},
            {pA, pB, pC} = TrianglePoints[g, h, l];
            pl1 = {
               ParametricPlot[{g(h^2 - t^2)/(h^2 + t^2), 2 h^2 t/(h^2 + t^2)},
                 {t, -200, -10},
                 DisplayFunction → Identity],
               ParametricPlot[{g(h^2 - t^2)/(h^2 + t^2), 2 h^2 t/(h^2 + t^2)},
                 {t, -10, 10},
                 DisplayFunction → Identity],
               ParametricPlot[{g(h^2 - t^2)/(h^2 + t^2), 2 h^2 t/(h^2 + t^2)},
                 {t, 10, 200},
                 DisplayFunction → Identity]
              };
            gr = Graphics[{Line[{pA, pB}],
                 Line[{pA, pC}], Line[{pB, pC}]}];
            Show[pl1, gr, AspectRatio → Automatic,
              DisplayFunction → $DisplayFunction,
              opts]
           ]/; And@@Positive[{g, h}]&&0 ≤ l ≤ g
```

Mit dieser Funktion können wir eine Ellipse mit einbeschriebenem Dreieck, dessen Schwerpunkt mit dem Ellipsenmittelpunkt übereinstimmt, visualisieren. Als Argumente verlangt die Funktion die Längen der Ellipsenhalbachsen g und h und den Abstand l des Dreieckspunktes C zu h. Die Ellipse wird mittels der Parameterdarstellung $E : \mathbb{R} \to \mathbb{R}^2$

$$t \mapsto \frac{1}{h^2 + t^2}\left(g(h^2 - t^2), 2h^2 t\right)$$

gezeichnet. Die Koordinaten der Dreieckspunkte werden über die in Abschnitt 3.2, Seite 64 beschriebene Funktion **TrianglePoints** ermittelt, mit Hilfe derer das Dreieck gezeichnet wird. Ein Beispiel ist etwa auf Seite 65 zu sehen.

Index

Literaturverzeichnis

[BW93] Thomas Becker and Volker Weispfenning. *Gröbner Bases, A Computational Approach to Commutative Algebra*. Springer, 1993.

[CKM97] S. Collart, M. Kalkbrener, and D. Mall. Converting Bases with the Gröbner Walk. *Journal of Symbolic Computation*, 24:465–469, 1997.

[CLO98] David Cox, John Little, and Donald O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 2nd edition, 1998.

[Cox80] H. S. M. Coxeter. *Introduction to Geometry*, volume 2. John Wiley & Sons, 1980.

[Eis95] David Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*. Springer, 1995.

[Fei91] Martin Feinberg. Some recent results in chemical reaction network theory. In *Patterns and dynamics in reactive media, Lect. Workshop, Minneapolis/MN (USA) 1989, IMA Vol. Math. Appl. 37, 43-70* . 1991.

[Fis00] Gerd Fischer. *Lineare Algebra*, volume 12. Vieweg, 2000.

[Frö97] Ralf Fröberg. *An Introduction to Gröbner Bases*. John Wiley & Sons, 1997.

[Gat03] Karin Gatermann. Gröbner-Basis, Implementation + Verwandte. Oberseminarvortrag Uni Kassel, April 2003.

[GCL93] Keith O. Geddes, Stephen R. Czapor, and George Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1993.

[GLGV99] Maria-Jose Gonzalez-Lopez and Laureano Gonzalez-Vega. Project 3: The inverse kinematics problem in robotics. In *Cohen, Arjeh M. (ed.) et al., Some tapas of computer algebra. Berlin: Springer. Algorithms Comput. Math. 4, 305-310* . 1999.

[GMN+91] Alessandro Gioviani, Theo Mora, Gianfranco Niesi, Lorenzo Robbiano, and Carlo Traverso. "One sugar cube, please" OR Selection strategies in the Buchbergeralgorithm. *ISSAC 1991, Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation, edited by S. Watt, ACM Press*, 1991.

[Hec96] André Heck. *A Bird's-Eye View of Gröbner Bases*. www.bangor.ac.uk/ma/teaching/modules/g3m30/heck/AIHENP96.html, 1996.

[Hol00] Audun Holme. *Geometry, Our Cultural Heritage*. Springer, 2000.

[Jen97] George A. Jennings. *Modern Geometry with Applications*. Springer, 1997.

[Koe97] Wolfram Koepf. Gröbner bases and triangles. *International Journal of Computer Algebra in Mathematics Education*, 4:371–386, 1997.

[KR00] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 1*. Springer, 2000.

[Sch00] Hans Schupp. *Kegelschnitte*. div-Verlag Franzbecker, 2000.

[Tra00] Quoc-Nam Tran. A fast algorithm for Gröbner basis conversion and its applications. *Journal of Symbolic Computation*, 30(4):451–467, 2000.

[vdB94] F. van der Blij. Computing with elbows. computer algebra and geometry. In *Vacation course 1994 computer algebra. Amsterdam: CWI. CWI Syllabus. 36, 50-76*. 1994.

[vzGG99] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[Wol] Wolfram Research. *Mathematica 4.1 Dokumentation*.

Ich versichere hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

TOBIAS HOFMANN