Gröbnerbasen in Ore–Algebren

Eine Implementation zum Arbeiten mit Ore–Algebren und die Untersuchung des Gröbner–Walks als Anwendung.

Dissertation zur Erlangung des akademischen Grades eines

Doktors der Mathematik (Dr.rer.nat)

Im Fachbereich 17 (Mathematik)
der Universität Kassel

vorgelegt von Detlef Müller
Kassel, im Februar 2006

Erster Gutachter:	Prof. Dr. Wolfram Koepf
Zweiter Gutachter:	Prof. Dr. Vladimir Gerdt
Tag der Disputation:	31.05.2006

Inhaltsverzeichnis

Vorwort und Übersicht

Gröbnerbasen sind unbestreitbar ein Grundelement der Computeralgebra. Über kommutativen Polynomringen sind sie sehr gut untersucht, es gibt viel Literatur zu diesem Thema. Verläßt man aber den Bereich der kommutativen Polynomringe und geht allgemeiner zu Ore-Algebren über, so findet man sehr viel weniger zum Thema Gröbnerbasen. Ore-Algebren liegen dabei zwischen dem kommutativen Polynomring und dem allgemeinen Schiefpolynomring in mehreren Variablen. Sie entstehen auf natürliche Weise aus der Betrachtung von Operatoralgebren.

In dieser Arbeit geht es darum, Operationen und grundlegende Algorithmen für Ore-Algebren in *Mathematica* zu realisieren. Auf diese Weise entsteht eine Plattform um die speziellen Beschränkungen und Möglichkeiten dieser Algebren insbesondere im Zusammenhang mit Gröbnerbasen an praktischen Beispielen auszuloten. Der Quellcode des Paketes ist gut kommentiert und dient als ausbaubares Werkzeug zur Untersuchung von Algorithmen. Im Gegensatz zu den existierenden Paketen wird dabei explizit auf die Struktur der Ore-Algebra aufgebaut.

Bei der Beschäftigung mit diesem Thema gelangt man automatisch zu den grundlegenden Arbeiten zu Gröbnerbasen über nichtkommutativen Polynomringen von [Mora86]. Er verallgemeinerte Gröbnerbasen auf freie nichtkommutative Polynomringe.

Kandri-Rody und V. Weispfenning untersuchten 1990 in [KRW] Verallgemeinerungen auf Algebren, in denen bestimmte Relationen zwischen den Variablen auftauchen und nannten diese Algebren “of solvable type”, was hier mit “ordnungserhaltender Art” übersetzt wird.

Algebren ordnungserhaltender Art verhalten sich dabei so, daß Buchbergers Algorithmus bedingungslos eine Gröbnerbasis findet. Im allgemeinen nichtkommutativen Fall muß es dagegen gar keine Gröbnerbasis geben.

Dann werden Algebren ordnungserhaltender Art näher betrachtet, es werden auch exotischere Beispiele für solche Ore-Algebren angegeben. In [ChySal] werden zwar umfangreiche Beispiele angegeben, die aber sehr anwendungsorientiert sind. Die Möglichkeiten werden daher dort nicht ausgeschöpft.

Als Anwendung wird der Gröbner-Walk nach [AmGlKü] mit den Mitteln der Implementation realisiert. Im nichtkommutativen Fall habe ich in der Literatur keinen Versuch gesehen, diese Methode zu analysieren.

Nach der Vorstellung des Gröbner-Walk mit einem instruktiven Beispiel zeige ich für Algebren ordnungserhaltender Art in Start- und Zielordnung, daß auf dem direkten Pfad die Bedingung ordnungserhaltender Art zu sein erhalten bleibt. Damit ist die Idee eines schrittweisen Transformierens einer Basis von der einen Ordnung zur anderen auch in diesem Fall denkbar.

Der Walk selbst läßt sich auch mit meiner Plattform realisieren und funktioniert sogar in einzelnen Fällen, aber es finden sich auch Gegenbeispiele, die aufzeigen, woran die naive Umsetzung scheitert. An einem minimalen Gegenbeispiel wird aufgezeigt, warum die Gröbnerbasis nicht sicher gefunden wird.

Als Ausblick werden einige Ideen, den Walk für unseren Fall zu “retten” angesprochen, was aber den Rahmen dieser Arbeit sprengen würde.

Danksagung

Ich bedanke mich bei den Mitarbeitern des Fachbereichs 17, die als Team für ein angenehmes Arbeitsklima sorgten.

Insbesondere Prof. Dr. Wolfram Koepf, der mit Rat und Tat zur Seite stand und Dr. Andreas Klein, der in TEX-Fragen immer zu helfen wusste.

Beiden zusammen und Ute Steenmeyer rechne ich besondere Verdienste im Kampf mit meiner Rechtschreibung und für tapferes Durchsuchen von Indizes an.

1 Grundlegende Bezeichnungen

Alle auftauchenden Körper seien, wenn nichts anderes gesagt wird, im folgenden kommutativ. Ringe seien Ringe mit Einselement.

Leider sind die Bezeichnungen der in multivariaten Polynomringen vorkommenden Strukturen in der Literatur keineswegs einheitlich. Insbesondere die Begriffe “Monom” und “Term” werden je nach Literatur mit vertauschten Rollen belegt.

Sei k ein Körper. $k[X_1, \ldots, X_m]$ bezeichne den kommutativen Polynomring in den m Unbestimmten $X_1, \ldots, X_m$.

Ein *Monom* sei ein Produkt von Unbestimmten, es hat also die Gestalt

$$\prod_{i=1}^{m} X_i^{n_i},\ n_i \in \mathbb{N}_0\ .$$

Ein *Multiindex* ist ein $n \in \mathbb{N}_0^m$, mit $n = (n_1, \ldots, n_m)$, hiermit gelte folgende Definition:

$$X^n := \prod_{i=1}^{m} X_i^{n_i}$$

Ein *Term* sei ein Produkt αX^n aus einem Körperelement α und einem Monom, dabei heiße α der *Koeffizient* und X^n der *Monomanteil* des Terms.

Ein Polynom $p \in k[X_1, \ldots, X_m]$ ist eine endliche Summe von Termen der Gestalt

$$p = \sum_{i \in I} \underbrace{\alpha_i \overbrace{X^i}^{\text{“Monom”}}}_{\text{”Term”}}, I \subset \mathbb{N}_0^m \text{ endlich }, \alpha_i \in k^*, i \in I\ . \tag{1}$$

Man kann die Koeffizienten aller Terme mit übereinstimmenden Monomanteilen durch Anwenden von Kommutativität der Addition und Distributivität zusammenfassen. So ist die Gestalt (1) erreichbar.

Da die Menge der Monome linear unabhängig über k ist, ist die Darstellung (1) bis auf die Reihenfolge der Monome eindeutig.

Hat man eine Ordnung auf der Menge der Monome definiert, kann man daher die Darstellung eindeutig machen, indem man die Terme zudem nach dieser Ordnung sortiert.

Die Monome in (1) werden mit $\mathrm{M}(p)$ und die Terme mit $\mathrm{T}(p)$ bezeichnet.

Der *Leitterm* $\mathrm{LT}\,(p)$ ist als der Term mit dem größten Monomanteil definiert, das *Leitmonom* ($\mathrm{LM}\,(p)$) als das größte auftretende Monom. Mit den Bezeichnungen aus (1) erhalten wir also:

$$\begin{aligned} \mathrm{M}(p) &:= \left\{X^i, i \in I\right\} \\ \mathrm{LM}\,(p) &:= \max(\mathrm{M}(p)) \\ \mathrm{T}(p) &:= \left\{\alpha_i X^i, X^i \in \mathrm{M}(p)\right\} \\ \mathrm{LT}\,(p) &:= \alpha_i X^i, X^i = \mathrm{LM}\,(p)\ . \end{aligned}$$

Für Mengen von Polynomen werden die entsprechenden Vereinigungen bezeichnet. Beispielsweise $\mathrm{LT}\,(\mathcal{I}) := \{\mathrm{LT}\,(p) \mid p \in \mathcal{I}\}$ und $\mathrm{T}(\mathcal{I}) := \bigcup_{p\in\mathcal{I}} \mathrm{T}(p)$.

Im Falle der Ore–Algebren werden wir zwei verschiedene Arten von Unbestimmten haben:

Die herkömmlichen Variablen

$$\{X_1, \ldots, X_m\}$$

und die Operatorvariablen

$$\{D_1, \ldots, D_n\}\ ,$$

wenn keine Mißverständnisse auftauchen können, auch kurz Operatoren genannt.

Die gesamte Menge $\{X_1, \ldots, X_m, D_1, \ldots, D_n\}$ wurde bereits als die Menge der Unbestimmten bezeichnet.

2 Ore–Algebren und Monomordnungen

2.1 Ore–Algebra im univariaten Fall

Gegeben sei der Polynomring $k[X]$, ein Algebrahomomorphismus

$$\sigma : k[X] \longrightarrow k[X]$$

sowie eine σ–Derivation

$$\delta : k[X] \longrightarrow k[X] \ .$$

Dabei ist eine σ–Derivation im allgemeineren Kontext (der für die iterierten Ore–Algebren gebraucht wird) wie folgt definiert:

Definition 2.1 *Gegeben sei ein (nicht notwendig kommutativer) Ring R und ein Endomorphismus $\sigma : R \longrightarrow R$. Dann heißt eine Abbildung $\delta : R \longrightarrow R$ eine* σ–Derivation, *wenn für alle $r, r' \in R$ gilt*

$$\begin{aligned} \delta(r + \tilde{r}) &= \delta(r) + \delta(\tilde{r}) \\ \delta(r\tilde{r}) &= \sigma(r)\delta(\tilde{r}) + \delta(r)\tilde{r} \ . \end{aligned}$$

In unserem Falle einer k-Algebra R mit einem k–Algebrahomomorphismus σ verlangen wir zudem, daß δ k–linear ist. Damit folgt $\delta|_k \equiv 0$, man sagt k liegt im Konstantenkörper von δ.

Sei $R = k[X]$ und eine Operatorvariable D gegeben. Dann wird eine Ore–Algebra

$$\mathcal{O} := k[X] \langle D, \sigma, \delta \rangle$$

als der Schiefpolynomring definiert, in dem das Produkt von D mit $P \in k[X]$ von rechts durch

$$DP := \sigma(P)D + \delta(P) \ , P \in k[X]$$

festgelegt und dann assoziativ und distributiv auf Polynome $\sum_{i=1}^{r} P_i D^i, P_i \in k[X]$ fortgesetzt wird. Die Eigenschaften von δ sorgen dabei dafür, daß die entstehende Struktur vernünftig definiert ist ([GoWa], Prop. 1.10).

Klassischerweise entspricht D dem Differentialoperator (hier ist $\sigma = \mathrm{id}$ und $\delta = \frac{\partial}{\partial X}$), ein Element der zugehörigen Ore–Algebra kann dann als ein zusammengesetzter Differentialoperator interpretiert werden.

Aber etwa auch der Shiftoperator ($\sigma : X \mapsto X + 1, \delta = 0$) und eine ganze Reihe anderer Operatoren lassen sich in dieser Weise in einheitlicher Form darstellen.

Dieses Konzept kann nun auf verschiedene Weisen auf den Fall mehrerer Variablen und Operatoren verallgemeinert werden. Ziel ist hierbei unter anderem, Ausdrücke mit gemischten Operatoren, etwa Differential- und Shiftoperatoren einheitlichen Algorithmen zugänglich zu machen.

2.2 Der multivariate Fall

2.2.1 Iterierte Ore–Erweiterungen

Das Verfahren der iterierten Ore-Erweiterung wird ausführlich in [Pesch] beschrieben, hier wird es nur der Vollständigkeit halber und wenig formal eingeführt. Diese Methode, die Algebra rekursiv durch neue nichtkommutative Variablen zu erweitern, ist allgemeiner als die im Rest der Arbeit benutzte (etwas intuitivere) Ore-Erweiterung.

Hier wird vom Körper k ausgegangen. Die Ore-Erweiterung wird durch sukzessive Erweiterungen durch je eine neue Variable erreicht.

Satz 2.2 *Sei R eine (nicht notwendig kommutative) k–Algebra, $\sigma : R \longrightarrow R$ ein k-Algebrahomomorphismus, $\delta : R \longrightarrow R$ eine σ-Derivation.*

Dann kann man stets den Schiefpolynomring $R\langle W, \sigma, \delta\rangle$ in der adjungierten Variablen W bilden, der durch die Multiplikation $WP = \sigma(P)W + \delta(P)\ , P \in R$ festgelegt wird.

Siehe hierzu auch ([GoWa], Prop. 1.10).

Auf diese Weise können sukzessive "Schief-Variablen" $\langle W_i, \sigma_i, \delta_i\rangle$ an die bereits konstruierten Ringe R_i adjungiert werden, um $R\langle W_1, \sigma_1, \delta_1; \ldots; W_n, \sigma_n, \delta_n\rangle$ zu definieren.

Dieses Verfahren erweitert das Konzept in einer Variablen und einem Operator. Es ist auch eine Verallgemeinerung, nämlich mit

$$R_0 = k\ ,$$

$$\langle W_0 = X, \sigma_0 = \mathrm{id}, \delta_0 = 0\rangle$$

und

$$\langle W_1 = D, \sigma_1 = \sigma, \delta_1 = \delta\rangle$$

ist $k[X]\langle D, \sigma, \delta\rangle$ ein Spezialfall.

Allerdings ist hier kein Unterschied zwischen "Variablen" und "Operatoren" mehr sichtbar. Zudem "leben" die σ_i, δ_i auf ganz verschiedenen Ringen
$k\langle W_1, \sigma_1, \delta_1; \ldots; W_{i-1}, \sigma_{i-1}, \delta_{i-1}\rangle$.

Im Folgenden wird ein speziellerer Fall betrachtet, der eher der Vorstellung von Operatoren auf Funktionenräumen gerecht wird.

2.2.2 Ore–Algebren über Polynomringen

Diese Verallgemeinerung des univariaten Falles lehnt sich stärker an die Vorstellung von Operatoren auf Funktionen an, sie stellt einen Spezialfall der iterierten Ore–Algebra dar.

Die hier vorgestellte Definition wird so auch in [ChySal] verwendet.

Hier trennen wir kommutative Variablen $X_1, \ldots, X_m$ und "Operatorvariablen" $D_1, \ldots, D_n$, die untereinander auch wieder kommutieren.

Die σ_i, δ_i sind hier Abbildungen

$$\sigma_i, \delta_i : k[X_1, \ldots, X_m] \longrightarrow k[X_1, \ldots, X_m] \ ,$$

mit Algebrahomomorphismen σ_i und σ_i–Derivationen δ_i.

Unsere Forderungen an die Vertauschbarkeit der $D_1, \ldots, D_n$ impliziert weitere Bedingungen an die σ_i, δ_i in $k[X_1, \ldots, X_m] \langle D_i, \sigma_i, \delta_i \ , i = 1, \ldots, n \rangle$.

Etwa die Kommutativität der Operatoren wird dadurch gewährleistet, daß die δ_i untereinander kommutieren, die σ_i untereinander kommutieren und für $i \neq j$ jeweils δ_i und σ_j kommutieren. Dies macht aber schon die Ore–Algebra aus:

Definition 2.3 (Ore–Algebra) *Sei k ein Körper, $k[X_1, \ldots, X_m]$ der Polynomring in m Unbestimmten. Es seien für $i = 1, \ldots, n$ Algebrahomomorphismen und zugehörige Sigma–Derivationen $\sigma_i, \delta_i : k[X_1, \ldots, X_m] \longrightarrow k[X_1, \ldots, X_m]$ gegeben.*

Dann bezeichne $\mathcal{D} = k[X_1, \ldots, X_m] \langle D_i, \sigma_i, \delta_i \ , i = 1, \ldots, n \rangle$ den zugehörigen Schiefpolynomring über $k[X_1, \ldots, X_m]$.

$\mathcal{D}$ heißt eine **Ore–Algebra**, *wenn*

- *alle k-Algebrahomomorphismen σ_i untereinander kommutieren,*
- *Alle σ_i–Derivationen δ_i untereinander kommutieren,*
- *$\sigma_i \delta_j = \delta_j \sigma_i$ für $i, j = 1, \ldots, n \ , i \neq j$.*

Die Bedingungen an die σ_i, δ_i stellen sicher, daß in $\mathcal{D}$ für $P \in k[X_1, \ldots, X_m]$ gilt $D_i D_j P = D_j D_i P$, so daß man nun $D_i D_j$ und $D_j D_i$ identifizieren kann und die Multiplikation in $\mathcal{D}$ wohldefiniert bleibt.

Multivariate Verallgemeinerungen auf Algebren, in denen Relationen zwischen den Variablen auftauchen, untersuchten für den Fall der Algebren ordnungserhaltender Art Kandri-Rody und V. Weispfenning in [KRW].

Hier wird nicht zwischen zwei Arten von Variablen unterschieden, sondern von einer gegebenen nichtkommutativen Multiplikation ausgegangen, die mit der kommutativen Multiplikation auf der selben Menge in Relation gesetzt wird.

Später werden wir sehen, daß Ore–Algebren und ordnungserhaltende Algebren eine echte Schnittmenge haben.

2.3 G–Algebren

Viktor Levandovskyy betrachtet in seiner Arbeit [Levand] viele Anwendungen für diese Algebren und implementiert Algorithmen für diese Algebren in *Plural*. Da G–Algebren sogar eine Verallgemeinerung der Ore-Algebren darstellen, sollten sie hier auch kurz vorgestellt werden.

Eine G–Algebra A ist ein Quotient des freien nichtkommutativen Polynomringes $k\langle x_1, \ldots, x_n\rangle$ nach einem zweiseitigen Ideal. Zudem müssen die Restklassen der Monome $x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ eine k–Vektorraumbasis von A bilden (man sagt A hat eine Poincaré-Birkhoff-Witt Basis).

Die hier behandelten Ore–Algebren stellen einen Spezialfall von G–Algebren dar. Um dies zu zeigen sind die sogenannten “non-degeneracy-conditions” zu testen, in [Levand] werden diese auf die Assoziativität $x_i(x_j x_k) = (x_i x_j)x_k$ zurückgeführt. In Ore–Algebren sind aber die σ_i, δ_i gerade so gewählt, daß diese Assoziativität gilt.

Im folgenden Abschnitt wird beschrieben, wie die speziellen Eigenschaften der Ore–Algebren für konkrete Berechnungen genutzt werden können. Dies zeigt zugleich, daß es durchaus sinnvoll sein kann, sich mit den spezielleren Ore–Algebren zu befassen.

2.4 Ausnutzen der Restriktionen für das Rechnen in Ore–Algebren

Die Restriktionen führen dazu, daß mit einem σ die zugehörige σ–Derivation δ schon weitgehend festgelegt ist.

Dies läßt sich in der konkreten Berechnung ausnutzen, denn Algebrahomomorphismen lassen sich effizient durch Substitutionen der Erzeugenden berechnen. Es ist also vorteilhaft, die Derivationen mit Hilfe von Algebrahomomorphismen darzustellen.

$\sigma(X_1), \ldots, \sigma(X_m)$ und $\delta(X_1), \ldots, \delta(X_m)$ legen σ und δ bereits eindeutig fest. Bei σ, da es sich um einen k-Algebrahomomorphismus handelt, und bei δ, indem man zusammengesetzte Ausdrücke bei bekanntem σ mit den Derivationseigenschaften auflösen kann, bis δ nur noch mit einzelnen Variablen als Argument auftaucht. Wir werden sehen, daß oft sogar noch weniger Information reicht.

Die Rekonstruktion von σ aus den Werten auf den Variablen ist durch den Substitutionsmechanismus in *Mathematica* erreichbar — die Werte von σ werden in meiner Implementation bereits in Form solcher Regeln eingegeben.

Etwas komplizierter gestaltet sich der Fall bei der Rekonstruktion von δ bei bekanntem σ. Man möchte δ als Funktion auf dem Grundring darstellen und nicht mit jeder neuen Rechnung aufgrund der Regeln für σ-Derivationen den gegebenen Ausdruck umformen, bis er auf die bekannten Werte für δ zurückgeführt ist.

Im Falle einer Unbestimmten kann man hier die bekannte Tatsache ausnutzen, daß eine σ-Derivation δ auf $k[X]$ nur entweder

- die Nullabbildung ist (in doppelter Hinsicht) ein Spezialfall,
- bis auf einen Faktor $a \in k[X]$ die formale Ableitung nach der Unbestimmten ist, also $\delta = a\frac{\partial}{\partial X}$ (falls $\sigma = \mathrm{id}$), oder
- (für $\sigma \neq \mathrm{id}$) eine “innere σ-Derivation” ist,

  also für ein $a \in k[X]$ die Gestalt $\delta(p) = a\,(p - \sigma(p))$ hat.

Im Falle mehrerer Variablen gestaltet sich die Sache etwas komplizierter.

Wir werden nun eine Verallgemeinerung dieser Darstellung zeigen, die so nicht in der Literatur zu finden war.

Satz 2.4 *Sei* $R = k[X_1, \ldots, X_m]$ *,* $m > 1$*,*

$\sigma : R \longrightarrow R$ *ein* k*-Algebrahomomorphismus.*

Weiter sei $\sigma[X_i] - X_i \neq 0$ *für mindestens ein* i*.*

Dann ist δ *eine innere Derivation, und zwar die Derivation*

$$\delta_i : g \mapsto \frac{\delta(X_i)}{\sigma(X_i) - X_i} \left(\sigma(g) - g\right) \ .$$

Beweis: Wir zeigen dies durch Übereinstimmung von δ und δ_i auf allen Unbestimmten:

Aus $\delta(X_iX_j) = \delta(X_jX_i)$ folgt:

$\sigma(X_i)\delta(X_j) + \delta(X_i)X_j = \sigma(X_j)\delta(X_i) + \delta(X_j)X_i$, nach $\delta(X_j)$ aufgelöst:

$\delta(X_j) = \frac{\delta(X_i)}{\sigma(X_i)-X_i}(\sigma(X_j) - X_j)$, dies ist aber gerade $\delta_i(X_j)$. □

Ist aber $\sigma = \mathrm{id}$, so folgt

Satz 2.5 *Sei* $R = k[X_1, \ldots, X_m], m \in \mathbb{N}$ *und* $\sigma : R \longrightarrow R$ *die identische Abbildung.*

Dann ist jede σ*-Derivation* δ *eine Linearkombination der partiellen Ableitungen* $\partial X_i, i = 1, \ldots, m$ *über* R*, und zwar:* $\delta = \sum_{i=1}^{m} \delta(X_i)\frac{\partial}{\partial X_i}$*.*

Beweis: Wegen $\frac{\partial}{\partial X_i}(X_j) = 0, i \neq j$ und $\frac{\partial}{\partial X_i}(X_i) = 1 \ , i = 1, \ldots, m$ gilt

$$\left(\sum_{i=1}^{n} \delta(X_i)\frac{\partial}{\partial X_i}\right)(X_l) = \delta(X_l), l = 1, \ldots, m.$$

Als Linearkombination von σ-Derivationen ist $\sum_{i=1}^{m} \delta(X_i)\frac{\partial}{\partial X_i}$ eine σ-Derivation. Diese stimmt auf den Erzeugenden mit δ überein und ist daher gleich δ. □

Damit läßt sich auch im multivariaten Fall δ durch den Algebrahomomorphismus σ ausdrücken.

Neben dieser Klassifizierung liefern die inneren Derivationen $\delta_i = \mathrm{id} - \sigma_i$ zu untereinander kommutierenden Homomorphismen σ_i sofort ein Beispiel, in dem die zahlreichen Kommutativitätsbedingungen aus der Definition (2.3) einer Ore–Algebra automatisch erfüllt sind:

Satz 2.6 *Sei R eine k–Algebra, und seien k–Algebrahomomorphismen σ_1, σ_2 mit $\sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1$ sowie die inneren Derivationen $\delta_i := \sigma_i - \mathrm{id}_R, i = 1, 2$ gegeben.*

Dann gelten für $i, j \in \{1, 2\}$ mit $i \neq j$ die Vertauschungen $\sigma_i \circ \delta_j = \delta_j \circ \sigma_i$ und $\delta_i \circ \delta_j = \delta_j \circ \delta_i$.

Beweis: Dies rechnet man sofort nach:

$$\sigma_i \circ \delta_j = \sigma_i \circ (\sigma_j - \mathrm{id}_R) = \sigma_i \circ \sigma_j - \sigma_i = (\sigma_j - \mathrm{id}_R) \circ \sigma_i = \delta_j \sigma_i$$

und

$$\delta_1 \circ \delta_2 = (\sigma_1 - \mathrm{id}_R) \circ (\sigma_2 - \mathrm{id}_R) = \sigma_1 \circ \sigma_2 - \sigma_1 - \sigma_2 + \mathrm{id}_R$$

Letzteres ist wegen $\sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1$ symmetrisch in den Indizes. □

2.5 Ordnungen und Monoidstruktur

2.5.1 Begriffe

Zunächst eine knappe Einführung der verwendeten Begriffe:

Eine partielle Ordnung $\preceq$ auf einer nichtleeren Menge Γ ist eine reflexive, antisymmetrische, transitive Relation.

Für Elemente $\gamma_1, \gamma_2 \in \Gamma$ mit $\gamma_1 \preceq \gamma_2$ schreiben wir wie üblich auch $\gamma_2 \succeq \gamma_1$. Ist zudem $\gamma_1 \neq \gamma_2$ schreiben wir $\gamma_1 \prec \gamma_2$ beziehungsweise $\gamma_2 \succ \gamma_1$.

Man spricht von einer totalen Ordnung, wenn für beliebige $\gamma_{1,2}$ stets $\gamma_1 \prec \gamma_2$ oder $\gamma_1 = \gamma_2$ oder $\gamma_2 \prec \gamma_1$ gilt.

Ist Γ mit einer assoziativen inneren Verknüpfung $\circ$ versehen, so heißt Γ eine Halbgruppe. Gibt es zusätzlich ein neutrales Element $1 \in \Gamma$ und haben wir die Kürzungsregel $ax = ay \Rightarrow x = y$, so sprechen wir von einem *Monoid*. Ist auf Γ eine Ordnung definiert, nennen wir die Verknüpfung $\circ$ *monoton*, wenn aus $\gamma_1 \preceq \gamma_2$ für jedes $\gamma \in \Gamma$ auch $\gamma \circ \gamma_1 \preceq \gamma \circ \gamma_2$ folgt.

2.5.2 Noethersche Induktion

Es werden im folgenden einige bekannte Eigenschaften von Wohlordnungen gezeigt und schließlich das Prinzip der Noetherschen Induktion. Dieses taucht erst später wieder auf, passt aber hier thematisch besser. Es werden Grundlagen und übliche Notationen wiederholt.

Es sei Γ eine nicht leere Menge und $\preceq$ eine partielle Ordnung auf Γ.

Definition 2.7 (Absteigende Kettenbedingung) *Eine partielle Ordnung auf einer nichtleeren Menge Γ erfüllt die Absteigende Kettenbedingung, wenn es keine unendliche absteigende (wir sagen auch fallende) Kette*

$$\gamma_1 \succ \gamma_2 \succ \ldots \succ \gamma_n \succ \ldots$$

gibt.

Daraus folgt

Satz 2.8 *Eine partielle Ordnung auf einer Menge Γ erfüllt die Absteigende Kettenbedingung genau dann, wenn jede nichtleere Teilmenge von Γ ein minimales Element besitzt.*

Beweis:

$\Rightarrow$ (Indirekt) Haben wir eine nichtleere Menge ohne minimales Element, so finden wir zu jedem Element dieser Menge ein kleineres. So kann man (mit dem Auswahlaxiom) eine nicht abbrechende fallende Kette konstruieren.

$\Leftarrow$ Die Menge der Elemente einer absteigenden Kette hat ein minimales Element, bei dem sie folglich abbricht. □

Nun können wir das Prinzip der Noetherschen Induktion beweisen. Es bezeichne zu $\gamma \in \Gamma$ die Menge der kleineren Elemente

$$\Gamma|_{\prec\gamma:=}\{\psi \in \Gamma | \psi \prec \gamma\} \ .$$

Satz 2.9 (Noethersche Induktion) *Sei $\preceq$ eine partielle Ordnung auf Γ und sei die absteigende Kettenbedingung erfüllt. G sei eine Teilmenge von Γ. Wenn für jedes $\gamma \in \Gamma$ die Folgerung $\Gamma|_{\prec\gamma} \subset G \Longrightarrow \gamma \in G$ gilt, so ist $G = \Gamma$*

Beweis: Wäre G nicht gleich Γ, so hätte $\Gamma \backslash G$ ein minimales Element γ, die Minimalität impliziert $\Gamma|_{\prec\gamma} \subset G$, es folgt $\gamma \in G$, Widerspruch. □

2.6 Monome in Ore–Algebren

Zunächst ist zu definieren, was überhaupt unter einem Monom in einer Ore–Algebra zu verstehen ist.

Es scheint kaum explizite Thematisierungen dieser Frage in der Literatur zu geben. Das Thema ist dennoch eine Überlegung wert.

Da Kommutativität von Variablen mit Operatoren nicht gegeben ist, kann ein reines Produkt aus Variablen und Operatoren durch die entsprechenden Relationen (wenn δ nicht verschwindet) zu einer echten Summe werden. Etwa im Polynomring mit Differentialoperator $k\langle X, D\rangle$ sind $XD = DX + 1$ und $XD - 1 = DX$.

Definiert man ein Monom einfach als reines Produkt von Unbestimmten, so kann man einem Polynom nicht ansehen, ob es ein Monom ist.

Dieses Problem existiert ohne Vertauschungsrelationen nicht.

Dadurch wird die Untersuchung komplizierter als in dem von Mora untersuchten Fall, in dem die Monome einfach Elemente aus der freien, nichtkommutativen von den Unbestimmten erzeugten Gruppe sind, also endliche *Wörter* über dem Alphabet bestehend aus den Unbestimmten.

Betrachten wir nun die Menge W der Wörter in den Unbestimmten

$$\{X_1, \ldots, X_m, D_1, \ldots, D_n\}$$

einer Ore–Algebra $\mathcal{D} = k[X_1, \ldots, X_m] \langle D_i, \sigma_i, \delta_i \ , i = 1, \ldots, n \rangle$. Man kann ein Wort auch als Produkt in der Ore–Algebra auffassen. Dabei ist die Abbildung $W \to \mathcal{D}, w_1 \ldots w_r \mapsto \prod_{i=1}^{r} w_i$ im Allgemeinen weder injektiv noch surjektiv.

Wir sagen ein Wort $w_1 \ldots w_r$ hat einen "Fehlstand" bei $i, 1 \leq i < r$, wenn $w_i \in \{D_1, \ldots, D_n\}$ und $w_{i+1} \in \{X_1, \ldots, X_m\}$. Wenn ein Fehlstand im Wort w existiert, können wir i maximal wählen, also den letzten auftretenden Fehlstand betrachten.

Das Wort hat dann die Gestalt $w = w_1 \ldots w_{i-1} D_k X_l w_{i+2} \ldots w_r$.

Dabei folgt auf den letzten Fehlstand notwendig eine nicht leere Sequenz von Variablen, nach der höchstens Operatoren auftauchen. Ansonsten hätte man noch einen größeren Fehlstand im Widerspruch zur Maximalität von i.

Unser $w \in \mathcal{D}$ ist mit der Vertauschungsrelation $D_k X_l = \sigma_k(X_l) D_k + \delta_k(X_l)$ gleich

$$w = w_1 \ldots w_{i-1} \sigma_k(X_l) D_k w_{i+2} \ldots w_r + w_1 \ldots w_{i-1} \delta_k(X_l) w_{i+2} \ldots w_r.$$

Wir tun so, als wäre $\sigma(X_l)$ ein Monom, in Allgemeinen handelt es sich um ein Polynom, das in eine Linearkombination von Monomen aufzuspalten ist, auf die die folgenden Überlegungen anzuwenden sind. Gleiches gilt für δ.

Wir verlassen dabei den Bereich der Worte mit Buchstaben aus W und betrachten k–Linearkombination solcher Worte (frei-nichtkommutative Polynome in den Unbestimmten).

Im ersten Summanden ist entweder die Anzahl der Operatorvariablen nach dem letzten Fehlstand größer geworden oder (falls w_{i+2} eine Variable ist) gleich geblieben, aber die Anzahl der Variablen nach dem letzten Fehlstand ist vermindert. Spätestens, wenn diese Anzahl Null geworden ist und noch ein Fehlstand existiert, wird also die Anzahl der Operatorvariablen nach dem letzten Fehlstand größer. Dies kann nur endlich oft geschehen.

Im zweiten Summanden ist die Anzahl der Operatorvariablen um eins kleiner geworden. Dies kann offenbar nur endlich oft geschehen, da wir nur Wörter endlicher Länge haben.

Durch sukzessives Anwenden der Vertauschungsrelation auf alle Summanden werden daher schließlich alle Fehlstände aufgelöst. Dann ist für jedes auftauchende Monom eine Normalform ohne Fehlstände erreicht (im Grunde genommen haben wir das hier mit noetherscher Induktion gezeigt).

Zwischenzeitlich ist dabei allerdings ein exponentielles Anschwellen der Anzahl der Summanden möglich.

Auf diese Weise können wir nun Monome einer Ore–Algebra definieren als Elemente von $\mathcal{D}$, die, in Normalform gebracht, im herkömmlichen Sinne Monome sind:

Definition 2.10 (Monom) *Ein Element p einer Ore–Algebra*

$k[X_1, \ldots, X_m] \langle D_i, \sigma_i, \delta_i \, , i = 1, \ldots, n \rangle$ *heißt* **Monom**, *wenn es eine Darstellung*

$$p = \prod_{i=1}^{m} X_i^{\alpha_i} \prod_{j=1}^{n} D_j^{\beta_j} \, , \alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n \in \mathbb{N}_0$$

hat.

Bemerkung 2.11 *Zu einer Ore–Algebra $\mathcal{O}$ bezeichne $\mathrm{M}(\mathcal{O})$ die Menge der Monome in $\mathcal{O}$.*

Die Frage, ob ein Element einer Ore–Algebra ein Monom ist, ist berechenbar, da wir auch die Normalform berechnen können.

Bemerkung 2.12 *Die Monom-Eigenschaft wie hier definiert ist unabhängig von der Ordnung!*

Dabei ist a priori durchaus eine Normalform der Darstellung denkbar, die von der jeweiligen Ordnung abhängig ist, indem die Potenzen der Unbestimmten nach der gegebenen Monomdnung sortiert sein sollen. Beispielsweise wäre in der Differentialalgebra $k[X]\langle D\rangle$ das Polynom DX^3 für die lexikografische Ordnung mit $D > X$ in Normalform, ist aber D lexikografisch kleiner als X, sollte die Form $X^3D + 3X^2$ "normal" sein. In dieser "ordnungstreuen Normalform" wäre also DX^3 kein Monom mehr.

Das später folgende Beispiel 2.13 zeigt eine Ordnung, in der die Ungleichungskette $X^2 > D > X$ gilt, wodurch in beiden Darstellungen $X^3D = X^2DX + 2X^2$ die Potenzen sortiert sind: man könnte ohne weitergehende Definitionen nicht wirklich von einer Normalform reden.

Ein noch unangenehmeres Beispiel ist $k[X]\langle Q, \sigma : X \mapsto X^2, \delta \equiv 0\rangle$, mit $X^2 > Q > X$. Hier kommt man auch, wenn man das Sammeln von Potenzen vorschreibt, wegen $QX = X^2Q$ zu einer Mehrdeutigkeit. In diesem Beispiel ist σ nicht surjektiv — damit können wir zum Beispiel das Element XQ auf keinen Fall in die Form $QP(X)$ bringen mit $P(X) \in k[X]$.

2.7 Monomordnungen

Auf der Menge der Monome kann man auf verschiedene Weisen Ordnungen definieren. Im kommutativen Fall verlangt man natürlich Verträglichkeit der Ordnung mit der Multiplikation im Sinne der Monotonie und, daß das Einselement minimal ist. Für Monome m_1, m_2, m besagt die Monotonie, daß $m_1 < m_2 \Rightarrow mm_1 < mm_2$. Sind diese Bedingungen erfüllt, so ist $<$ eine zulässige Monomordnung.

2.7.1 Zulässige Monomordnungen im kommutativen Fall

Im kommutativen Fall $k[X_1, \ldots, X_m]$ können wir Monome X^ι mit den Multiindizes im Exponenten, also Elementen aus $\mathbb{N}_0{}^m$, identifizieren. Das Produkt zweier Monome ist stets wieder ein Monom, und wir haben die Zuordnung:

$$\log : \mathrm{M}(k[X_1, \ldots, X_m]) \to \mathbb{N}_0{}^m, X^i \mapsto i, i \in \mathbb{N}_0 \ .$$

Diese Abbildung ist bijektiv und ein Homomorphismus von Monoiden (bei dem Multiplikation in Addition übergeht).

In diesem Fall werden zulässige Ordnungen stets durch den Positivbereich gegeben, der durch Auswertung einer geeigneten reellen Linearform definiert wird. Dies zeigte V. Weispfenning in [WPF] 1987.

Wichtiger für die Anwendung der Computeralgebra dürfte die Tatsache sein, daß hier jede zulässige Ordnung auf $\mathbb{N}_0^m$ durch Transformation mittels einer reellen $m \times m$–Matrix vom Rang m in die lexikografische Ordnung auf $\mathbb{R}^m$ überführt werden kann, siehe [AmGlKü], p. 182. Wir werden gleich einige Beispiele für Matrixordnungen sehen.

Es ergibt sich, daß jede Monomordnung eine Verfeinerung einer durch eine Linearform auf $\mathbb{N}_0^m$ gegebenen Halbordnung der Gestalt

$$X <_\omega Y :\Leftrightarrow \langle \log X, \omega \rangle < \langle \log Y, \omega \rangle \ \ , \omega \in \mathbb{R}^m$$

ist (dabei sei $\langle \cdot, \cdot \rangle$ das kanonische Skalarprodukt).

Die Linearformen entsprechen Punkten $\omega \in \mathbb{R}^m$. So wird es möglich, Klassen von Ordnungen mit linearer Algebra zu untersuchen. Ein Beispiel ist der sogenannte Gröbner-Fan eines Ideales $\mathcal{I}$, eine Klasseneinteilung des $\mathbb{R}^m$. Bezüglich einer Ordnung hat $\mathcal{I}$ eine reduzierte Gröbnerbasis G. Die Klasse dieser Ordnung ist dann die Menge der Ordnungen, bezüglich derer $\mathcal{I}$ dieselbe reduzierte Gröbnerbasis G hat (siehe hierzu [MoRo]).

Leider scheitert eine einfache Übertragung der Erkenntnisse aus dem Kommutativen bereits an der Abgeschlossenheit der Menge der Monome bezüglich der Multiplikation.

2.8 Matrixordnungen

Auf reellen n-Tupeln $(a_1, \ldots, a_n)$ können wir die lexikographische Ordnung definieren durch

$$\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} <_{\text{lex}} \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \Leftrightarrow \exists m \mid a_j = b_j, j = 1, \ldots, m-1 \ , a_m < b_m$$

Haben wir eine reguläre Matrix $M \in \mathbb{R}^{n\times n}$ gegeben, so erhalten wir eine neue Ordnung $<_M$ auf $\mathbb{R}^n$ vermöge

$$\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} <_M \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \Leftrightarrow M \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} <_{\text{lex}} M \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} .$$

Diese Ordnung heißt dann die Matrixordnung auf $\mathbb{R}^n$ bezüglich M.

Die Matrix M kann a priori eine beliebige $r \times n$-Matrix vom Rang n sein. Um eine totale Ordnung zu erzeugen muß die Matrixmultiplikation injektiv sein, es folgt also $r \geq n$.

Man kann sukzessive aus M jeweils die letzte von den vorherigen linear abhängige Zeile streichen, da die korrespondierende Komponente im Vektor $M \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$ offenbar keine entscheidende Rolle spielen kann.

So erhält man stets eine reguläre quadratische Matrix, die dieselbe Matrixordnung definiert.

Offenbar definiert dieselbe Matrix M auf $\mathbb{Q}^n$ und $\mathbb{Z}^n$ die entsprechenden Einschränkungen von $<_M$.

Im generischen Fall, wenn die Einträge der ersten Zeile algebraisch unabhängig über $\mathbb{Q}$ sind, legt die erste Zeile von M die Ordnung auf $\mathbb{Q}^n$ schon vollständig fest. Dann definiert schon die durch die erste Zeile gegebene Liniearform die eingeschränkten Ordnungen vollständig.

Das bedeutet aber für unsere Zwecke keine Entwarnung. Die folgenden Beispiele zeigen, daß die real vorkommenden Matrizen praktisch immer über $\mathbb{Q}$ oder gar $\mathbb{Z}$ definiert sind. Erspart bleiben uns dadurch die Probleme mit der Darstellung von reellen Zahlen.

Für den kommutativen Fall wird in [Robb] gezeigt, daß jede Monomordnung durch eine reelle Matrix darstellbar ist. Dort wird eine vollständige Klassifizierung der Ordnungen gegeben. Das Resultat ist, daß die darstellende Matrix M einer Ordnung in der Regel weniger als n Zeilen haben muß. In diesem Falle kann man die nötigen Zeilen durch beliebige Zeilen zu einer quadratischen Matrix ergänzen. Ordnungen, für die M regulär sein muß, heißen bei [Robb] von

lexikografischem Typ.

Alle folgenden Beispiele sind von diesem Typ. Aufgrund des Problems transzendente Zahlen im Computer zu realisieren dürften die anderen Ordnungen auch schwerer zu behandeln sein.

Um für Monome mit n Unbestimmten eine Matrixordnung zu definieren muß eine Reihenfolge der Variablen festgelegt sein. Ändert man diese Anordnung, so ist klar, daß die Monomordnung eine Matrixordnung bleibt. Die zugehörige Matrix multipliziert sich mit einer entsprechenden Permutationsmatrix[1].

Ist die Reihenfolge der Unbestimmen klar, so nennen wir die einer Matrix M zugeordnete Monomordnung $\mathrm{Ord}\,(M)$.

2.8.1 Beispiele für Matrixordnungen

Zunächst eine Matrixordnung auf den Monomen von $\mathbb{Q}[X_1, X_2]$, in der $X_1 < X_2 < X_1^2$ gilt:

Beispiel 2.13 *In der Matrixordnung zu* $M = \begin{pmatrix} 2 & 3 \\ 0 & 1 \end{pmatrix}$ *auf* $\mathbb{Q}[X_1, X_2]$ *gilt die Ungleichungskette* $X_1 < X_2 < X_1^2$.

Man rechnet sofort nach:

$$M\log(X_1) = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, M\log(X_2) = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, M\log(X_1^2) = \begin{pmatrix} 4 \\ 0 \end{pmatrix}.$$

Nach Definition der Matrixordnung gilt die behauptete Ungleichungskette. □

Als Beispiele gebe ich die zugehörigen Matrizen für einige klassische Monomordnungen auf $k[X_1, \ldots, X_n]$ an, und zwar für die lexikografische, die gradlexikografische und die inverse gradlexikografische Monomordnung:

lex: Hier ist trivialerweise M die Einheitsmatrix.

glex: Hier entscheidet als erstes die Summe der Exponenten:

[1] Auf die Weise sind die Ordnungen im Grundpakete zum Rechnen mit Ore–Algebren intern realisiert.

$$M = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$

invglex: Hier entscheidet zunächst wieder die Summe der Exponenten.

Ist diese Summe gleich, so gewinnt der Term, bei dem der erste verschiedene Exponent *von rechts her* (daher invers) *kleiner* ist:

$$M = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & \cdots & 0 & 0 & -1 \\ 0 & \cdots & 0 & -1 & 0 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ 0 & -1 & 0 & \dots & 0 \end{pmatrix}$$

Dabei könnte man die negativen Einträge beseitigen, indem man die erste Zeile auf alle folgenden aufaddiert.

2.8.2 Zulässige Monomordnungen für Ore–Algebren

Gegeben sei die Ore–Algebra $\mathcal{D} = k[X_1, \dots, X_m] \langle D_i, \sigma_i, \delta_i \, , i = 1, \dots, n \rangle$ mit $\delta = (\delta_1, \dots, \delta_n)$ und $\sigma = (\sigma_1, \dots, \sigma_n)$.

Auf der Menge der Monome in $\mathcal{D}$ sei eine Ordnungsrelation gegeben.

Ein Produkt von Monomen muß kein Monom mehr sein. Um dennoch eine Monotonieaussage zu formulieren, werden in diesem Fall nur die Leitmonome verglichen: $p <_l q$ steht also abkürzend für $\mathrm{LM}\,(p) \prec \mathrm{LM}\,(q)$.

Definition 2.14 (Zulässige Monomordnung) *Eine Monomordnung auf der Ore–Algebra $\mathcal{D}$ heiße zulässig , wenn für $P, Q \in \mathrm{M}(\mathcal{D})\,; R, R' \in \mathcal{D}$ gilt:*

$$P \prec Q \Rightarrow RPR' <_l RQR'; R, R' \in \mathcal{D} \setminus \{0\} \; .$$

Die Zulässigkeit braucht man nur für Monome zu prüfen:

Lemma 2.15 *Eine Ordnung ist zulässig, wenn für alle $P, Q \in \mathrm{M}(\mathcal{D})\,, P \prec Q$, und alle $R, R' \in \mathrm{M}(\mathcal{D})$ folgt $RPR' <_l RQR'$.*

Beweis: Zu folgern ist aus der Prämisse im Lemma:

sind $P, Q \in \mathrm{M}(\mathcal{D}), P \prec Q$ und $S, S' \in \mathcal{D}$, so gilt $\mathrm{LM}(SPS') \prec \mathrm{LM}(SQS')$.

Zunächst gilt für $T, U \in \mathcal{D}$: $\mathrm{LM}(TU) = \mathrm{LM}(\mathrm{LM}(T) \cdot \mathrm{LM}(U))$. Denn für beliebige Terme T_i aus T und U_j aus U gilt mit unserer Prämisse:

$$\mathrm{LM}(T_i U_j) \preceq \mathrm{LM}(\mathrm{LM}(T)\, U_j) \preceq \mathrm{LM}(\mathrm{LM}(T)\, \mathrm{LM}(U)). \tag{2}$$

Folglich kann $\mathrm{LM}(TU)$ höchstens kleiner sein als $\mathrm{LM}(\mathrm{LM}(T)\, \mathrm{LM}(U))$, wenn sich das Monom $\mathrm{LM}(\mathrm{LM}(T)\, \mathrm{LM}(U))$ weghebt. Das kann aber nicht geschehen, denn wir haben eine Wohlordnung auf den Termen, so daß bei allen Paaren T_i, U_j in (2) die nicht mit $\mathrm{LM}(T), \mathrm{LM}(U)$ übereinstimmen nach unserer Prämisse an einer Stelle die echte Ungleichung steht! Es folgt dann für $T, U, V \in \mathcal{D}$:

$$\mathrm{LM}(TUV) = \mathrm{LM}(\mathrm{LM}(T)\, \mathrm{LM}(U)\, \mathrm{LM}(V)) \ ,$$

Damit ist die Situation auf den Monomfall zurückgeführt! □

2.8.3 Restriktionen für zulässige Ordnungen

Aus den Bedingungen an eine zulässige Ordnung ergeben sich Konsequenzen für σ und δ.

Hat man etwa als Koeffizientenring $k[X]$, einen kommutativen Operator D_1 und einen Operator D_2 mit der Vertauschungsregel $D_2 X = X^2 D_2$, (also $\sigma(X) = X^2, \delta(X) = 0$), so kann die Monomordnung nicht lexikographisch mit $X > D_1 > D_2$ sein.

Denn aufgrund der lexikografischen Ordnung wäre dann $X^2 D_2 > X D_1$. Mit den Vertauschungsregeln folgte $D_2 X > D_1 X$. Zugleich ist aber $D_2 < D_1$. Ordnung und Multiplikation sind in diesem Falle also nicht verträglich.

In [KRW] wird dieser Fall schlicht durch die zusätzliche Forderung an das Produkt von zwei Unbestimmten W_i, W_j aus dem Schiefpolynomring $k\langle W_1, \ldots, W_n \rangle$

$j > i : W_j W_i = \underbrace{c_{ij}}_{\in k^*} W_i W_j +$ kleinere Terme

verhindert, diese Eigenschaft heißt dort “of solvable type”. In (2.9) werden diese Algebren unter der Bezeichnung “ordnungserhaltender Art” vorgestellt. Algebren

ordnungserhaltender Art erfassen also nicht alle möglichen Ore–Algebren mit zulässigen Ordnungen.

Für die lexikographische Monomordnung mit $D_1 > D_2 > X$ ergeben sich im obigen Beispiel allerdings keine Probleme und wir haben eine bezüglich $<$ zulässige Algebra.

Ein weiteres, noch einfacheres Beispiel für eine Ore–Algebra, die für viele Ordnungen nicht zulässig ist:

$$k[X]\left\langle Q, \sigma : X \mapsto X^2, \delta = 0 \right\rangle,$$

Hier folgt für eine gegebene Monomordnung ganz allgemein für $n \in \mathbb{N}$:

$$\begin{array}{rccc} & 1 & < & Q \\ \Longrightarrow & X^{2n} \cdot 1 & < & X^{2n} \cdot Q \\ \Longrightarrow & X^n \cdot X^n & < & Q \cdot X^n \\ \Longrightarrow & X^n & < & Q, \end{array}$$

Es bleibt hier also nur die lexikografische Ordnung mit $X < Q$ als zulässige Monomordnung.

Im *Mathematica* Paket zu Ore–Algebren habe ich als voreingestellte Ordnung die lexikografische gewählt, wobei die Operatoren alle Variablen dominieren.

Man könnte meinen, daß (alle $\sigma_i \neq 0$ vorausgesetzt), sich eine Relation $p < q$ durch Linksmultiplizieren mit D_i nicht verändert, da sich (nachdem man die Ausdrücke in kanonische Form gebracht hat, an der (die Ordnung dominierenden) Operatorkonstellation nichts ändert, und bei gleicher Operatorkonstellation sowohl links als auch rechts des Ungleichungszeichens die Operatoren auf die gleiche Weise wirken. Letzteres ist leider ein Trugschluß.

Es kann σ_i auf mehrere Variablen zugleich wirken, wodurch ein Operator sehr wohl eine Ungleichung zwischen Monomen in Variablen ändern kann. Ein geradezu universelles Gegenbeispiel ist:

$$k[X, Y]\left\langle S, (\sigma : X \mapsto Y, Y \mapsto X); \delta = 0 \right\rangle,$$

auf dem es gar keine zulässige Ordnung geben kann wegen

$$\begin{aligned} X &< Y \\ \Longleftrightarrow S \cdot X &< S \cdot Y \\ \Longleftrightarrow Y \cdot S &< X \cdot S \\ \Longleftrightarrow Y &< X. \end{aligned}$$

Es gibt noch eine Fülle weiterer Ordnungen, die oft der lexikografischen überlegen sind. Es wäre schön, wenn sich die Überprüfung auf Zulässigkeit automatisieren ließe. Dies führt zu zwei Fragestellungen:

Sind δ,σ gegeben, welche Kriterien muß eine Ordnung erfüllen, damit sie zu der Ore–Algebra paßt, und welche Kriterien stellt umgekehrt eine gegebene Monomordnung an die σ_i und δ_i?

2.8.4 Ein Kriterium an die Monomordnung

Folgendes Kriterium läßt sich relativ leicht überprüfen:

$\mathcal{D} = k[X_1, \ldots, X_m] \langle D_i, \sigma_i, \delta_i \, , i = 1, \ldots, n \rangle$ und eine Relation "$<$" auf $\mathcal{D}$ seien gegeben.

Für eine einheitliche Schreibweise von Monomen sei $r = m + n$ und wir setzen $W_1 = X_1, \ldots, W_m = X_m, W_{m+1} = D_1, \ldots, W_r = D_n$. Hiermit hat nun jedes Monom w eine eindeutige Darstellung $w = \prod_{i=1}^{r} W_i^{n_i}$. Die Relation $<$ nennen wir *monoton in den Exponenten*, wenn es das Potenzieren mit Multiindizes $\mathbb{N}_0^{\,r} \ni \eta \mapsto W^\eta$ mit der kanonischen partiellen Ordnung ist:

$$\eta < \mu \Rightarrow W^\eta < W^\mu \, .$$

Eine Ordnung auf der Basis heiße zulässig, wenn für alle erzeugenden Monome $W_{i,j}, w \in \{X_1, \ldots, X_m, D_1, \ldots, D_n\}$ gilt:

$$\begin{aligned} W_i < W_j &\Leftrightarrow wW_i <_l wW_j \\ W_i < W_j &\Leftrightarrow W_i w <_l W_j w. \end{aligned}$$

Bemerkung 2.16 *Aus der Zulässigkeit einer Ordnung $<$ folgt die Monotonie in den Exponenten und Zulässigkeit auf der Basis.*

Beweis: Zulässigkeit auf der Basis ist trivial. Die Monotonie in den Exponenten ergibt sich, da Linksmultiplizieren mit einer Variablen oder Rechtsmultiplizieren mit einer Operatorvariablen ein Monom vergrößern und ein W^β durch solche Multiplikationen aus W^α mit $\alpha < \beta \in \mathbb{N}_0^r$ hervorgeht. □

Interessanter wäre die Umkehrung, daß eine auf der Basis zulässige Monomordnung schon zulässig ist. Dies ist leider falsch. Ein Gegenbeispiel liefert wieder die Algebra:

$$\mathcal{Q} = k[X]\left\langle Q, \sigma : X \mapsto X^2, \delta \equiv 0 \right\rangle$$

mit $QX = X^2Q$, die wir schon kennen.

Betrachten wir eine Ordnung, in der X das Gewicht 1 und Q das Gewicht 3 hat, etwa die Matrixordnung $\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}$.

Wegen $X \prec Q$ $(1 < 3)$ und $QX = X^2Q \prec Q^2$ $(2 + 3 < 2 * 3)$ sowie $X^2 \prec XQ$ $(2 < 1 + 3)$ ist die Ordnung auf der Basis zulässig.

Trotzdem ist diese Ordnung nicht zulässig, denn es gilt $X^2 \prec Q$ $(2 < 3)$ aber $QX^2 = X^4Q \not\prec Q^2$ $(7 > 6)$.

Es ist also im Allgemeinen nicht so leicht, die Zulässigkeit einer Monomordnung festzustellen.

2.9 Algebren ordnungserhaltender Art

Algebren ordnungserhaltender Art werden in ([KRW]) wie folgt eingeführt:

Gegeben sei zu einem Körper k der kommutative Polynomring $R = k[z_1, \ldots z_r]$ mit der herkömmlichen, kommutativen Multiplikation "$\cdot$".

Die Menge R sei eine nicht notwendig kommutative k–Algebra vermöge einer weiteren Multiplikation "$*$".

Die Monome aus R seien mit einer Monomordnung $\prec$ versehen, die bezüglich "$*$" zulässig ist.

Für alle $f, g \in R$ gebe es ein $\alpha \in k^*$ mit $f * g - \alpha f \cdot g < f \cdot g$.

Anschaulich entstehen also bei der Multiplikation "$*$" (im Vergleich zur kommu-

tativen) nur “Störterme” niedrigerer Ordnung.

Dies wird in ([KRW]) auf wenige Axiome zurückgeführt:

Definition 2.17 (Algebra ordnungserhaltender Art) *Sei k ein Körper, $R = k[z_1, \dots z_r]$, “$<$” eine zulässige Monomordnung und $* : R \times R \to R$ eine binäre Operation auf R. Dann heißt $(R, *)$ Algebra ordnungserhaltender Art, wenn folgende Axiome erfüllt sind:*

1. $(R, 0, 1, +, -, *)$ *ist ein assoziativer Ring mit 1.*
2. *für alle $a, b \in k, 1 \leq h \leq i \leq j \leq k \leq r, t \in \mathrm{T}(k[z_1, \dots, z_j])$ gelten:*
   - *(a)* $a * bt = (bt) * a = abt$,
   - *(b)* $z_h * (bt) = bz_h t$,
   - *(c)* $bt * z_k = btz_k$,
3. *Für alle $1 \leq i \leq j \leq r$ gibt es ein $c_{ij} \in k^*$ und ein $p_{ij} \in R$ mit $z_j * z_i = c_{ij} z_i z_j + p_{ij}, \mathrm{LT}(p_{ij}) < z_i z_j$.*

Satz 2.18 *Sei $R = (k[z_1, \dots, z_r], *, \prec)$ ein Ring ordnungserhaltender Art und $f, g \in R$. Dann gelten*

1. $\mathrm{LM}(f * g) = \mathrm{LM}(f)\,\mathrm{LM}(g)$
2. $\mathrm{LM}(f * g) = \mathrm{LM}(g * f) = \mathrm{LM}(fg)$
3. *für $h \in R$ und $\mathrm{LM}(f) \prec \mathrm{LM}(g)$ folgen $\mathrm{LM}(f * h) \prec \mathrm{LM}(g * h)$ und $\mathrm{LM}(h * f) \prec \mathrm{LM}(h * g)$*

Beweis: [KRW], p 4, Lemma 1.5 □

Wegen Eigenschaft 3 wurde für “of solvable type” die Übersetzung “ordnungserhaltender Art” gewählt.

2.9.1 Beispiele für Ore–Algebren ordnungserhaltender Art

Wenden wir uns wieder dem Setting einer Ore–Algebra zu.

Sei also $\mathcal{D} = k[X_1, \ldots, X_m] \langle D_i, \sigma_i, \delta_i \, , i = 1, \ldots, n \rangle$ mit $\delta = (\delta_1, \ldots, \delta_n)$ und $\sigma = (\sigma_1, \ldots, \sigma_n)$ gegeben.

Mit den Setzungen $z_i := X_i, i = 1, \ldots, m$ und $z_m + j := D_j$ in (2.17) sind die ersten beiden Axiome automatisch erfüllt.

Die dritte Bedingung ergibt weitere Restriktionen für die $\sigma_i, \delta_i, i = 1, \ldots, n$.

Dennoch gibt es noch reichlich Ore–Algebren ordnungserhaltender Art.

Etwa die Weyl–Algebra mit $\sigma_i = \mathrm{id}$ und $\delta_i(P) := \frac{\partial}{\partial_i} P$ erfüllt bezüglich jeder Monomordnung, in der die D_i dominieren, die gegebenen Bedingungen.

Auch Mischformen sind hier möglich, zum Beispiel aus Differential- und Shiftoperator:

$$\mathcal{D} = k[X, n] \left\langle D, \mathrm{id}, \frac{\partial}{\partial X}; S : n \mapsto n + 1, \delta = 0 \right\rangle$$

Solche Algebren tauchen zum Beispiel auf, wenn man eine diskrete Variable n shiftet und nach einer kontinuierlichen Variablen differenziert.

Es sind auch bizarrere Beispiele denkbar, für die man sich wahrscheinlich erst eine Anwendung ausdenken müßte:

$$\mathcal{D} = k[X_1, X_2] \langle D_1; D_2 \rangle \, , X_1 \prec X_2 \prec D_1 \prec D_2$$

mit lexikografischer Monomordnung und folgenden σ, δ

$$\sigma_1 \quad : \quad \left\{ \begin{array}{c} X_1 \mapsto X_1 + X_2^2 \\ X_2 \mapsto X_2 \end{array} \right. \, , \delta_1(P) = \sigma_1(P) - P$$

$$\sigma_2 \quad : \quad \left\{ \begin{array}{c} X_1 \mapsto X_1 - X_2 \\ X_2 \mapsto X_2 \end{array} \right. \, , \delta_2(P) = \sigma_2(P) - P$$

Die σ_i sind Substitutionshomomorphismen, man rechnet sofort nach, daß die σ_i kommutieren:

$$\begin{array}{rcl} \sigma_1 \circ \sigma_2(X_1) = \sigma_2 \circ \sigma_1(X_1) & = & X_1 + X_2^2 - X_2 \\ \sigma_1 \circ \sigma_2(X_2) = \sigma_2 \circ \sigma_1(X_2) & = & X_2 \, , \end{array}$$

Nach (2.6) erhalten wir mit den inneren Derivationen $\delta_i = \sigma_i - \mathrm{id}$ ein Beispiel für eine Ore–Algebra, obendrein gelten

$$\begin{aligned}
D_1X_1 &= X_1D_1 + X_2^2D_1 + X_1 + X_2^2 - X_1 = X_1D_1 + \overbrace{X_2^2D_1 + X_2^2}^{\prec_* X_1D_1} \\
D_1X_2 &= X_2D_1 \\
D_2X_1 &= X_1D_2 - X_2D_2 + X_1 - X_2 - X_1 = X_1D_2 + \overbrace{(-X_2D_2 - X_2)}^{\prec_* X_1D_2} \\
D_2X_2 &= X_2D_2
\end{aligned}$$

womit wir ein Beispiel einer Algebra ordnungserhaltender Art gefunden haben. Offensichtlich ist hier die Ordnung wesentlich!

3 Die Implementationen in *Mathematica*

Ein großer Teil dieser Arbeit besteht aus einer Implementation der Grundoperationen in Ore–Algebren in *Mathematica.*

Ziel war es, ein Grundpaket mit den grundlegenden Operationen Addition, Multiplikation, Zugriffen auf Strukturen etc. und ein erweitertes Paket mit Algorithmen zu realisieren.

Im folgenden werde ich die Struktur der Pakete vorstellen. Wer wirklich jede Feinheit sehen will, sei auf den Anhang verwiesen. Dort findet sich auch eine Auflistung aller Befehle.

Die folgenden Abschnitte stellen also einen Kompromiß aus detaillierter Auflistung und allgemeiner gehaltener Beschreibung dar.

Beim Arbeiten mit den implementierten Paketen wird im Laufe einer Sitzung die Struktur der Algebra aufgebaut. Es können auch im Verlauf zusätzliche Operatoren definiert und die gegebene Termordnung verändert werden.

Das Vorgehen ist dabei recht intuitiv. Typischerweise werden nach dem Laden des Paketes einige Operatorobjekte erzeugt, zugewiesen und dabei die zugehörigen Variablen erklärt. Anschließend werden mit den Grundoperationen Ausdrücke der Ore–Algebra definiert und Variablen zugewiesen, mit denen gerechnet wird.

Für das Grundpaket ist die Eigenschaft ordnungserhaltend zu sein unwichtig, sie ist für das einfache Rechnen nicht relevant. Für die erweiterten Algorithmen wie etwa das Berechnen von Gröbnerbasen ist der Erfolg nur für Algebren ordnungserhaltender Art garantiert.

Es wurde bei der Implementation mehr auf ausführlich dokumentierten Quelltext als auf Geschwindigkeit geachtet, da es für die Grundoperationen bereits andere Pakete gibt und *Mathematica* sowieso nicht für schnelle Programmausführung bekannt ist. Der Code ist daher eher als Labor zur Erstellung neuer Algorithmen gedacht.

So können die Pakete von jedem Interessierten variiert werden und als erweiterbares Werkzeug für Entwicklungen genutzt werden. Denkbar ist unter vielem Anderen Zähler für alle bisher durchgeführten Multiplikationen einzufügen und so ein hardwareunabhängiges Komplexitätsmaß zu erhalten.

3.1 Strukturen

Bei der Implementation wurde die Philosophie von *Mathematica* übernommen: Funktionsnamen werden in der Regel groß- und ausgeschrieben.

Die Multiplikation der Ore–Algebra heißt "`OreMultiplication`". Da die Addition in Ore–Algebren kommutiert, spricht nichts gegen die Verwendung von "+" für Ore–Objekte.

Es erwiesen sich einige Eingabehilfen wie "`OMu`" für iteriertes Ausführen des Befehles "`OreMultiplication`" als hilfreich.

Für eine lesbare Ausgabe werden die Objekte des Paketes nicht in der Form ausgegeben, in der sie intern dargestellt sind. Dies kann man mit `FullForm` erzwingen. Die Ausgabeform ist gewiß noch zu verbessern, reicht aber, um Ausdrücke zu überblicken.

Damit sind wir schon bei den grundlegenden Objekten angelangt, im Folgenden wird die Implementation in ihrem Aufbau erklärt.

3.2 Datenstrukturen der Implementierung

Da das Paket speziell auf Ore–Algebren ausgelegt ist, spiegeln sich die Strukturen der Ore–Algebren in der Datenstruktur der Implementation wieder. Diese Datenstruktur wird nun vorgestellt, der Anwender kommt dabei praktisch nur mit Ore–Polynomen in Kontakt.

Intern werden Listen für die definierten Operatoren gehalten. In den Listen werden Ausgabesymbole, die zugehörigen σ- und δ-Funktionen, sowie (optional) eine Anwendungsoperation hinterlegt. Die interne Position eines Operators in der Liste ändert sich nach dem Anlegen nicht mehr.

Ein Operatormonom $D_{r_1}^{n_1} \cdot \ldots \cdot D_{r_k}^{n_k}$ ist dabei als Liste

`OreMonomial[{`r_1`,`n_1`},..., {`r_k`,`n_k`}]`

dargestellt.

Ebenso gibt es eine Liste der Variablennamen.

Ein Summand ist dann ein Paar aus einem Monom in den Variablen und ei-

nem Operatormonom, dabei wird das Variablenmonom als normaler *Mathematica*-Ausdruck dargestellt. Insbesondere können hier auch Konstanten und Parameter auftauchen, also Variablen, die nicht in der Liste der Ore-Variablen auftauchen:

`OreSummand[<Expr>, OreMonomial[...]]`

Vorteil dieser Darstellung ist, daß Ausdrücke in *Mathematica* ohne weiteren Implementationsaufwand korrekt behandelt werden, und es ist `<Expr>` ein symbolisch kommutativ zu bearbeitendes Objekt, auf das die σ, δ wirken können, indem die in *Mathematica* implementierten Substitutionsmechanismen angewandt werden.

Der Nachteil dieser uneinheitlichen Mischung aus Ausdrücken und Listen ist allerdings, daß beim Arbeiten mit der Ordnung ein Kontextwechsel erforderlich ist. Für Vergleiche "<" muß man beispielsweise oft den Vektor der Exponenten für ein Monom der Gestalt `OreSummand[<Expr>, OreMonomial[...]]` finden ($\log$–Abbildung). Dabei bleibt man für den Operatoranteil `OreMonomial[...]` in der Welt der Listen, während man die `<Expr>` analysieren und dann die Liste der Exponenten aufbauen muß.

Die eigentlichen Polynome setzen sich schließlich aus "Summanden" zusammen, jeder Summand ist dabei ein Produkt eines herkömmlichen, kommutativen Ausdruckes und eines Operatormonoms:

`OrePolynomial[_OreSummand, _OreSummand, ...]`.

Dabei entsprechen die Summanden den Termen des Polynomes und werden nach jeder Operation entsprechend der Termordnung geordnet. Das Ordnen und Zusammenfassen leistet der Befehl `OreSimplify`, der nach Ändern der Termordnung auf alle relevanten Objekte anzuwenden ist.

Die Termordnung ist über die Reihenfolge der Unbestimmten (also eine Permutation der internen Reihenfolge) und eine Matrix definiert. Um eine Termordnung zu definieren kann man eine Anzahl k von relevanten Unbestimmten und eine nichtsinguläre $k \times k$–Matrix angeben. Etliche übliche Ordnungen sind aber vordefiniert.

Im folgenden Beispiel zeigen die *Mathematica*-Funktion `FullForm` und der Befehl `OreShowStatus` aus der Implementation die wesentlichen Interna.

3.2.1 Die interne Struktur am Beispiel

Folgende Sitzung deklariert zwei Operatoren, generiert einen Ausdruck und gibt dessen interne Struktur und den Status des Systems aus. Dabei wird auf vordefinierte Operatoren zurückgegriffen.

Needs["Ore‘"];
DX = OreDefineDiff[Dx, x]; SN = OreDefineShift[Sn, n];

DE = OMu[SN, OPo[nx], DX, OPo[x]]

$(nx^2)(\text{DxSn}) + x^2(\text{DxSn}) + (nx)\text{Sn} + x\text{Sn}$

FullForm[DE]

OrePolynomial[OreSummand[Times[n, Power[x, 2]], OreMonomial[List[1, 1], List[2, 1]]],
OreSummand[Power[x, 2], OreMonomial[List[1, 1], List[2, 1]]],
OreSummand[Times[n, x], OreMonomial[List[2, 1]]],
OreSummand[x, OreMonomial[List[2, 1]]]]

(Hier wird klar, daß mit der internen Darstellung als Ausgabe nicht gearbeitet werden könnte.)

OreShowStatus

OreNumberOfVariables: 2

OreNumberOfOperators: 2

OreNumberOfObjects: 4

OperatorNames: {Dx, Sn}

σ‘s: {#1/.Ore‘Private‘fSig$17&, #1/.Ore‘Private‘fSig$28&}

δ‘s: {Function[{Ore‘Private‘P$},
Plus@@(#1[[2]]$\partial_{\text{First[\#1]}}$Ore‘Private‘P$&)/@Ore‘Private‘fDelt$17], 0&}

δ‘s: {Dx, Sn}

OperatorVariables: {x, n}

OreOperators: {Dx, Sn}

OreListOfOrderObjects: {Dx, Sn, x, n}

In current 'canonical Order': $\{\mathrm{Dx}, \mathrm{Sn}, x, n\}$

Ordered via Matrix:$\{\{1,0,0,0\},\{0,1,0,0\},\{0,0,1,0\},\{0,0,0,1\}\}$

3.3 Elementare Funktionen

Im Grundpaket sind außer den Ein- und Ausgabefunktionen, Addition und Multiplikation noch einige weitere elementare Funktionen implementiert.

3.3.1 Ein- und Ausgabe

Operatoren werden mittels `OreDefineOperator` eingeführt, sofern man keinen der vordefinierten Operatoren verwenden möchte.

Übergeben wird der zugehörige Algebrahomomorphismus σ durch seine Wirkung auf den Variablen und ebenso die zugehörige δ–Derivation. Weiterhin wird ein Ausgabesymbol festgelegt. Es kann auch eine "Wirkung" angegeben werden, mit der der Operator später Funktionen abändert.

Die folgenden Zeilen definieren explizit einen n–Shift, einen Differentialoperator und eine q–Differentiation[2] :

Needs["Ore`"]

NShift = OreDefineOperator[Sn, {n → n + 1}, {n → 0},#/.{n → n + 1}&]
Diff = OreDefineOperator[Dx, {X → X}, {X → 1}, D[#, X]&]
Qdiff = OreDefineOperator[Dq, {X → qX}, {X → 1},
Together[((#/.{X → qX}) − #)/((q − 1)X)]&]

Für Standardoperatoren liegen natürlich generierende Funktionen vor, etwa
NShift = OreDefineShift[Sn, n]
leistet dasselbe wie die komplizierte Form im Beispiel.

Die Rückgabe von `OreDefineOperator` ist ein Algebraelement, welches aus dem Operator besteht (vom Typ `OrePolynomial`). Dieses sollte man unbedingt

[2]Die q–Differentiation D_q operiert vermöge $f(X) \mapsto \frac{f(qX)-f(X)}{qX-X}$. Die Vertauschungsregel in der Ore–Algebra ist $D_q X = qXD_q + 1$.

zuweisen, da ein weiteres `OreDefineOperator` die Anzahl der Unbestimmten der Ore–Algebra erhöhen würde. Im Aufruf auftauchende Variablen werden automatisch der Liste der bekannten Variablen zugefügt.

Tauchen noch weitere Variablen auf, so können diese mit `OreDeclareVariables` deklariert werden.

Sind die Daten der Algebra bekannt, kann man aus vorgegebenen Monomordnungen eine auswählen oder eine beliebige Matrixordnung angeben. Die Ordnung kann man auch später wieder umändern. Standardordnung ist lexikografische Ordnung der Unbestimmten in der Reihenfolge ihrer Deklaration.

Fügt man im Laufe einer Rechnung weitere Operatoren ein, erweitert also die Algebra, so sollte natürlich auch die Monomordnung wieder angepaßt werden.

3.3.2 Addition und Multiplikation

Durch die Attribute “Flat” und “Orderless” wird bei der Auswertung von Ausdrücken der Form `OreMonomial[{`r_1`,`n_1`},...,{`r_k`,`n_k`}]` durch *Mathematica* eine automatische Auflösung von verschachtelten Operatormonomen und eine Sortierung erreicht. Mit der in *Mathematica* eingebauten Mustererkennung und -abarbeitung wird durch Zusammenfassen gleicher Operatoren erreicht, daß die $r_i, i = 1, \ldots, k$ stets verschieden sind. Damit ist nebenher die Multiplikation zweier Operatormonome a, b mittels `OreMonomial[a,b]` realisiert.

Zentrales Hilfsmittel ist die Vereinfachung eines Ausdruckes der Gestalt `OrePolynomial[_OreSummand, _OreSummand, ...]`.

Mit der Prozedur `OreSimplify[...]` wird ein Ore-Polynom in eine kanonische Form gebracht: die Summanden werden gemäß der Monomordnung fallend sortiert und gegebenenfalls zusammengefaßt.

Damit ist aber die Addition bereits implementiert, indem einfach alle in Frage kommenden Summanden hintereinandergeschrieben werden und `OreSimplify` angewandt wird.

Die Multiplikation ist naturgemäß schwerer, insbesondere, da wir nicht mehr als σ_i, δ_i haben, um Ausdrücke der Gestalt $D_i^k P, P \in K[X_1, \ldots, X_m]$ in kanonische Form zu bringen. Im Wesentlichen wird in der Implementation die Regel $DP = \sigma(P)D + \delta(P)$ iteriert, bis die Normalform erreicht ist. Das ist das im Abschnitt

(2.6) vorgestellte Verfahren.

Potenzierung wurde via $\alpha^k = (\alpha^{[\frac{k}{2}]})(\alpha^{k-[\frac{k}{2}]})$ realisiert, um mit dem Memoryeffekt logarithmischen Aufwand zu erreichen.

Exzessives Anwenden der Memorytechnik führte allerdings praktisch zu Speicherproblemen.

3.3.3 Untersuchung von Ore–Polynomen

Hierunter fallen beispielsweise das Bilden der Koeffizientenliste, Ausführen einer beliebigen Operation auf alle Koeffizienten eines Ore–Polynomes und das Bestimmen des Leitkoeffizienten.

Das Bestimmen des Inhalts eines Ore–Polynomes mit ganzzahligen Koeffizienten, also des ggt der Koeffizienten und des durch diesen dividierten Polynomes kann einen “intermediate expression swell“ etwas eindämmen.

Eine damit verwandte Funktion bestimmt den “Variablen-Inhalt”, also den ggt der Polynome in den kommutativen Ore–Variablen, die man erhält, wenn man gleiche Operatormonome zusammenfaßt. Sie ergibt eine Teilfaktorisierung, denn dieses Polynom kann man nach links ausklammern.

Im Grundpaket ist auch das S-Polynom realisiert, welches man vielleicht dort nicht erwartet. Der Grund dafür ist, daß für diese Funktion stark auf die interne Struktur zurückgegriffen wird. Gleiches gilt für das Extrahieren des Initialtermes eines Ore–Polynoms.

3.3.4 Matrixordnungen betreffende Funktionen

Später wird der Gröbner–Walk untersucht. Bei dieser Anwendung ändert sich sukzessive die gegebenen Matrixordnung — somit ist eine effiziente Ordnungsänderung notwendig (`RedefineOrder`).

Polynome werden in der interenen Darstellung nach der aktuellen Monomordnung sortiert dargestellt. Einige Funktionen verlassen sich auf diese Darstellung. Daher ist es nötig mit jeder Änderung der Ordnung die in der Polynomdarstellung inhärente Sortierung neu anzustoßen, etwa durch `OreSimplify`.

Mit `OreRedefineMonomOrder[Monomials_List, Matrix_List]` ändert man die aktuelle Ordnung. Die übergebene Liste aller Unbestimmten legt deren Reihenfolge fest. Das zweite Argument ist eine die Ordnung definierende Matrix, die die Gewichtung der Unbestimmten beschreibt.

Der Benutzer muß dabei nichts über die interne Variablenliste wissen.

Umgekehrt läßt sich die aktuelle Ordnung mit `OreGetMonomOrder[]` auslesen. Es wird die Variablenliste und die die aktuelle Ordnung repräsentierende Matrix zurückgegeben. Die Beispiele im Anhang zeigen, wie man recht effizient damit arbeiten kann.

3.3.5 Die höheren Pakete

Auf dem Grundpaket aufbauend definiert das Paket "OreAlgorithms" höhere Funktionen, wie den Divisionsalgorithmus und das Bestimmen der Linksgröbnerbasis.

Das Paket "Groebnerwalk" schließlich realisiert als oberste Schicht eine Anwendung, die den bisher im nichtkommutativen Fall noch nicht untersuchten Gröbner–Walk untersuchen hilft. Die nötigen Berechnungen und auch der Algorithmus selbst sind dort implementiert. Die Anwendung funktionierte in einigen untersuchten Beispielen, was eine Beschäftigung mit dem Prinzip für Ore–Algebren ordnungserhaltender Art motivierte.

Die Vorstellung dieser Pakete greift dem folgenden theoretischen Teil zu Gröbnerbasen und Gröbner–Walk etwas voraus, findet der Vollständigkeit halber aber trotzdem hier statt.

Mein Paket liefert (im Gegensatz zu anderen) die Möglichkeit eines automatischen Korrektheitsbeweises dafür, daß die gefundene Basis das gegebene Ideal erzeugt.

In "OreAlgorithms" sind erweiterte Versionen für einige Algorithmen implementiert. Dabei wird zu Elementen $f \in \langle g_1, \ldots, g_n \rangle$ eines endlich erzeugten Linksideals als zusätzliche Information eine Liste von Koeffizienten $a_1, \ldots, a_n$ mit $f = \sum_{i=1}^{n} a_i g_i$ mitgeführt.

Der erweiterte Algorithmus zur Bestimmung der Linksgröbnerbasis gibt nicht nur Erzeuger der Basis zurück, sondern zu jedem berechneten Element f_j der

Gröbnerbasis auch "Zeugen", nämlich Koeffizienten a_{ij}, aus denen man sofort $f_j = \sum_{i=1}^{n} a_{ij} g_i \in \mathcal{I}$ nachrechnen kann. Die Beziehung $\langle f_1, \ldots, f_m \rangle \subset \mathcal{I}$ läßt sich bekanntlich durch Polynomdivision aller Erzeuger g_i durch $\langle f_1, \ldots, f_m \rangle$ nachweisen. Die umgekehrte Beziehung $\mathcal{I} \subset \langle f_1, \ldots, f_m \rangle$ läßt sich nun auch ohne blindes Vertrauen in die Computerberechnung mit den Zeugen nachweisen.

Diese Buchführung hat sich als effektiv etwa bei der Fehlersuche in Algorithmen erwiesen: sie führt eine (mittels "CheckExtended") nachprüfbare Redundanz ein.

Der Gröbner–Walk für Algebren ordnungserhaltender Art schließlich läßt sich im durch das Grundpaket und "OreAlgorithms" gegebenen "Labor" recht einfach implementieren. Da dieser Algorithmus sehr auf der Ordnungsstruktur basiert, mußten einige recht spezielle Befehle in das Grundpaket aufgenommen werden, damit die Interna der Darstellung weiter abgekapselt bleiben ("NextDegeneration").

Die Hauptfunktion des "Groebnerwalk"–Paketes übernimmt eine Gröbnerbasis zur aktuellen Ordnung und eine Zielordnung. Dann werden die Transformationsschritte des Gröbner–Walk durchgeführt.

Als Nebeneffekt wird im Erfolgsfall in die Zielordnung gewechselt. Wir werden sehen, daß der Erfolg leider im Allgemeinen nicht garantiert ist.

3.4 Schwächen der Implementation

Das Paket ist bereits für Experimente mit Ore–Algebren verwendbar. Um aber große Beispiele aus der Praxis zu behandeln, muß man sich über etliche Optimierungen Gedanken machen und eventuell maschinennähere Implementationen für zeitintensivere Routinen in Erwägung ziehen.

Der Ansatz, die Variablen und Operatoren auch in der Implementation getrennt zu halten, um auf den Variablen die kommutativen, in *Mathematica* implementierten Operationen nutzen zu können (etwa die Substitution um die σ_i effektiv zu realisieren), hat sich rückblickend als nicht befriedigend herausgestellt.

Der entwickelte Code ist allerdings in zwei Schichten möglichst modular aufgebaut, so daß auch Grundoperationen auf alternative Weise implementiert werden können, ohne die anderen Ebenen zu ändern.

Weiter ist die Frage berechtigt, ob *Mathematica* das optimale Computeralgebrasy-

stem für diesen Zweck ist. Es gab einen gewissen Ehrgeiz diese Funktionalitäten in einem System zu realisieren, in dem sie bisher noch *nicht* implementiert wurden. Die Verwendung der Substitution in *Mathematica* ist zwar sehr bequem, aber macht, so wie sie implementiert ist, viele Kontextwechsel nötig (es werden z.B. Terme in *Mathematica*–Darstellung in eine Vektordarstellung konvertiert).

Insgesamt ist aufgrund der hohen Rechenzeiten die vorliegende Implementierung für kompliziertere Anwendungen in der Praxis so noch nicht tauglich. Es wurden allerdings auch nahezu keine Maßnahmen zur Beschleunigung ergriffen, so daß hier Raum für Verbesserungen ist.

4 Gröbnerbasen

Es folgt eine Erinnerung an die Definition von Gröbnerbasen in kommutativen Polynomringen und anschließend Verallgemeinerungen für verschiedene nichtkommutative Strukturen.

4.1 Der kommutative Fall

Für kommutative Polynomringe sind Gröbnerbasen sehr intensiv erforscht worden. Eines von vielen Büchern dazu ist [BeWpf].

Grundsituation ist hier ein Körper k, Unbestimmte $X_1, \ldots, X_n$ und der Polynomring $R := k[X_1, \ldots, X_n]$.

Ein Element dieses Ringes hat dann die Gestalt:

$$f = \sum_{i \in I} \underbrace{a_i \overbrace{X^i}^{\text{“Monom”}}}_{\text{”Term”}}$$

mit einer endlichen Menge I von Multiindizes. Gegeben sei weiter eine “passende” Ordnung auf der Menge der Monome. Die Darstellung sei so, daß Monome der Terme strikt geordnet sind.

Der Leitsummand sei der Term mit dem größten Monom als Koeffizienten.

Zu einem Ideal $\mathcal{I} \in R$ heißt eine Menge $F = \{f_i \in I, i \in A\}$ mit einer Indexmenge A eine Gröbnerbasis, wenn $\mathcal{I} = \langle F \rangle$ und zudem die Leitsummanden der f_i alle Leitsummanden von Elementen aus $\mathcal{I}$ erzeugen: $f \in \mathcal{I} \Rightarrow \exists f_i \in F, r \in R$ mit $\mathrm{LT}(f) = \mathrm{LT}(r\mathrm{LT}(f_i))$. Die letzte etwas gestelzt wirkende Formulierung ist auch für Algebren ordnungserhaltender Art sinnvoll.

Natürlich interessieren vor allem endliche Gröbnerbasen, wenn man damit Ideale in einem Computer darstellen will[3].

[3] Meines Erachtens wäre es allerdings auch denkbar, gewisse unendliche Erzeugendensysteme über endliche Rekursionsregeln zu erfassen. Im später folgenden Beispiel für ein Ideal ohne endliche Gröbnerbasis werden wir eine solche Darstellung sehen. Eine Untersuchung, inwiefern solche Rekursionen automatisiert erkannt werden können, dürfte recht anspruchsvoll sein.

Im kommutativen Fall haben alle endlich erzeugten Ideale auch endliche Gröbnerbasen.

4.2 Der Buchberger-Algorithmus

Der Buchbergeralgorithmus aus [BeWpf] wird hier kurz skizziert.

4.2.1 Grundbegriffe des Algorithmus

Zunächst definieren wir "S-Polynom" und "reduziert sein". Dabei ist (im kommutativen Polynomring) das S-Polynom von Polynomen $p, q \neq 0$ eine Differenz $M_p p - M_q q$ mit Monomen M_p, M_q möglichst kleinen Grades derart, daß sich die Leitmonome von $M_p p, M_q q$ in der Differenz herausheben. Diese Monome sind gerade der kgv der Leitmonome dividiert durch den jeweiligen Leitterm. Somit haben wir:

Definition 4.1 (S–Polynom im Polynomring) *Sei $R = k[X_1, \ldots, X_n]$ der Polynomring über einem Körper k versehen mit einer Monomordnung. Zu normierten Polynomen $p, q \in R \backslash \{0\}$ sei das S–Polynom durch*

$$\mathrm{S}(p,q) := p \cdot \mathrm{kgv}(\mathrm{LT}(p), \mathrm{LM}(q))/\mathrm{LM}(p) - q \cdot \mathrm{kgv}(\mathrm{LM}(p), \mathrm{LT}(q))/\mathrm{LM}(q)$$

definiert.

Definition 4.2 (Reduktion) *Sei R wie in Definition 4.1. Seien $f \in R$ und $P \subset R$. Haben wir einen Term $t \in \mathrm{T}(f)$ und ein $h \in R$ mit $t = \mathrm{LT}(hp), p \in P$, so nennen wir $f' := f - hp$ eine* Reduktion *von f nach p und schreiben $f \xrightarrow{p} f'$. Wir sagen dann auch f ist modulo P reduzierbar und f' ist eine Reduktion von f modulo P.*

Definition 4.3 (Reduziert bezüglich einer Menge von Polynomen) *Ein Polynom q heißt reduziert bezüglich einer Menge P von Polynomen, wenn kein Term von q Vielfaches eines Leittermes eines Polynomes aus P ist.*

Durch fortgesetzte Reduktion modulo P kann man ein Polynom q in eine bezüglich P reduzierte Form q' bringen. Dieses Verfahren ist eine verallgemeinerte Division mit Rest und q' nennt man auch eine Normalform von q bezüglich P.

Diese Darstellung ist im Allgemeinen nicht eindeutig, sondern hängt von der Reihenfolge ab, in der die Terme eliminiert werden.

Wie hier beschrieben, wird eigentlich nur der Rest bestimmt, aber man kann natürlich buchführen, welche Vielfachen man subtrahiert hat.

Ist P eine Gröbnerbasis, so reduzieren sich genau die Elemente des von P erzeugten Ideales zu 0.

Definition 4.4 *Eine Menge von Polynomen P heißt reduziert, wenn jedes $p \in P$ reduziert ist bezüglich $P \setminus \{p\}$.*

Für eine reduzierte Gröbnerbasis P von $\mathcal{I}$ liefert jede Polynomdivision sogar einen eindeutigen Repräsentanten der Restklasse $q + \mathcal{I}$. Genau genommen ist der Begriff Normalform also nur für reduzierte Gröbnerbasen korrekt.

4.2.2 Der einfache Buchberger–Algorithmus

Der Buchberger–Algorithmus startet mit einer endlichen Idealbasis B. Gegeben sei ferner eine Zuordnung, die jedem Polynom q eine Normalform q' bezüglich B zuweist.

Findet man nun zwei Polynome $p, q \in B$, für die $s = \mathrm{S}(p, q)$ mit Normalform $s' \neq 0$ gilt, so nimmt man s' zu der Menge B hinzu.

Das Abbruchkriterium ist die Reduktion aller möglichen S-Polynome zu 0.

Die Terminierung ist stets gegeben. Ansonsten bildeten die jeweils neu hinzukommenden s_i' eine unendliche Folge, mit $\mathrm{LM}\left(s_j\right) \nmid \mathrm{LM}\left(s_i\right), i > j$. Dies ist ein Widerspruch zu Dicksons Lemma, das besagt, daß zu jeder beliebigen Menge M von Monomen eine *endliche* Teilmenge $M' \subset M$ existiert, so daß jedes $m \in M$ Vielfaches eines $m' \in M'$ ist ([BeWpf], p. 163, Corr. 4.8, auf die Exponenten angewandt).

In [BeWpf] werden etliche Beschleunigungen angegeben, die meist überflüssige Berechnungen von S-Polynomen einsparen.

In [KRW] wird für Algebren ordnungserhaltender Art gezeigt, daß dieser Algorithmus auch hier zum Erfolg führt. Die Bedingung ordnungserhaltender Art zu sein, läßt sich relativ leicht überprüfen und ist in vielen Anwendungsbeispielen erfüllt.

Es mag auch interessant sein zu untersuchen, in welchen Fällen der Algorithmus von Buchberger auch ohne diese Bedingung zum Ziel führt, was hier aber nicht weitergeführt wird.

4.3 Der allgemeine nichtkommutative Fall

Eine Übertragung des Buchbergeralgorithmus auf den allgemeinsten nichtkommutativen Fall einer freien Algebra über n Unbestimmten hat [Mora86] 1986 gegeben — eine Fleißarbeit, die recht schwer zu lesen ist. In diesem Falle sind bereits die Monome recht unhandliche Gebilde. Andererseits vermeidet das Fehlen von Relationen im frei-nichtkommutativen Fall auch einige Komplikationen.

In genauso allgemeinem Kontext hat [Keller] 1997 in seiner Dissertationsarbeit Algorithmen und Ordnungen zum Auffinden von Gröbnerbasen untersucht.

4.3.1 Eine allgemeine Konstruktion ist nicht möglich

Ein klassisches, nicht entscheidbares Problem ist das Wortproblem (die Entscheidung, ob ein bestimmtes Wort zu einer formalen Sprache gehört.). Es gibt nachweislich keinen Algorithmus, der es (bei hinreichend komplizierten Sprachen) in endlicher Zeit entscheiden kann.

Dieses Problem läßt sich in das Ideal-Element-Problem einbetten, so daß auch das Ideal-Element-Problem nicht für allgemeine nichtkommutative Algebren algorithmisch entscheidbar ist.

Hätte man einen Algorithmus, der in endlicher Zeit zu jedem Ideal eine Gröbnerbasis findet, so könnte man mit dieser Basis auch die Ideal-Element Frage klären.

Auch gibt es keinen Algorithmus zur Berechnung einer partiellen Gröbnerbasis, der alle Elemente bis zu einem gegebenen Grad berechnet (auch damit ist das Ideal-Element-Problem ja gelöst) [Mora94],p 158, ff.

An derselben Stelle wird angegeben, daß es aber für homogene Ideale sehr wohl

möglich ist, eine solche grad-partielle Gröbnerbasis zu bestimmen.

Möchte man einen Algorithmus finden, kommt man nicht umhin, sich in der Wahl der Algebra oder der untersuchten Ideale einzuschränken.

4.4 Spezialfall Ore–Algebren über Polynomringen

In dieser Arbeit besteht diese Spezialisierung in der Betrachtung von Ore–Algebren ordnungserhaltender Art.

Viktor Levandovskyy hat sich in [Levand] intensiv mit Gröbnerbasen in G–Algebren (aufbauend auf Konstruktionen von Joachim Apel) und verschiedene Anwendungen in diesem Kontext untersucht. Dabei sind G–Algebren etwas allgemeiner als Ore–Algebren. Seine Resultate gehen auch in die Erweiterung “Plural” des Computeralgebrasystemes “Singular” ein.

Der Ansatz Algorithmen zu suchen, die eine bestimmte Idealstruktur voraussetzen wäre auch denkbar, (schließlich wird ja auch das Wortproblem durch Beschränkung auf gewisse Sprachen angegangen).

Im gewählten Setting führt das Verfahren von Buchberger zum Ziel. Dabei ist das Vorgehen das gleiche wie im kommutativen Fall beschrieben (im von Mora beschriebenen frei-nichtkommutativen Fall sind stärkere Abänderungen nötig). Die wesentliche Änderung liegt bei der Bestimmung des S-Polynomes, die im Folgenden beschrieben wird.

4.4.1 S-Polynome in Ore–Algebren

Für beliebige Ore–Algebren aber stellt sich bereits die Frage, was das S-Polynom ist. Nehmen wir folgende Ore–Algebra:

$$\mathcal{O} = k\left\langle X, D, \sigma : X \mapsto X^2, \delta = 0 \right\rangle \ ,$$

Nun betrachten wir das Linksideal $\mathcal{I} = \langle X, XD \rangle$. Im Kommutativen hätten wir $\mathrm{kgv}(X, XD) = XD$ und das S-Polynom wäre $D \cdot X - 1 \cdot XD = 0$.

Dies verschwindet aber mit der Ore-Multiplikation in $\mathcal{D}$ offenbar nicht! Allerdings handelt es sich bei $\{X, XD\}$ um eine Gröbnerbasis:

Für ein Monom $m = X^a D^b$ gelten $mX = X^a D^b X = X^a X^{2^b} D^b$ und $mXD = X^a D^b XD = X^a X^{2^b} D^{b+1}$.

Alle Terme aus $\mathcal{I}$, also auch die Leitkoeffizienten der Elemente des erzeugten Ideales, sind also von der Gestalt $\alpha X^i D^j \ , i \geq 2^{j-1} \ , \alpha \in k$ und damit für $j = 0$ sofort aus $\mathcal{I}$ oder gleich $\alpha X^{i-2^{j-1}} D^{j-1} XD \in \mathcal{I}$.

In diesem Beispiel wird das Wegheben der Leitmonome dadurch boykottiert, daß ein zusätzlicher Faktor X entsteht, wenn DX in Normalform gebracht wird (also $\mathrm{kgv}(X, XD)/X$ so nicht definiert ist). Die richtigen Faktoren g, f, damit $g \cdot X - f \cdot XD$ verschwindet sind hier $g = D, f = X$. Dabei werden die Faktoren g, f jeweils von links an die Elemente multipliziert, deren S-Polynom zu bilden ist.

Bildet man das (Rechts) S–Polynom durch Multiplizieren mit geeigneten Faktoren von rechts, ergibt sich hier dasselbe Resultat wie im kommutativen Fall (hier taucht der Unterschied etwa bei $\{D, XD\}$ auf).

In Zukunft wird das S-Polynom durch Multiplizieren von links gebildet, dies ist nötig um Gröbnerbasen von Linksidealen zu finden.

Untersucht man beispielsweise Nullstellenmengen von Operatorausdrücken, die Elementen $\{\varphi_i, i = 1, \ldots, n\}$ einer Ore–Algebra entsprechen, so kann man zu dem von diesen Ausdrücken erzeugten Linksideal $\langle \varphi_i, i = 1, \ldots, n \rangle$ übergehen.

In [Levand] ist auch ein Verfahren aufgeführt, wie die Berechnung von Rechtsgröbnerbasen mit demselben Algorithmus wie die Berechnung von Linksgröbnerbasen durchgeführt werden kann, indem von der Ursprungsalgebra zur opponierten Algebra übergegangen wird, in der $a \cdot b$ durch das Produkt ba aus der Ursprungsalgebra definiert ist.

Für ein Monom

$$t = \prod_{i=1}^{m} X_i^{\alpha_i} \prod_{j=1}^{n} D_j^{\beta_j} \ , \alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n \in I\!N_0$$

bezeichne $\mathrm{op}(t) := \prod_{j=1}^{n} D_j^{\beta_j}$ und $\mathrm{va}(t) := \prod_{i=1}^{m} X_i^{\alpha_i}$.

Gesucht sind nun die richtigen Linksmultiplikatoren für Monome $p, q \in \mathcal{D}$, also Monome f, g derart, daß fp und gq den gleichen Leitterm haben (wenn wir eine zulässige Ordnung betrachten, ist die Annahme p, q seien Monome keine wirkliche Einschränkung, für Polynome kann man zu den Leittermen übergehen).

Um f, g zu finden kann man wie folgt vorgehen, ausnutzend, daß die Operatoren und die Unbestimmten unter sich kommutieren:

Zunächst wird der ggt der Operatorvariablen $\mathrm{op}(p), \mathrm{op}(q)$ bestimmt und die Faktoren $f_{\mathrm{op}}, g_{\mathrm{op}}$, mit denen die Operatoranteile der gegebenen Monome auf ihren kgv gebracht werden.

Damit sind schon die "Operatorstrukturen" $f_{\mathrm{op}}, g_{\mathrm{op}}$ der Faktoren f, g bestimmt, sofern beim "Durchtauschen" der Variablen nach vorne in den Ausdrücken $f_{\mathrm{op}}p$, $g_{\mathrm{op}}q$ keine Operatoren verschwinden. Das ist der Fall, wenn die σ_i injektiv sind.

Setzt man nun $f = \overbrace{f_{\mathrm{op}}}^{\mathrm{op}(f)}\overbrace{f_{\mathrm{va}}}^{\mathrm{va}(f)}$ und $g = \overbrace{g_{\mathrm{op}}}^{\mathrm{op}(g)}\overbrace{g_{\mathrm{va}}}^{\mathrm{va}(g)}$ an, und geht davon aus, daß die δ_i vernachlässigbare Terme erzeugen, so bestimmt sich der Leitterm von fp zu $f_{\mathrm{va}}\mathrm{LT}\left(\sigma_f(p_{\mathrm{va}})\right) f_{\mathrm{op}}p_{\mathrm{op}}$ und der von gp ist in $g_{\mathrm{va}}\mathrm{LT}\left(\sigma_g(q_{\mathrm{va}})\right) g_{\mathrm{op}}q_{\mathrm{op}}$.

Dabei entsprechen die $\sigma_{f,g}$ den Verknüpfungen der sukzessive beim "Durchschieben" der Variablenanteile auf dem Weg zur Normalform anzuwendenden Homomorphismen σ_i. Die "Störterme", die durch die δ_i entstehen, verringern jeweils zumindest eine Operatorpotenz: für den Fall, daß die Operatoren in der Ordnung die Variablen dominieren, kann man die Störterme daher weglassen und erhält so das Leitmonom des Produktes.

Problematisch wird es, wenn man die entstehenden Terme also nicht vernachlässigen kann, es also gewisse δ_i und Polynome P mit $\sigma_i(P)D_i \preceq_l \delta_i(P)$ gibt.

Genau dieses Problem ist bei Ore–Algebren ordnungserhaltender Art nicht vorhanden — hier wird verlangt, daß $\mathrm{LM}\left(\sigma_i(P)D_i\right) > \delta_i(P_i)$.

Hier ist das Verfahren kompakt aufgeschrieben:

Definition 4.5 (S-Polynom) *Sei $\mathcal{D}$ eine Ore–Algebra mit einer zulässigen Termordnung $<$. Dann ist zu $p, q \in \mathcal{D}$ das S-Polynom $\mathrm{S}(p, q)$ wie folgt konstruiert:*

- *Setze* $r := \mathrm{kgv}(\mathrm{op}(\mathrm{LM}\left(p\right)), \mathrm{op}(\mathrm{LM}\left(q\right)))$.
- *Bestimme* $f_{\mathrm{op}}, g_{\mathrm{op}}$ *mit* $\mathrm{op}(\mathrm{LM}\left(p\right)) f_{\mathrm{op}} = r$, $\mathrm{op}(\mathrm{LM}\left(q\right)) g_{\mathrm{op}} = r$.

★$_1$ *Setze* $v := \mathrm{LT}\left(f_{\mathrm{op}}\mathrm{LM}\left(p\right)\right)$, $w := \mathrm{LT}\left(g_{\mathrm{op}}\mathrm{LM}\left(q\right)\right)$.

- *Setze* $s := \mathrm{kgv}(\mathrm{va}(v), \mathrm{va}(w))$.

- *Bestimme* $f_{\mathrm{va}}, g_{\mathrm{va}}$ *mit* $vf_{\mathrm{va}} = s, wg_{\mathrm{va}} = s$.
- *Erhalte* $f = f_{\mathrm{op}} f_{\mathrm{va}}$ *und* $g = g_{\mathrm{op}} g_{\mathrm{va}}$.

★$_2$ *Setze* $\mathrm{S}(p, q) := fp - gq$.

Kritisch sind dabei die mit ★ markierten Punkte. Bei ★$_1$ müssen die Leitterme der auftretenden Produkte auch die Terme mit den bestimmten Operatorvariablen $f_{\mathrm{op}}, g_{\mathrm{op}}$ sein. Bei ★$_2$ sollten sich die Leitterme in $fp - gq$ wegheben, insbesondere dürfen die niedrigeren Terme aus p, q dabei nicht stören. Das wird jeweils durch die Eigenschaft ordnungserhaltend zu sein garantiert.

4.5 Endlichkeit einer Gröbnerbasis

Bei Ore–Algebren muß zu einem gegebenen Ideal und gegebener Ordnung keine endliche Gröbnerbasis existieren. Wir nennen ein Ideal bezüglich einer gegebenen Ordnung *gröbnerdarstellbar*, wenn es bezüglich dieser Ordnung eine endliche Gröbnerbasis besitzt.

Im kommutativen Fall ist jedes endlich erzeugte Ideal gröbnerdarstellbar. Für Ore–Algebren muß das nicht mehr gelten, wie das folgende Beispiel zeigt.

4.6 Ein Ideal ohne endliche Gröbnerbasis

Sowohl für Links- als auch für Rechtsideale gibt es Beispiele von Nichtdarstellbarkeit. Es wird nun ein nicht gröbnerdarstellbares Linksideal angegeben.

Das hier verwendete Beispiel stammt im Wesentlichen aus [Pesch], Kap 8.5. Es dient als Beispiel für ein Ideal ohne endliche Gröbnerbasis.

Ich habe das dort recht allgemeine Beispiel hier auf den wesentlichen Spezialfall reduziert und an einer Stelle klarer argumentiert. Der Effekt wird so meiner Meinung nach besser demonstriert.

Wir betrachten die Ore–Algebra

$$\mathcal{O} = k[X_1, X_2] \langle D_1, \sigma_1, 0; D_2, \mathrm{id}, 0 \rangle \,,\ \sigma_1 \colon X_1 \mapsto X_1^2, X_2 \mapsto X_2,$$

in der die Menge der Terme multiplikativ abgeschlossen ist. Als Termordnung nehmen wir die lexikografische mit $X_1 \prec X_2 \prec D_1 \prec D_2$

Bemerkung 4.6 *$\mathcal{O}$ ist mit dieser Ordnung zulässig.*

Beweis: Mit 2.15 können wir uns auf Monome beschränken.

Seien $P, Q \in \mathrm{M}(\mathcal{O}), P \prec Q$ und $R, R' \in \mathrm{M}(\mathcal{O})$ gegeben, also

$$\begin{aligned} P &= X_1^{p_1} X_2^{p_2} D_1^{k_1} D_2^{k_2}, \\ Q &= X_1^{q_1} X_2^{q_2} D_1^{l_1} D_2^{l_2}, \\ R &= X_1^{r_1} X_2^{r_2} D_1^{s_1} D_2^{s_2}, \\ R' &= X_1^{r'_1} X_2^{r'_2} D_1^{s'_1} D_2^{s'_2}\ . \end{aligned}$$

Es folgt damit

$$RP = X_1^{r_1+p_1 2^{s_1}} X_2^{r_2+p_2} D_1^{k_1+s_1} D_2^{k_2+s_2}, \tag{3}$$

$$RPR' = X_1^{r_1+p_1 2^{s_2}+r'_1 2^{k_2+s_2}} X_2^{r_2+p_2+r'_2} D_1^{s_1+k_1+s'_1} D_2^{s_2+k_2+s'_2}, \tag{4}$$

$$RQR' = X_1^{r_1+q_1 2^{s_2}+r'_1 2^{l_2+s_2}} X_2^{r_2+q_2+r'_2} D_1^{s_1+l_1+s'_1} D_2^{s_2+l_2+s'_2}, \tag{5}$$

Die Exponenten von X_2, D_1, D_2 aus (4) und (5) unterscheiden sich von den entsprechenden in P und Q durch dieselbe Konstante. Soll der Exponent von X_1 die Relation zwischen (4) und (5) entscheiden, muß gemäß der lexikografischen Ordnung daher $p_2 = q_2, k_1 = l_1, k_2 = l_2$ gelten. In jedem Falle bleibt die Ordungsrelation erhalten. □

Wir betrachten das Linksideal $\mathcal{I} = \langle D_1 + D_2, X_1 D_1 \rangle$. Dann erhalten wir

Behauptung 4.7 $\mathrm{LT}(\mathcal{I}) = \mathrm{T}(\mathcal{O}) \cdot \left(\{D_2\} \cup \{X_1 D_1^k \mid k \in I\!N \setminus \{0\}\} \right),$

Beweis:

"$\supseteq$": Es gilt $D_2 = \mathrm{LT}(D_1 + D_2)$ und rekursiv

$$X_1 D_1^k = (X_1 D_1^{k-1})(D_1 + D_2) - D_2(X_1 D_1^{k-1})$$

"$\subseteq$": Sei $t \in \mathrm{LT}(\mathcal{I})$. Dann gibt es $f_1, f_2 \in \mathcal{O}$ mit

$$t = \mathrm{LT}(f_1 \cdot X_1 D_1 + f_2 \cdot (D_1 + D_2)) \ .$$

Wenn t Linksvielfaches von $X_1 D_1$ ist, sind wir fertig. Wenn t Linksvielfaches von D_2 ist, stammt es entweder aus $\mathrm{T}(f_1 X_1 D_1)$, und wir sind fertig, oder aus $\mathrm{T}(f_2 \cdot (D_1 + D_2))$. Dann ist $t = \mathrm{LT}(f_2 \cdot (D_1 + D_2))$ und wir sind ebenfalls fertig.

Ansonsten hat t die Gestalt $X_1^i X_2^j D_1^k$ und stammt aus $\mathrm{T}(f_2 D_1)$, damit ist $k > 0$, und es folgt $i = 0$, also $t = X_2^j D_1^k \in \mathrm{T}(f_2 D_1)$.

Dann gibt es ein $s = X_2^j D_1^{k-1} D_2 \in \mathrm{T}(f_2 D_1)$. Alle Terme, in denen D_2 auftaucht, sind größer als t, also muß s sich wegheben (t ist ja ein Leitterm).

Da der Faktor X_1 in s nicht vorkommt, kann sich s nur gegen ein $s' \in \mathrm{T}(f_2 D_1)$ wegheben, es gibt also ein $s' = (X_2^j D_1^{k-2} D_2) D_1 \in \mathrm{T}(f_2 D_1)\,, s' > t$, dazu wieder ein Pendant $s'' = X_2^j D_1^{k-2} D_2^2 \in \mathrm{T}(f_2 D_2)$ und so fort.

Auf die Weise erhielte man eine unendliche Folge von nicht verschwindenden Termen aus $\mathrm{T}(f_2 D_2)$, was nicht möglich ist.[4] □

Behauptung 4.8 *Das Linksideal $\mathcal{I}_1 = \langle X_1 \cdot D_1^n \mid n \in \mathbb{N} \rangle$ ist nicht endlich erzeugt*[5].

Beweis: Angenommen, I_1 wäre endlich erzeugt, dann wäre für ein $N \in \mathbb{N}$

$$\mathcal{I}_1 = \langle X_1 \cdot D_1^n \mid 0 \leq n < N \rangle \ .$$

Mit $X_1 D_1^N \in I_1$ gäbe es eine Darstellung mit $f_0, \ldots, f_{N-1} \in \mathcal{O}$ und

$$X_1 D_1^N = \sum_{i=0}^{N-1} f_i \cdot X_1 D_1^i \ .$$

[4] Die Argumentation bei [Pesch] ist hier nicht schlüssig.

[5] Dies ist ein Beispiel für ein nicht endlich erzeugtes Ideal, das man mit dem Parameter n durchaus symbolisch im Computer darstellen könnte. Ob eine Chance besteht, beim Buchbergeralgorithmus Rekursionen beim Generieren der S-Polynome zu erkennen und so auch gewisse nicht endliche Gröbnerbasen in den Griff zu bekommen? Es scheint mir ein ehrgeiziges Projekt.

Da in unserem Beispiel die Terme multiplikativ abgeschlossen sind, folgt aus dieser Gleichung, daß bereits für ein Monom $t = aX_1^c D_1^d \in \mathrm{T}(\mathcal{O}), a \in k^*$ für ein $f_i = X_1 D_1^i$ gilt:

$$X_1 D_1^N = t \cdot X_1 D_1^i, i < N \ .$$

Damit ergibt sich

$$X_1 D_1^N = aX_1^c D_1^d \cdot X_1 D_1^i = aX_1^{c+2^d} D_1^{d+i} \ ,$$

Vergleich der Exponenten in X_1, D_1 ergibt

$$\begin{aligned} 1 = c + 2^d &\Rightarrow c = 0, d = 0 \\ N = d + i &\Rightarrow N = 0 + i = i \end{aligned}$$

im Widerspruch zur Wahl von $i < N$. □

Bisher haben wir also eine Ore–Algebra $\mathcal{O}$, ein Ideal $\mathcal{I} \subset \mathcal{O}$ und eine Monomordnung $\prec$ gefunden, so daß $\mathcal{I}$ keine endliche Gröbnerbasis besitzt (da ja das von den Leitmonomen erzeugte Ideal ansonsten bereits endlich erzeugt wäre).

Andererseits kann durch einen Wechsel der Ordnung erreicht werden, daß doch eine endliche Gröbnerbasis existiert.

Betrachten wir dazu dasselbe Ideal $\mathrm{LT}(\mathcal{I}) = \mathrm{LT}(\langle D_1 + D_2, X_1 D_1 \rangle)$ aber die Ordnung $X_1 \prec_* X_2 \prec_* D_2 \prec_* D_1$.

Bemerkung 4.9 *$\mathcal{O}$ ist mit $\prec_*$ zulässig.*

Beweis: Sei wie in (4.6) $P \in \mathrm{M}(\mathcal{O})$ und $R, R' \in \mathrm{M}(\mathcal{O})$ gegeben, also $P = X_1^{p_1} X_2^{p_2} D_1^k D_2^l, R = X_1^{r_{x_1}} X_2^{r_{x_2}} D_1^{s_1} D_2^{s_2}, R' = X_1^{r'_{x_1}} X_2^{r'_{x_2}} D_1^{s'_1} D_2^{s'_2}$. Es folgt wie in (4.6)

$$RPR' = X_1^{p_1 \cdot 2^{s_2} + r_{x_1} + r'_{x_1} \cdot 2^{l+s_2}} X_2^{p_2 + r_{x_2} + r'_{x_2}} D_1^{k+s_1+s'_1} D_2^{l+s_2+s'_2}$$

Hier sind die Exponenten in D_1, D_2, X_2 offenbar streng monoton wachsend in k, l beziehungsweise p_2. Der Exponent von X_1 entscheidet wegen der lexikographischen Ordnung nur, falls die restlichen Exponenten konstant bleiben. Dieser Exponent wächst aber linear in p_1. □

Behauptung 4.10 *Die Menge* $\{D_2 + D_1, X_1D_2, X_1D_1\}$ *ist eine Gröbnerbasis von* $\mathcal{I} = \langle D_2 + D_1, X_1D_1\rangle$ *bezüglich der Monomordnung* $\prec_*$.

Beweis: Zunächst ist das ergänzende Element das S–Polynom $X_1D_2 = X_1(D_2 + D_1) - X_1D_1 \in \mathcal{I}$, also erzeugt die gegebene Menge nicht mehr als $\mathcal{I}$.

Es ist zu zeigen, daß jedes Leitmonom eines Idealelementes, also jedes $t = \mathrm{LT}\,(f \cdot (D_2 + D_1) + gX_1D_1)$ für Polynome f, g von der Gestalt rD_1, rX_1D_2 oder rX_1D_1 ist.

Sobald D_1 ein Faktor von t ist, ist nichts mehr zu zeigen, denn t hat dann die Gestalt rD_1.

Es bleibt der Fall $t = X_1^iX_2^jD_2^k \in \mathrm{T}(f \cdot (D_2 + D_1), k > 0)$ zu betrachten. Für $i > 0$ ist dann $t = (X_1^{i-1}X_2^jD_2^{k-1}X_1D_2$ von der Gestalt rX_1D_2.

Somit bleibt noch die Möglichkeit $t = X_2^jD_2^k \in \mathrm{T}(f \cdot (D_2 + D_1))$. Diese Möglichkeit können wir aber ausschließen, denn dann wäre $X_2^jD_2^{k-1}D_1 \in \mathrm{T}(f \cdot (D_2 + D_1))$ und könnte sich mangels X_1 nicht gegen einen Term aus gX_1D_1 wegheben. Mit $t = X_2^jD_2^k \prec_* X_2^jD_2^{k-1}D_1 \in \mathrm{T}(f \cdot (D_2 + D_1))$ hätten wir dann einen Widerspruch dazu, daß t ein Leitterm ist. □

Das untersuchte Beispiel zeigt also auch, wie die Ordnung die Endlichkeit einer Gröbnerbasis beeinflussen kann.

5 Der Gröbner–Walk

Der Gröbner–Walk ist eine Technik um eine in einer Ordnung gegebene Gröbnerbasis zu einer Gröbnerbasis in einer anderen Termordnung umzuwandeln.

Dies ist praktisch, da sich die Rechenzeiten für das Finden einer Gröbnerbasis abhängig von der verwendeten Ordnung sehr stark unterscheiden können: man kann dann in einer zum Berechnen günstigen Ordnung eine Gröbnerbasis erstellen, und diese in die neue Ordnung “transferieren”.

Da ich in der Literatur kein explizites vorführendes Beispiel für den Walk finden konnte (insbesondere mit den Randberechnungen), wird ein ausführliches Beispiel im kommutativen Fall gebracht.

Im kommutativen Fall ist der Gröbner–Walk auf jeden Fall durchführbar und realisiert. Die Untersuchung dieses Verfahrens im nichtkommutativen Fall ist allerdings Neuland.

Der Algorithmus nach ([AmGlKü]) wird nochmals aufgeführt. Dabei wird aufgezeigt, wo Sachverhalte genutzt werden, die in Ore–Algebren so nicht gelten. Beweise werden möglichst so formuliert, daß sie für Algebren ordnungserhaltender Art gelten: überall, wo zum Beispiel in ([AmGlKü]) mit Erhalt der ω-Homogenität argumentiert wird, muß die Argumentation oder der Algorithmus angepasst werden.

Das Verfahren ist mit den für diese Arbeit erstellten Paketen relativ leicht zu implementieren. Bei Beispielrechnungen für Ore–Algebren ordnungserhaltender Art führte der Walk zum Ziel. Es scheint unwahrscheinlich, daß in den betrachteten mehrschrittigen Beispielen die Gröbnerbasen verschiedener Längen nur zufällig korrekt ineinander überführt wurden. Damit ergab sich die Vermutung, daß der Gröbner–Walk auch für Ore–Algebren ordnungserhaltender Art funktioniert.

Schließlich zeigte aber ein Gegenbeispiel, daß der Gröbner–Walk zumindest für die gemischte Shift- und Differentialalgebra nicht 1:1 übertragbar ist.

Erstaunlich ist weniger der Fund eines Gegenbeispiels, als daß der Algorithmus so oft korrekte Resultate lieferte. Jedenfalls stand am Anfang der Analyse das Experiment mit dem für diese Arbeit erstellten *Mathematica*-Paket.

Wenngleich eine korrekte Variante des Gröbner–Walk für Algebren ordnungserhaltender Art ein sehr viel schönerer Abschluß dieser Arbeit wäre, ist mein Resul-

tat, daß auf dem Pfad des Gröbner–Walk (der verschiedene Zwischenordnungen repräsentiert) alle Ordnungen zur gegebenen Algebra passen — aber dennoch der Übergang nach der Methode von ([AmGlKü]) scheitern kann. Die Analyse des Gegenbeispiels gibt Hinweise darauf, wie sich der Walk modifizieren ließe, was aber hier nicht mehr ausgeführt werden konnte.

Zunächst aber werden die Begrifflichkeiten eingeführt.

5.1 Bezeichnungen

Wir haben einen Polynomring $k[X_1, \ldots, X_n]$ in n Unbestimmten und eine $n \times n$–Matrix M.

Mit $\prec_M$ bezeichnen wir die durch M definierte Matrixordnung auf den Monomen.

Die erste Zeile von M stellt eine Linearform ω dar, die *Leitform* von $\prec_M$.

Eine Linearform ω definiert den ω–Grad in $k[X_1, \ldots, X_n]$: Einem Monom $p = \prod_{i=1}^{n} X_i^{a_i}$ wird der ω-Grad $\deg_\omega(p) := \sum_{i=1}^{n} a_i\omega_i$ zugeordnet. Dies ist sozusagen die erste Instanz der Matrixordnung zu M.

Einem Polynom f wird als ω–Grad $\deg_\omega(f)$ das Maximum $\max_{p \in \mathbb{M}(f)} \deg_\omega(p)$ der ω-Grade all seiner Monome zugeordnet.

Jedes Polynom $f \in k[X_1, \ldots, X_n]$ ist dann in ω–homogene Summanden aufspaltbar, indem die Terme gleichen ω–Grades zusammengefaßt werden.

Der Anteil mit den Summanden maximalen ω–Grades heiße dann das *Initialpolynom* $\mathrm{in}(f, \omega)$. Für eine Menge G sei $\mathrm{in}(G, \omega) := \{\, \mathrm{in}(f, \omega) | f \in G\}$.

Beispiel 5.1 *Für die gradlexikografische Ordnung ist das Initialpolynom von* $f = 3X_1^3 - X_1X_2X_3 + X_3^2 - X_1X_2 + 1$ *gerade* $3X_1^3 - X_1X_2X_3$.

Da die Leitform der gradlexikografischen Ordnung alle Exponenten addiert, ist die Aufspaltung in homogene Summanden gerade $f = (3X_1^3 - X_1X_2X_3) + (X_3^2 - X_1X_2) + 1$. □

Definition 5.2 (Verfeinerung einer Halbordnung) *Ist eine Halbordnung $<$ auf den Monomen gegeben, so kann diese durch eine beliebige Ordnung $\prec_h$ zu einer totalen Ordnung $\prec$ verfeinert werden vermöge*

$$x \prec y \quad :\Leftrightarrow \quad x < y \vee (x \not< y \wedge y \not< x \wedge x \prec_h y)$$

Definition 5.3 ($\mathbf{Ord}\,(\boldsymbol{\omega}, \boldsymbol{A})$) *Sei A eine $n \times n$–Matrix, $\omega : \mathbb{R}^n \to \mathbb{R}$ eine Linearform, dargestellt als $1 \times n$–Matrix. Dann sei $\mathrm{Ord}\,(\omega, A)$ die Matrixordnung gegeben durch die Matrix, in der die Zeile ω den Zeilen von A vorangestellt wird.*

Damit haben wir gleich ein Beispiel für 5.2:

Beispiel 5.4 *Sei eine Linearform durch einen Zeilenvektor ω gegeben. Verfeinert werde die Halbordnung $<_\omega$ durch $\prec_A$ mit der Matrix A zu der Ordnung $\prec$. Dann ist $\prec$ gerade die Ordnung $\mathrm{Ord}\,(\omega, A)$.*

Beweis: Um $x <_{\mathrm{Ord}(\omega,A)} y$ zu entscheiden, werden nach Definition der Matrixordnung zunächst $\omega^{\mathrm{T}} \log x$ und $\omega^{\mathrm{T}} \log y$ verglichen. Bei Gleichheit lexikografisch die weiteren Komponenten $A \log x, A \log y$. Das ist aber gerade die Verfeinerung von $<_\omega$ mit $\prec_A$ nach 5.2. □

Der folgende Satz ist im kommutativen Fall klar, wir zeigen ihn auch für Algebren ordnungserhaltender Art.

Satz 5.5 *Gegeben sei eine Ore–Algebra $\mathcal{D}$ ordnungserhaltender Art. Die Monomordnung sei eine Matrixordnung mit der Initialform ω. Für $h, g \in \mathcal{D}$ gilt dann $\mathrm{in}(hg, \omega) = \mathrm{in}(\mathrm{in}(h, \omega)\,\mathrm{in}(g, \omega), \omega)$.*

Beweis: Wir schreiben $h = \mathrm{in}(h, \omega) + h'$ und $g = \mathrm{in}(g, \omega) + g'$. Dann ist

$$hg = \mathrm{in}(h, \omega)\,\mathrm{in}(g, \omega) + hg' + h'\,\mathrm{in}(g, \omega)$$

Wir zeigen: für alle $t \in \mathrm{T}(hg')$ gilt $t \prec_\omega \mathrm{LT}\,(hg)$. Ist $t \in \mathrm{T}(h\mathrm{LT}\,(g'))$, so folgt das wegen $\mathrm{LT}\,(g') \prec_\omega \mathrm{LT}\,(g)$ mit der Monotonie von $\prec_\omega$. Für beliebige $\tilde{g} \in \mathrm{T}(g')$ ist dann $h\tilde{g} \preceq_l h\mathrm{LT}\,(g')$. Mit dem gleichen Argument sind alle Terme aus $h'\,\mathrm{in}(g, \omega)$ bezüglich $\prec_\omega$ echt kleiner als $\mathrm{LT}\,(hg)$. Bleiben nur die Terme aus $\mathrm{in}(h, \omega)\,\mathrm{in}(g, \omega)$ als Kandidaten für den Initialterm von hg bezüglich ω. □

Definition 5.6 (kompatible Halbordnung) *Sei eine Halbordnung $<$ auf den Monomen gegeben sowie eine Monomordnung $\prec$ und eine Menge G von Polynomen.*

Dann lassen sich die Terme jedes Polynoms $g \in G$ fallend nach $\prec$ sortieren. Wir erhalten eine Darstellung $g = \sum_{i=1}^{m} g_i$.

Die Halbordnung $<$ heißt dann kompatibel zu $\prec$ auf G*, wenn dabei stets $g_j \leq g_1$ gilt, also für alle $g \in G$ das Leitmonom bezüglich $\prec$ in der Halbordnung $<$ nicht kleiner als eines der anderen Monome ist.*

Die Verfeinerung $\prec_+$ einer Halbordnung $<$ durch eine beliebige Ordnung $\prec$ ist daher per Definition kompatibel zu $<$.

Der folgende Satz wird in [AmGlKü] ohne Beweis implizit mit der Bemerkung, das Gewicht ω_k sei kompatibel zu der vorherigen Ordnung $\prec_{k-1}$ verwendet. Er gibt auch gleich einen Einblick, wie die Zwischengewichte im Gröbner–Walk zustande kommen.

Satz 5.7 *Seien Matrixordnungen $\prec_A, \prec_B$ zu den Matrizen mit den Zeilen*

$$A = \begin{pmatrix} \omega \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{pmatrix}, B = \begin{pmatrix} \omega_1 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{pmatrix}$$

gegeben. Bezeichne $\tau(t) := (1-t)\omega + t\omega_1$ einen Punkt auf der Geraden $\overline{\omega\omega_1}$.

Sei G eine endliche Menge von Polynomen. Sei $s \in [0,1]$ maximal mit der Eigenschaft, daß für kein $t \in [0,s)$ ein $g \in G$ mit einem Term $g' \in \mathrm{T}(g) \setminus \mathrm{T}(\mathrm{in}(g,\omega))$ mit $\deg_{\tau(t)}(g) = deg_{\tau(t)}(g')$ existiert.

Für alle $g \in G$, $g' \in \mathrm{T}(g)$ gilt dann $\mathrm{LM}_{\prec_A}(g) \succeq_{\tau(s)} g'$.

Beweis: Zunächst bemerken wir, daß $\deg_{\tau(t)}(g)$ als Maximum von Linearformen eine stetige Funktion in t ist.

Sei $g \in G$ mit $g_0 := \mathrm{LM}_{\prec_A}(g)$. Wir betrachten einen weiteren Term $g_1 \in \mathrm{T}(g)$.

Sei zunächst $g_1 \notin \mathrm{T}(\mathrm{in}(G,\omega))$. Angenommen, es wäre $\deg_{\tau(s)}(g_0) < deg_{\tau(s)}(g_1)$.

Dann gäbe es wegen $\deg_{\tau(0)}(g_0) = \deg_{\tau(0)}(g) > deg_{\tau(0)}(g_1)$ nach dem Zwischenwertsatz ein $t \in (0, s)$ mit $\deg_{\tau(t)}(g) = \deg_{\tau(t)}(g_1)$. Das ist ein Widerspruch zur Definition von s. Daraus folgt $g_0 \succeq_{\tau(s)} g_1$.

Sei nun $g_1 \in \mathrm{T}(\mathrm{in}(g, \omega))$. Mit $\vec{g_0}, \vec{g_1}$ bezeichnen wir die zugehörigen Vektoren. Da g_0 das Leitmonom ist, gilt $g_1 \prec_A g_0$. Es folgt $\omega_1\vec{g_0} \geq \omega_1\vec{g_1}$ und daher

$$\tau(s)\vec{g_0} = (1-s)\underbrace{\omega\vec{g_0}}_{=\omega\vec{g_1}} + s\underbrace{\omega_1\vec{g_0}}_{\geq\omega_1\vec{g_1}} \geq \omega_1\vec{g_1}.$$

□

5.2 Der Gröbner–Walk im kommutativen Fall

In der kommutativen Situation muß man sich über die Zulässigkeit von Matrixordnungen keine Gedanken machen, wie es in der nichtkommutativen Situation später der Fall sein wird.

Wie schon erwähnt, sind die folgenden Ausführungen wie in ([AmGlKü]) beschrieben, alle Argumente zum nichtkommutativen Fall stammen von mir.

Es gibt eine Startordnung $\prec_A$ und eine Zielordnung $\prec_B$, dargestellt durch Matrizen A resp. B.

Die erste Zeile von A sei σ, die erste Zeile von B sei τ.

Der Pfad des "Gröbner–Walk" ist die Strecke $\overline{\sigma\tau}$, (die ersten Zeilen der Matrizen spielen ja eine herausragende Rolle für die Bestimmung der Ordnung). Genau genommen werden die Matrixordnungen $\mathrm{Ord}\,(\omega, B)\,, \omega \in \overline{\sigma\tau}$ betrachtet. Bevor also die eigentliche "Wanderung" beginnt, wird von der Ordnung $\prec_A$ zur Ordnung $\mathrm{Ord}\,(\sigma, B)$ übergegangen und danach werden die Zwischenordnungen, die sich nur in der ersten Zeile der Matrizen unterscheiden, mit den ersten Zeilen $\omega \in \overline{\sigma\tau}$ aus dem Pfad identifiziert.

Diesen Pfad wird entlanggewandert, bis sich in einem der gegebenen Basiselemente der Leitterm ändert. Es stellt sich heraus, das sich dieser Punkt durch Lösen einer linearen Gleichung berechnen läßt. Praktisch handelt es sich also nicht um kontinuierliches Wandern, sondern man springt zum nächsten interessanten Punkt. So wird das Verfahren erst algorithmisch faßbar.

Auf dem Pfad ergeben sich die Zwischenschritte

$$\sigma = \omega_1, \omega_2, \ldots, \omega_m = \tau \ .$$

Die Ordnungen, die auf diesem Weg eine Rolle spielen, werden wie folgt einheitlich notiert:

$$\begin{aligned} \prec_0 &:= \prec_A = \mathrm{Ord}\,(\sigma, A) \\ \prec_1 &:= \mathrm{Ord}\,(\omega_1, B) = \mathrm{Ord}\,(\sigma, B) \\ &\vdots \\ \prec_m &:= \mathrm{Ord}\,(\omega_m, B) = \mathrm{Ord}\,(B) \end{aligned}$$

Der Pfad wird "abgearbeitet", und im Verlauf werden Gröbnerbasen zu jedem $\prec_k, k = 1, \ldots, m$ berechnet.

Im nichtkommutativen Fall stellt sich immer die Frage nach der Zulässigkeit der definierten Ordnungen. Insbesondere ist später zu untersuchen, ob bei gegebener zulässiger Start- und Zielordnung auch alle Ordnungen auf dem Pfad dazwischen zulässig sind.

5.2.1 Die Einzelschritte

Gegeben ist anfangs ein Ideal $\mathcal{I}$ aus unserer Ore–Algebra, die Startordnung $\prec_A$, zu ω_0 die Zielordnung $\prec_B$ sowie eine Gröbnerbasis $(I_0, \prec_0)$ von $\mathcal{I}$.

Nun wird der Schritt von $\prec_{k-1}, I_{k-1}, \omega_{k-1}$ zu $\prec_k, I_k, \omega_k$ beschrieben.

Dabei ist als Vorbedingung $\prec_{k-1}, I_{k-1}, \omega_{k-1}$ ein Tripel, in dem I_{k-1} eine reduzierte Gröbnerbasis bezüglich der Ordnung $\prec_{k-1}$ darstellt.

Nach dem Schritt ist I_k eine reduzierte Gröbnerbasis bezüglich $\prec_k$.

Den Algorithmus sehen wir in Kurzform in Abbildung 1. Die Schritte werden nun erläutert und die Korrektheit des Verfahrens gezeigt.

Im Schritt 2 werden die Initialpolynome $\mathrm{in}(\mathcal{I}_{k-1}, \omega_{k-1})$ in der Ordnung $\prec_{k-1}$ betrachtet.

Bemerkung 5.8 *Gegeben sei ein Ideal $\mathcal{I} = \langle g_1, \ldots, g_r \rangle$, wobei $\{g_1, \ldots, g_r\}$ eine Gröbnerbasis von $\mathcal{I}$ ist. Dann ist $\mathrm{in}(g_i, \omega_k), i = 1, \ldots, r$ zugleich eine Gröbnerbasis des Initialideales $\langle \mathrm{in}(g_1, \omega_k), \ldots, \mathrm{in}(g_r, \omega_k) \rangle$.*

Eing: Ordnungen: "$<$"=$\mathrm{Ord}\,(A)$, "$<<$"=$\mathrm{Ord}\,(B)$, $G \subset \mathcal{O}$ reduzierte Gröbnerbasis bezüglich "$<$". Der Startgewichtsvektor σ, der erste Zeilenvektor von A und der Zielgewichtsvektor ω, der erste Zeilenvektor von B.

Ausg: Modifiziertes $G \subset \mathcal{O}$, Gröbnerbasis desselben Ideales bezüglich "$<<$".

Bez: $\prec, \prec^+$ aktuelle und nächste Ordnung. ω: aktueller Gewichtsvektor.

Algo:

1	$\omega = \sigma$, $\prec = \mathrm{Ord}\,(A)$, $\prec^+ = \mathrm{Ord}\,(\sigma, B)$	Initialisieren.
2	$G_\omega = \mathrm{in}(G, \omega)$	Initialpolynome G_ω extrahieren.
3	$G_\omega^+ = \mathrm{sort}\,(G_\omega, \prec^+)$	Nach der neuen Ordnung sortieren.
4	$G_\omega^+ = \mathrm{init\_gb}(G_\omega^+, \prec^+)$	Gröbnerbasis der G_ω bestimmen
5	$G_\omega^+ = \mathrm{reduktion}(G_\omega^+, \prec^+)$	Zwischenreduktion.
6	$G = \mathrm{lift}(G_\omega^+, \prec^+, G_\omega, G, \prec)$	Liften zu einer Gröbnerbasis von G.
7	$G = \mathrm{reduktion}(G, \prec^+)$	Zwischenreduktion.
8	$\omega = \tau \Rightarrow$ Rückgabe: G	Zielordnung erreicht?
9	$t = \mathrm{Rand}(\omega, t, G)$	Nächste Grenze suchen.
	$\nexists t \Rightarrow$ Rückgabe: G	Kein weiterer Schritt nötig!
	$\omega = (1-t)\omega + t\tau$	Nächstes Grenzgewicht bestimmen.
10	$\prec = \prec^+$, $\prec^+ = \mathrm{Ord}\,(\omega, B)$, gehe nach 2.	Nächste Iteration vorbereiten.

Abbildung 1: Algorithmus "Gröbner–Walk"

Beweis: Sei $\overline{g_i} := \mathrm{in}(g_i, \omega_k)$. Weil $\overline{g_i}$ und g_i die selben Leitterme haben, gibt es Monome a, b mit $g := \mathrm{S}(g_i, g_j) = ag_i - bg_j$ sowie $\overline{g} := \mathrm{S}(\overline{g_i}, \overline{g_j}) = a\overline{g_i} - b\overline{g_j}$.

Dieses $\overline{g}$ ist ω_k–homogen. Modulo $g_1, \ldots g_r$ reduziert sich g zu 0. Insbesondere kann man von g sukzessive Vielfache $c_1 g_{i_1}, \ldots, c_s g_{i_s}$ mit Monomen c_i subtrahieren, bis alle Monmom vom selben ω_k-Grad wie $\overline{g}$ eliminiert sind. Wir erhalten $\overline{g} - \sum_{i=1}^{s} c_i \overline{g_i} = 0$ weil die ω_k–Homogenität stets erhalten beleibt.

Damit reduziert sich jedes $\mathrm{S}(\overline{g_i}, \overline{g_j})$ modulo $\{\overline{g}_1, \ldots, \overline{g}_r\}$ zu 0 und wir haben eine Gröbnerbasis. □

Bemerkung 5.9 *Mit den Bezeichungen und Voraussetzungen aus Bemerkung 5.8*

gilt $\langle \operatorname{in}(\mathcal{I}, \omega_k)\rangle = \langle \operatorname{in}(g_1, \omega_k), \ldots, \operatorname{in}(g_r, \omega_k)\rangle$.

Beweis: Für ein beliebiges $g \in \mathcal{I}$ haben wir, da $\{g_1, \ldots, g_r\}$ eine Gröbnerbasis ist, Polynome $p_1, \ldots, p_r$ mit $g = \sum_{i=1}^{r} p_i g_i$ und $p_i g_i \preceq g$. Für $\overline{g}$ ergibt sich daraus $\overline{g} = \sum_i \operatorname{in}(p_i, \omega_k) \operatorname{in}(g_i, \omega_k) \in \langle \operatorname{in}(g_1, \omega_k), \ldots, \operatorname{in}(g_r, \omega_k)\rangle$, wobei über die i mit $\deg_{\omega_k} g = \deg_{\omega_k}(p_i g_i)$ zu summieren ist. Damit ist "$\subseteq$" gezeigt, die andere Richtung ist aber klar. □

Bemerkung 5.10 *Bemerkung 5.8 ist für Ore–Algebren im Allgemeinen falsch. Ein Gegenbeispiel ist die Ore–Algebra mit den Variablen n, x und dem Shiftoperator S_n. Die Ordnung sei lexikografisch mit $n > x > S_n$ und $\mathcal{I} = \langle nx, (n+1)S_n\rangle$.*

Beweis: Die Erzeugenden bilden eine Gröbnerbasis, denn es ist $\mathrm{S}(nx, (n+1)S_n) = S_n(nx) - x((n+1)S_n) = (n+1)xS_n - (n+1)xS_n = 0$. Für die Initialpolynome gilt jedoch $\mathrm{S}(nx, nS_n) = S_n(nx) - x(nS_n) = (n+1)xS_n - nxS_n = xS_n$, was sich modulo der Initialpolynome nicht weiter reduziert. □

Für das spätere Liften in Algebren ordnungserhaltender Art werden wir also mit der Gröbnerbasis selbst rechnen statt nur mit den Initialpolynomen. Leider ist dann nicht mehr garantiert, daß mit dem Leitterm in der vorherigen Ordnung automatisch auch der Leitterm in der neuen Ordnung im gelifteten Polynom vorkommt.

In Schritt 3 wird nach der neuen Ordnung sortiert.

Schritte 4 und 5 sind die Bestimmung der reduzierten Gröbnerbasis G_ω^+ des Initial–Ideals. Dies ist gewöhnlich sehr einfach, da die meisten Initialformen nur aus einem Monom bestehen.

Satz 5.11 *Die mit dem Buchberger–Algorithmus gewonnene Gröbnerbasis eines von ω–homogenen Elementen erzeugten Ideales besteht wieder aus ω–homogenen Elementen. Die reduzierte Gröbnerbasis besteht ebenfalls aus ω–homogenen Elementen.*

Beweis: Dies folgt aus der leicht nachzurechnenden Tatsache, daß S–Polynome ω–homogener Polynome wieder ω–homogen sind. Ebenso ist die Reduktion eines eines ω–homogenen Polynomes nach einem anderen ω–homogenen Polynom

selbst ω–homogen. □

Bemerkung 5.12 *Im Nichtkommutativen sind beide Aussagen von Satz 5.11 im Allgemeinen falsch.*

Beweis: Ein Gegenbeispiel ist die Operatoralgebra $\mathcal{D}(x, D_x; n, S_n)$ mit der lexikografischen Ordnung $n > x > D_x > S_n$ und das Ideal $\mathcal{I} = \langle xS_n, nD_x \rangle$. Das S–Polynom $\mathrm{S}(xS_n, nD_x) = nD_xxS_n - xS_nnD_x = nS_n - xS_nD_x$ wird reduziert zu $nS_n - xS_nD_x \xrightarrow{xS_n} nS_n + S_n$. Durch den Buchbergeralgorithmus kommt also genau das Element $nS_n + S_n$ hinzu.

Die entstehende Gröbnerbasis ist bereits reduziert. □

Wir werden ein Beispiel sehen, wo durch diesen Effekt die Basis der Initialpolynome kollabiert. Ohne die Homogenität muß ein S-Polynom von ω_k-Initialpolynomen in der neuen Ordnung bezüglich der alten Ordnung nicht mehr notwendig einen Lift haben.

Schritt 6 des Algorithmus schließlich erzeugt die nächste Gröbnerbasis auf dem Weg. Dazu wird jedes Polynom

$$m_i \in G_\omega^+ = \{m_1, \ldots, m_s\} \tag{6}$$

durch Polynomdivision mit der Gröbnerbasis $\mathrm{in}(G_\omega, \prec)$ dargestellt als Summe

$$m_i = \sum_{j=1}^{r} h_{ij} \,\mathrm{in}(g_j, \omega_k) \tag{7}$$

mit g_j aus I_{k-1} und $h_{ij}\,\mathrm{in}(g_j, \omega_k) \preceq m_i$. Das Liften besteht nun darin, jedes $\mathrm{in}(g_j, \omega_k)$ in der Darstellung aus (7) der m_i aus (6) durch g_j selbst zu ersetzen und so $f_i = \sum_{j=1}^{r} h_{ij} g_j$ zu erhalten. Damit erhalten wir

$$F := \{f_1, \ldots, f_s\}. \tag{8}$$

Nun ist zu zeigen, daß F eine Gröbnerbasis von $\mathcal{I}$ bezüglich $\prec$ ist.

Hierfür zeigen wir zunächst einige einfache Aussagen:

Satz 5.13 *Sei $\mathcal{I} \subset \mathcal{D}$ ein Ideal einer Ore–Algebra. $\mathcal{D}$ sei ordnungserhaltend bezüglich der Monomordnung $\prec$. Ist dann $F \subset \mathcal{I}$ eine endliche Teilmenge, und gilt $\mathrm{LM}(\mathcal{I}) \subseteq \mathrm{LM}(\langle \mathrm{LM}(F) \rangle)$ so folgt $\langle F \rangle = \mathcal{I}$.*

Beweis: Sofort folgt $\langle F \rangle \subseteq \mathcal{I}$ aus $F \subset \mathcal{I}$.

Der Beweis von $\mathcal{I} \subset \langle F \rangle$ folgt mit noetherscher Induktion. Es bezeichne $\preceq_*$ die durch $\prec$ induzierte Halbordnung auf $\mathcal{D}$. Dann erfüllt $\preceq_*$ die absteigende Kettenbedingung, da $\prec$ eine totale Ordnung ist: Zu einem $g \in \mathcal{D}$ gibt es nur endlich viele Monome $m \prec \mathrm{LM}(g)$ daher muß jede fallende Kette $g \succ_* g_1 \ldots \succ_* g_s \ldots$ abbrechen.

Nun wird zu $g \in \mathcal{I}$ die Aussage $g \in \langle F \rangle$ auf $g' \in \langle F \rangle$ für ein $g' \prec_* g$ zurückgeführt:

Sei $g \in \mathcal{I}$. Dann gibt es nach Voraussetzung $p_1, \ldots, p_n \in \mathcal{D}$, $f_1, \ldots, f_n \in F$ mit $\mathrm{LM}(g) = \mathrm{LM}\left(\sum_{i=1}^{n} p_i \mathrm{LM}(f_i)\right)$. Nun muß sich $\mathrm{LM}(g)$ in einem der Summanden, etwa $p_i \mathrm{LM}(f_i)$ finden, also auch in $a\mathrm{LM}(f_i)$ für ein Monom $a \in \mathrm{M}(p_i)$. Wir erhalten $\mathrm{LT}(g) = \mathrm{LT}\left(\frac{\mathrm{LC}(g)}{\mathrm{LC}(af_i)} a f_i\right)$ und es folgt $g = \frac{\mathrm{LC}(g)}{\mathrm{LC}(af_i)} a f_i + g', g' \prec_* g$. Der linke Summand ist aus $\langle F \rangle$ und g' nach Induktionsannahme. □

Bemerkung 5.14 *Die Ordnung $\prec_{k-1}$ verfeinert die Halbordnung $\prec_{\omega_k}$.*

Dies zeigen wir, wenn die Bestimmung von ω_k genauer betrachtet wird. Für $k = 1$ ist die Aussage klar, da ω_1 die Leitform von $\prec_A$ ist.

Satz 5.15 *Es gilt* $\mathrm{LT}(m_i) = \mathrm{LT}(f_i), i = 1, \ldots, s$

Beweis: Es ist $f_i - m_i = \sum_{j=1}^{r} (h_{ij} g_j - h_{ij} \,\mathrm{in}(g_j, \omega_k))$. Für jeden Summanden gilt mit Satz 5.5 $h_{ij} g_j - h_{ij} \,\mathrm{in}(g_j, \omega_k) \prec_{\omega_k} m_i$. Da ω_k die Leitform der neuen Ordnung ist, erhalten wir $\mathrm{LM}(f_i - m_i) \prec_{\omega_k} \mathrm{LM}(m_i)$ was bedeutet, daß f_i und m_i das gleiche Leitmonom haben müssen. □

Satz 5.16 *F ist eine Gröbnerbasis von $\mathcal{I}$ bezüglich $\prec$.*

Beweis: Es ist $F \subseteq \mathcal{I}$. Wenn $\mathrm{LM}(\mathcal{I}) \subseteq \mathrm{LM}(\langle \mathrm{LM}(F) \rangle)$ gezeigt werden kann, erzeugt F mit Satz 5.13 auch $\mathcal{I}$ und ist eine Gröbnerbasis. Sei $M = G_\omega^+$.

Wir haben

$$\mathrm{LM}(\mathcal{I}) =$$
$$\mathrm{LM}(\,\mathrm{in}(\mathcal{I}, \omega_k)) \subseteq \mathrm{LM}(\langle\, \mathrm{in}(\mathcal{I}, \omega_k)\rangle) \overset{(*)}{=} \mathrm{LM}(\langle \mathrm{LM}(\langle M\rangle\,)\rangle\,) \overset{(\circ)}{=} \mathrm{LM}(\langle \mathrm{LM}(M)\rangle\,).$$

Wobei $(*)$ Bemerkung 5.9 ist, und $(\circ)$ eine Eigenschaft von Gröbnerbasen.

Mit Satz 5.15 ist $\langle \mathrm{LM}(M)\rangle = \langle \mathrm{LM}(F)\rangle$, und wir erhalten $\mathrm{LM}(\mathcal{I}) \subseteq \mathrm{LM}(\langle \mathrm{LM}(F)\rangle)$ □

Im folgenden Schritt 7, der Zwischenreduktion, erhalten wir also eine reduzierte Gröbnerbasis I_k von $\mathcal{I}$ bezüglich $\prec_k$.

Wenn nun die Zielordnung erreicht ist, sind wir fertig.

Ansonsten folgt die Berechnung der nächsten Ordnung. Hierfür muß der nächste Gewichtsvektor auf der Strecke von ω_k bis τ gefunden werden, für den Initialformen der reduzierten Gröbnerbasis I_k degenerieren. Um einen solchen Wechsel der Initialformen zu erkennen, parametrisieren wir die Gewichtsvektoren auf der Strecke $\overline{\omega_k \tau}$ durch

$$\omega(s) := \omega_k + s(\tau - \omega_k), 0 < s \leq 1. \tag{9}$$

Wir stellen die Elemente von I_k in der Form $\left\{ g = \sum_{i=1}^{m} p_{g_i} \right\}$ als Summe von Termen dar. Dann bezeichnet $d_{g_i}(s) := \deg_{\omega(s)}(p_{g_i})$ einen linearen Ausdruck in s.

$$t = \min\left(\{ s \,|\, d_{g1}(s) = d_{gi}(s), d_{g1}(0) \neq d_{gi}(0), g \in I_k \}\right) \cap [0, 1] \tag{10}$$

Für diese Berechnung braucht man pro Monom ein Skalarprodukt in $\mathbb{Q}^n$ und eine rationale Division, um die Gleichungen mit der Parametrisierung (9) nach s aufzulösen. Wenn also t existiert, ist es eine positive rationale Zahl. Ist $t = 1$, so wird der nächste Schritt der letzte sein.

Nun kennen wir das neue Gewicht ω_{k+1} und können den Beweis von Bemerkung 5.14 für $k > 1$ nachtragen:

Beweis: von Bemerkung 5.14. Zu zeigen ist: $\prec_{\omega_{k+1}}$ wird durch $\prec_k$ verfeinert.

Diese Bemerkung ist in [AmGlKü] nur erwähnt und nicht ausgeführt. In [CoKaMa] wird mit "Gröbner cones" argumentiert, was hier vermieden werden soll. Zum Beweis reicht die Beobachtung, daß nach Konstruktion bezüglich aller Gewichte $\omega \in \overline{\omega_k\omega_{k+1}}$ stets das Leitmonom bezüglich $\prec_k$ unter den Initialmonomen ist: $\mathrm{LM}_{\prec_k}(g) \in \mathrm{M}(\,\mathrm{in}(g,\omega))$.

Die gefundene neue Leitform stößt an die Grenze, wo eines der bisher durch $\prec_{\omega_k}$ dominierten Monome Kandidat für das bisherige Leitmonom wird.

Verfeinert man diese Grenzform durch die alte Ordnung, ergeben sich somit die ursprünglichen Leitmonome. Also wird $\prec_{\omega_{k+1}}$ durch $\prec_k$ verfeinert. □

5.2.2 Eine Beispielrechnung

In diesem Abschnitt sollte ein Ausdruck $\{a, b, c, \ldots\}$ als Liste aufgefaßt werden, denn manchmal spielt die Reihenfolge der Einträge eine Rolle.

Das Ideal $\mathcal{I}$ im folgenden Beispiel stammt aus Kap. 3.3 in [Roda]. Er listet dort die möglichen Gröbnerbasen von $\mathcal{I}$ und die dazu führenden Ordnungen auf.

Satz 5.17 *Gegeben sei der Polynomring $\mathbb{Q}[x,y,z]$ mit der gradlexikografischen Ordnung (mit $x > y > z$).*

Dann sind die Elemente $g_1 = x^3 + xz - 2z, g_2 = y^3 - xz$ eine reduzierte Gröbnerbasis von $\mathcal{I} = \langle x^3 + xz - 2z, y^3 - xz\rangle$.

Beweis: Das S-Polynom wird mit $\{g_1, g_2\}$ reduziert:

$$\begin{aligned}
\mathrm{S}(g_i, g_2) \quad &= \quad y^3 g_1 - x^3 g_2 = xy^3z - 2y^3z + x^4z \\
&\overset{y^3-xz}{\longrightarrow} \quad -2y^3z + x^2z^2 + x^4z \\
&\overset{y^3-xz}{\longrightarrow} \quad x^4z - 2xz^2 + x^2z^2 \\
&\overset{x^3+xz-2z}{\longrightarrow} \quad -xz(x^3 + xz - 2z) + x^4z - 2xz^2 + x^2z^2 = 0.
\end{aligned}$$

Da sich das einzige S-Polynom zu Null reduziert, liegt eine Gröbnerbasis vor. Offensichtlich teilt aus Gradgründen keiner der Leitterme einen der anderen in g_1, g_2 auftauchenden Terme und somit ist die Basis reduziert. □

Gesucht ist eine Gröbnerbasis bezüglich der durch die Matrix $\begin{pmatrix} 3 & 2 & 6 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ gegebenen Ordnung. Die Basis bezüglich der Zielordnung wird drei Elemente haben. Anfangs wird der Algorithmus recht ausführlich abgearbeitet, später werden wir einige abkürzende Überlegungen benutzen.

Unsere Ausgangslage (Schritt 1) ist:

$$\begin{aligned} A &= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \\ B &= \begin{pmatrix} 3 & 2 & 6 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \\ \omega &= \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \\ \tau &= \begin{pmatrix} 3 & 2 & 6 \end{pmatrix} \\ \prec^+ &= \operatorname{Ord}(\omega, B) = \operatorname{Ord} \begin{pmatrix} 1 & 1 & 1 \\ 3 & 2 & 6 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Dabei ist die Ordnung $\prec^+$ durch die ersten Zeilen von B bestimmt, da die letzte Zeile der Zielordnungsmatrix von den $\prec^+$ definierenden Zeilen linear abhängig ist.

Im Verlauf des Gröbner–Walks kann es passieren, daß zur Definition von $\prec^+$ auch die letzte Zeile der Zielordnung mit herangezogen werden muß, falls die ersten beiden mit dem aktuellen Gewicht eine singuläre Matrix bilden.

In Schritt 2 des Verfahrens werden nun die Initialpolynome extrahiert. Das sind hier gerade die Monome $G_\omega = \{x^3, y^3\}$. Dies ist bezüglich $\prec^+$ bereits sortiert und offenbar (als Menge von Monomen) auch eine reduzierte Gröbnerbasis (Schritte 3,4,5). Auch Schritt 6, das Liften, ist hier trivial. Nachdem nichts verändert wurde, erhalten wir die alte Basis zurück:

$G = \operatorname{lift}(G^+_\omega, \prec^+, G_\omega, G, \prec) = \{y^3 - xz, x^3 + xz - 2z\}$.

Die Zielordnung ist offenbar noch nicht erreicht (Schritt 8). Nun kommen wir zur Randberechnung, dem Schritt 9:

Wir haben die Teilmonome $\{\{x^3, xz, z\}, \{y^3, xz\}\}$, hier fallend sortiert nach ω–Gewicht. Diese entsprechen den Mengen von Vektoren

$$T_1 = \left\{ \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}, T_2 = \left\{ \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

Nun suchen wir das kleinste t, für das sich ein Leitterm durch Übergang zu $v_t = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} + t(\begin{pmatrix} 3 & 2 & 6 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}) = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} + t(\begin{pmatrix} 2 & 1 & 5 \end{pmatrix})$, $t \in (0, 1]$ ändert. Da der Leitterm im generischen Fall, wenn t nicht auf dem Rand liegt, ein Monom ist, sagen wir, das Polynom "degeneriert" . Das bedeutet hier

für ein $w \in T_1 \setminus \left\{ \begin{pmatrix} 3 & 0 & 0 \end{pmatrix} \right\}$ gilt $v_t \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix} = v_t w$ oder

für ein $w \in T_2 \setminus \left\{ \begin{pmatrix} 0 & 3 & 0 \end{pmatrix} \right\}$ gilt $v_t \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix} = v_t w$.

Für T_1 berechnen wir $v_t \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix} = 3+6t$ und $v_t \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 2+7t, v_t \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 1+5t$, wir erhalten die Gleichungen $3 + 6t = 2 + 7t, 3 + 6t = 1 + 5t$, also die Lösungen $\left\{1, -\frac{1}{2}\right\}$. Der Leitterm degeneriert also nur für die Zielordnung.

Für T_2 ist $v_t \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix} = 3+3t$ und für den einzigen Kandidaten für w ist $v_t \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 2+7t$, es ergibt sich die Gleichung $3+3t = 2+7t$, also $t = \frac{1}{4}$. Von den Lösungen kommen nur $\frac{1}{4}, 1$ in Frage, das Minimum ist $t = \frac{1}{4}$.

Der nächste potentielle Verzweigungspunkt ist daher bei dem Gewichtsvektor $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 2 & 1 & 5 \end{pmatrix} = \begin{pmatrix} \frac{6}{4} & \frac{5}{4} & \frac{9}{4} \end{pmatrix}$. Damit sind wir bei Schritt 10, das alte $\prec^+$ wird die Startordnung, und $\prec^+$ wird durch das neue Gewicht, verfeinert durch die Zielordnung, definiert:

$$A := \begin{pmatrix} 1 & 1 & 1 \\ 3 & 2 & 6 \\ 0 & 0 & 1 \end{pmatrix}, \omega := \begin{pmatrix} \frac{6}{4} & \frac{5}{4} & \frac{9}{4} \end{pmatrix}, \prec^+ := \mathrm{Ord} \begin{pmatrix} \frac{6}{4} & \frac{5}{4} & \frac{9}{4} \\ 3 & 2 & 6 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Hier sind die ersten drei Zeilen linear unabhängig, die vierte ist also für die Ordnung redundant:

$\prec^+ = \mathrm{Ord}\begin{pmatrix} \frac{6}{4} & \frac{5}{4} & \frac{9}{4} \\ 3 & 2 & 6 \\ 0 & 0 & 1 \end{pmatrix}$. Die Initialterme der Basiselemente sind nun keine Monome mehr und $y^3 - xz$ ist ω–homogen (so wurde ja das neue Gewicht gewählt), wobei die zweite Zeile der neuen Ordnung entscheidet, daß $y^3 \prec^+ xz$ gilt. Die neuen Initialterme sind somit:

$$G_\omega = \left\{x^3, -xz + y^3\right\} .$$

Dies ist nach der neuen Ordnung bereits sortiert. Nun bestimmen wir die Gröbnerbasis nach Buchberger:

Als erstes S-Polynom erhalten wir x^2y^3, welches, da kein z auftaucht und der Grad in x zu niedrig ist, nicht reduziert werden kann.

Im kommutativen Fall verschwindet jedenfalls das S-Polynom zweier Monome. Damit kommen sukzessive $\mathrm{S}(x^2y^3, -xz + y^3) = xy^6$ und $\mathrm{S}(xy^6, -xz + y^3) = y^9$ hinzu.

Wir erhalten damit folgende Gröbnerbasis der Initialpolynome:

$G_\omega = \{-xz + y^3, x^3, x^2y^3, xy^6, y^9\}$.

Der Prozeß des Liftens kann nun durch Division durch die Menge von Polynomen G stattfinden. Wir werden hier allerdings die Bildung der S-Polynome einfach mit den ursprünglichen Basiselementen statt der Initialformen nachvollziehen (was im Wesentlichen dasselbe ist):

$$\begin{aligned} -xz + y^3 &\leftarrow t_1 = -xz + y^3 \\ x^3 &\leftarrow t_2 = x^3 + xz - 2z \\ x^2y^3 &\leftarrow t_3 = zt_2 + x^2t_1 = x^2y^3 + xz^2 - 2z^2 \\ xy^6 &\leftarrow t_4 = zt_3 + xy^3t_1 = xy^6 + xz^3 - 2z^3 \\ y^9 &\leftarrow t_5 = zt_4 + y^6t_1 = y^9 + xz^4 - 2z^4 \end{aligned}$$

Das Resultat nun noch einmal nach aufsteigenden Leitmonomen und mit geordnet sortierten Polynomdarstellungen aufgeschrieben:

$G =$
$\{-xz + y^3, x^3 + xz - 2z, x^2y^3 + xz^2 - 2z^2, xy^6 + xz^3 - 2z^3, y^9 + xz^4 - 2z^4\}$

Reduktion dieser Basis führt (via $xz \to y^3$) auf:

$G =$
$\{-xz + y^3, x^3 + y^3 - 2z, x^2y^3 + zy^3 - 2z^2, xy^6 + y^3z^2 - 2z^3, y^9 + z^3y^3 - 2z^4\}$.

Wegen $t \neq 1$ sind wir noch nicht fertig und gelangen wieder zur Randberechnung.

Von den ersten beiden Polynomen wissen wir, daß sie keinen weiteren Wert für t liefern werden. Bleiben die drei neuen Basiselemente zu betrachten:

$$T_3 = \left\{ \begin{pmatrix} 2 \\ 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \right\}, T_4 = \left\{ \begin{pmatrix} 1 \\ 6 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 3 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 3 \end{pmatrix} \right\},$$

$$T_5 = \left\{ \begin{pmatrix} 0 \\ 9 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 4 \end{pmatrix} \right\}.$$

Nun ist

$v_t = \begin{pmatrix} \frac{6}{4} & \frac{5}{4} & \frac{9}{4} \end{pmatrix} + t(\begin{pmatrix} 3 & 2 & 6 \end{pmatrix} - \begin{pmatrix} \frac{6}{4} & \frac{5}{4} & \frac{9}{4} \end{pmatrix}) = \frac{1}{4}(\begin{pmatrix} 6 & 5 & 9 \end{pmatrix} + t \begin{pmatrix} 6 & 3 & 15 \end{pmatrix})$,
und wir erhalten die Gleichungen

$$\begin{aligned} 3 - 3t = 0\,, 9 - 9t = 0 &\Rightarrow t = 1 \\ 3 - 15t = 0\,, 9 - 21t = 0 &\Rightarrow t \in \left\{ \frac{1}{5}, \frac{3}{7} \right\} \\ 3 - 27t = 0\,, -9 + 33t = 0 &\Rightarrow t \in \left\{ \frac{1}{9}, \frac{3}{11} \right\}. \end{aligned}$$

Also degeneriert $xy^6 + y^3z^2 - 2z^3$ beim Gewichtsvektor $\frac{1}{4}(\begin{pmatrix} 6 & 5 & 9 \end{pmatrix} + \frac{1}{5} \begin{pmatrix} 6 & 3 & 15 \end{pmatrix}) = \frac{1}{5} \begin{pmatrix} 9 & 7 & 15 \end{pmatrix}$. Aber $y^9 + z^3y^3 - 2z^4$ degeneriert bereits bei $t = \frac{1}{9}$ unter dem Gewichtsvektor $\frac{1}{4}(\begin{pmatrix} 6 & 5 & 9 \end{pmatrix} + \frac{1}{9} \begin{pmatrix} 6 & 3 & 15 \end{pmatrix}) = \frac{1}{3} \begin{pmatrix} 5 & 4 & 8 \end{pmatrix}$.

Hiermit werden die Daten für die “dritte Runde” aktualisiert:

$$A := \begin{pmatrix} \frac{6}{4} & \frac{5}{4} & \frac{9}{4} \\ 3 & 2 & 6 \\ 0 & 0 & 1 \end{pmatrix}, \omega := \begin{pmatrix} \frac{5}{3} & \frac{4}{3} & \frac{8}{3} \end{pmatrix}, \prec^+ := \mathrm{Ord} \begin{pmatrix} \frac{5}{3} & \frac{4}{3} & \frac{8}{3} \\ 3 & 2 & 6 \\ 0 & 0 & 1 \end{pmatrix}.$$

Nun schreibt sich G als

$\{-xz+y^3, x^3+y^3-2z, x^2y^3+zy^3-2z^2, xy^6+y^3z^2-2z^3, z^3y^3+y^9-2z^4\}$, inzwischen hat z^3y^3 das Monom y^9 als Leitmonom "überholt". Wir erhalten als Initialterme (wobei nach Wahl von ω der letzte degeneriert)

$G_\omega = \{-xz, x^3, x^2y^3, xy^6, z^3y^3+y^9\}$.

Zur Bestimmung der Gröbnerbasis kommen nur die S-Polynome mit dem Nicht–Monom in Betracht. Das ist die Menge $\{xy^9, x^3z^3y^9, x^2y^9, xy^3\}$. Alle Elemente sind Vielfache von Monomen aus G_ω, reduzieren sich sich also zu 0. G_ω ist daher eine Gröbnerbasis bezüglich $\prec^+$ und G bleibt wie zuvor.

Der Leitterm von $z^3y^3+y^9-2z^4$ hat sich verändert. Der entsprechende Randwert ergibt sich aus den Gleichungen

$$v_t\left(\begin{pmatrix}0\\3\\3\end{pmatrix}-\begin{pmatrix}0\\9\\0\end{pmatrix}\right)=0, v_t\left(\begin{pmatrix}0\\3\\3\end{pmatrix}-\begin{pmatrix}0\\0\\4\end{pmatrix}\right)=0,$$

mit $v_t = \omega + t(\begin{pmatrix}3 & 2 & 6\end{pmatrix}-\omega) = \begin{pmatrix}\frac{5}{3} & \frac{4}{3} & \frac{8}{3}\end{pmatrix} + t\begin{pmatrix}\frac{4}{3} & \frac{2}{3} & \frac{10}{3}\end{pmatrix}$ ergeben sich die Lösungen $t=0, t=1$, dieses Polynom degeneriert also erst in der Zielordnung.

Damit können wir sofort zum nächsten, zuvor für $xy^6+y^3z^2-2z^3$ berechneten, Gewicht $\omega = \frac{1}{5}\begin{pmatrix}9 & 7 & 15\end{pmatrix}$ übergehen. Wir erhalten

$$A := \begin{pmatrix}\frac{5}{3} & \frac{4}{3} & \frac{8}{3}\\ 3 & 2 & 6\\ 0 & 0 & 1\end{pmatrix}, \omega := \begin{pmatrix}\frac{9}{5} & \frac{7}{5} & \frac{15}{5}\end{pmatrix}, \prec^+ := \mathrm{Ord}\begin{pmatrix}\frac{9}{5} & \frac{7}{5} & \frac{15}{5}\\ 3 & 2 & 6\\ 0 & 0 & 1\end{pmatrix}.$$

Mit der neuen Ordnung schreibt sich G als

$\{-xz+y^3, x^3+y^3-2z, x^2y^3+zy^3-2z^2, y^3z^2+xy^6-2z^3, z^3y^3+y^9-2z^4\}$. Die Leitmonome sind nun

$G_\omega = \{-xz, x^3, x^2y^3, y^3z^2+xy^6, z^3y^3\}$.

Im Reduktionsschritt verkleinert sich G_ω um das Element z^3y^3, das mittels $y^3z^2+xy^6$ zu $-xzy^6$ und dann vermöge $-xz$ zu 0 reduziert wird. Die Menge der S-Polynome mit dem einzigen Nicht-Monom ist nun $\{y^3z^3, x^2y^3z^2, xy^3z^2\}$, alle Terme werden zu Null reduziert.

Damit ist das Liften wieder trivial, es fällt nur das letzte Basiselement weg:

$G = \{-xz+y^3, x^3+y^3-2z, x^2y^3+zy^3-2z^2, y^3z^2+xy^6-2z^3\}$.

Der Leitterm von $x^2y^3+zy^3-2z^2$ ist gleich geblieben, aber der von $y^3z^2+xy^6-2z^3$ hat sich verändert. Damit bleibt nur noch für dieses Polynom das nächste Gewicht zu bestimmen. Wir haben $T_4 = \left\{ \begin{pmatrix} 0 \\ 3 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 6 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 3 \end{pmatrix} \right\}$. Hier findet man für t die Gleichungen $3t = 0, \frac{6}{5} - \frac{6}{5}t = 0$, also sind wir im nächsten Schritt am Ziel angelangt:

$$A := \begin{pmatrix} \frac{9}{5} & \frac{7}{5} & \frac{15}{5} \\ 3 & 2 & 6 \\ 0 & 0 & 1 \end{pmatrix}, \omega := \begin{pmatrix} 3 & 2 & 6 \end{pmatrix}, \prec^+ := \mathrm{Ord} \begin{pmatrix} 3 & 2 & 6 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

Hier tritt erstmals der Fall ein, daß die dritte Zeile der Zielmatrix herangezogen wird, dies mag in anderen Fällen aber auch zwischendurch passieren. Nun ist

$$G = \{-xz+y^3, x^3-2z+y^3, -2z^2+zy^3+x^2y^3, -2z^3+y^3z^2+xy^6\}.$$

Wir befinden uns an einem Schnittpunkt von "Gröbnercones", es degenerieren in der Zielordnung gleich mehrere Basispolynome!

Allerdings ist diese Basis noch nicht reduziert, der Leitterm des letzten Polynoms läßt sich mit dem zweiten Polynom eliminieren, es bleibt $xy^6 - x^2y^3z$, was sich wiederum zu $xy^6 - xy^6 = 0$ reduziert. Damit bleibt

$$G = \{x^3-2z+y^3, -xz+y^3, -2z^2+x^2y^3+zy^3\}.$$

Wir erhalten für die Initialpolynome:

$$G_\omega = \{x^3\ , -xz\ , -2z^2+x^2y^3+zy^3\}$$

Es gibt nur zwei S–Polynome zu bilden:

$$-2z^2(-xz)+x(-2z^2+x^2y^3+zy^3) = x^3y^3+xy^3z \xrightarrow{xz} x^3y^3 \xrightarrow{x^3} 0$$

und

$$2z^2(x^3)+x^3(-2z^2+x^2y^3+zy^3) = x^3y^3+x^3y^3z \xrightarrow{xz} x^3y^3 \xrightarrow{x^3} 0.$$

Da wir bei der Zielordnung angelangt sind, kommen wir zum letzten Liftungsschritt, der wieder trivial ist und uns zur Gröbnerbasis bezüglich der Zielordnung führt:

$$G = \left\{x^3-2z+y^3, -xz+y^3, -2z^2+zy^3+x^2y^3\right\}.$$

6 Gröbner–Walk für Algebren ordnungserhaltender Art

Der Gröbner–Walk läßt sich recht einfach implementieren. Nach der Implementation des Algorithmus mit dem vorgestellten Paket stellte ich fest, daß die Konversion in den getesteten Beispielen funktionierte.

Ein wichtiger Punkt in den Argumentationen aus dem kommutativen Fall ist, daß $\mathrm{LM}(p \cdot q) = \mathrm{LM}(\mathrm{LM}(p) \cdot \mathrm{LM}(q))$ gilt[6] . Diese Eigenschaft kennzeichnet Algebren ordnungserhaltender Art.

Dies motivierte zu untersuchen, ob der Gröbner–Walk auf Ore–Algebren ordnungserhaltender Art anwendbar ist, und war der Anfangspunkt zu den bereits vorgestellten Überlegungen.

So führten Experimente mit meinem Paket zur Betrachtung des Gröbner–Walks für Ore–Algebren ordnungserhaltender Art.

6.1 Klippen der Adaption auf den Ore–Fall

Für eine gegebene Ore–Algebra muß man sich fragen, ob eine Matrixordnung überhaupt zulässig ist. Beim Gröbner–Walk sind zwei Monomordnungen gegeben. Die Zulässigkeit beider Ordnungen für die gegebene Ore–Algebra wird natürlich angenommen.

Doch stellt sich nicht nur die Frage, ob Start- und Zielordnung zulässig sind, sondern auch, wie es mit den auftretenden Zwischenordnungen aussieht. Dies wird im folgenden Abschnitt mit positivem Ergebnis geklärt.

Ein Problem ist die Zerstörung der Homogenität nach Bilden des S-Polynomes. Dies hat einige Konsequenzen.

Im Schritt 4 sichert Satz 5.11, daß die Gröbnerbasis der Initialmonome aus homogenen Elementen bezüglich der Leitform besteht und diese Homogenität auch bei Reduktion der Basis im Schritt 5 erhalten bleibt.

Im nichtkommutativen Fall gilt dieser Satz nicht, wir sahen das Beispiel 5.12.

[6] Im Kommutativen ist natürlich sogar $\mathrm{LM}(\mathrm{LM}(p) \cdot \mathrm{LM}(q)) = \mathrm{LM}(p) \cdot \mathrm{LM}(q)$.

Der Algorithmus von Buchberger generiert eventuell aus den Leitformen Elemente, deren Leitterme in der Gröbnerbasis der ursprünglichen Erzeugenden nicht als Leitterme vorkommen. Diese zusätzlichen Elemente lassen sich später natürlich nicht liften.

Solche Elemente können aber auch nicht zu einem Leitmonom des Ideales beitragen. Diese Elemente können aber schädlich sein, wenn die Gröbnerbasis der Leitformen reduziert wird. Dann werden eventuell korrekt gefundene neue Leitterme eliminiert.

Dies wurde durch simplen Verzicht auf den Reduktionsschritt 5 verhindert. Wir werden in einem Gegenbeispiel sehen, daß auch das nicht reicht, da in der Berechnung der Gröbnerbasis nach Buchberger implizit eine Reduktion nach allen bereits gefundenen Elementen steckt.

6.2 Voraussetzungen für den Gröbner–Walk

Wir betrachten eine Ore–Algebra $\mathcal{D} = k[X_1, \ldots, X_m] \langle D_i, \sigma_i, \delta_i \, , i = 1, \ldots, n \rangle$. Die Eigenschaft "ordnungserhaltend" hängt nun von der gegebenen Monomordnung $<$ ab; übersetzt in unsere Notation besagt sie für alle

$D_j X_i = \sigma_j(X_i) D_j + \delta_j(X_i)$:

$$\mathrm{LM}\left(\sigma_j(X_i)\right) \in k^* \cdot X_i, \quad \text{und} \quad \mathrm{LM}\left(\sigma_j(X_i)\right) \cdot D_j > \mathrm{LM}\left(\delta_j(X_i)\right) \ .$$

Eine Grundvoraussetzung dafür, daß die Argumentationen für das Funktionieren des Gröbner–Walk übernommen werden können, ist natürlich, daß die Voraussetzung "ordnungserhaltend" bei den betrachtenden Monomordnungen erhalten bleibt.

Seien also $<, <<$ Matrixordnungen und A, B die zugeordneten Matrizen, so ist zunächst zu zeigen, daß die Verfeinerung der durch die erste Zeile von A gegebenen Teilordnung durch B ordnungserhaltender Art ist:

Satz 6.1 *Eine Ore–Algebra $\mathcal{D}$ sei bezüglich der Matrixordnungen zu den Matrizen A, B ordnungserhaltender Art. Dann ist auch die Ordnung, gebildet aus der Leitform von A, verfeinert durch die Ordnung B ordnungserhaltender Art.*

Beweis: Ordnungserhaltend heißt, daß für Unbestimmte X, D stets

$$DX = \alpha X D + P$$

mit $\mathrm{LM}\,(P) < XD, \alpha \in k^*$ gilt.

Es entspreche XD dem Ordnungsvektor $u \in I\!R^n$ und die Terme von P der endlichen Menge von Ordnungsvektoren $R \subset I\!R^n$.

Die erste Zeile a von A bezeichnet eine Linearform. Die durch B verfeinerte Ordnung entspricht damit der Matrixordnung C, die entsteht, wenn B zusätzlich die Zeile a vorangestellt wird. Weil $\mathcal{D}$ bezüglich $\mathrm{Ord}\,(A)$ ordnungserhaltender Art ist, folgt $\forall r \in R : ar \leq au$. Im Falle der Gleichheit aber ergibt sich aus der Voraussetzung für B, daß $\forall r \in R : Br <_{\mathrm{lex}} Bu$. Damit folgt insgesamt $\mathrm{LM}_C(P) <_C XD$, was zu zeigen war. □

Dieses Resultat zeigt, daß die erste neue Ordnung beim Gröbner–Walk, die Verfeinerung A' der Leitform von A durch B, die Eigenschaft ordnungserhaltender Art zu sein erhält.

Ersetzen wir nun A durch A', so sind die weiteren Zwischenstationen Linearkombination der Ausgangsmatrizen.

Satz 6.2 *Eine Ore–Algebra $\mathcal{D}$ sei bezüglich der Matrixordnungen zu den Matrizen A, B ordnungserhaltender Art. Dann ist sie auch auf den zwischen diesen Ordnungen liegenden Matrixordnungen $C_\tau, \tau \in [0,1]$ ordnungserhaltender Art.*

Beweis: Seien wieder X, D Unbestimmte und P so, daß $DX = \alpha XD + P$, wobei $\mathrm{LM}\,(P) < XD, \alpha \in k^*$ bezüglich der Zwischenordnungen zu zeigen ist. Die ersten Zeilen a, b von A resp. B sind Linearformen. Es entspreche XD dem Ordnungsvektor $u \in I\!R^n$ und die Terme von P der endlichen Menge von Ordnungsvektoren $R \subset I\!R^n$.

Weil $\mathcal{D}$ bezüglich beider Ordnungen ordnungserhaltender Art ist, gilt $au \geq \max\limits_{r \in R}(ar)$ sowie $bu \geq \max\limits_{r \in R}(br)$.

Ist nun $c = (1-\tau)a + \tau b$ so folgt für $\tau \in [0,1]$ sofort $cu = (1-\tau)au + \tau bu \geq (1-\tau)ar + \tau br = cr \forall r \in R$. □

Es ergibt sich sofort aus (6.1) und (6.2):

Korollar 6.3 *Eine Ore–Algebra $\mathcal{D}$ sei bezüglich der Matrixordnungen zu den Matrizen A, B ordnungserhaltender Art.*

Dann ist sie auch bezüglich aller im Gröbner–Walk auftauchenden Zwischenordnungen ordnungserhaltender Art.

Eine weitere Bedingung ist das Erhalten der Zulässigkeit auf dem Pfad.

Satz 6.4 *Bezüglich einer Ore–Algebra $\mathcal{D}$ seien die Matrixordnungen zu den Matrizen A, B zulässig. Dann sind auch die auf den zwischen diesen Ordnungen liegenden Matrixordnungen $C_\tau, \tau \in [0,1]$ zulässig.*

Beweis: Dies ist nach (6.3) dann erfüllt, wenn die Zulässigkeit einer Ordnung auf $\mathcal{D}$ aus der Eigenschaft ordnungserhaltend zu sein gefolgert werden kann. In [KRW], p. 6, Lemma 1.5 wird für Algebren ordnungserhaltender Art gezeigt, daß $\mathrm{LM}(fg) = \mathrm{LM}(f) \cdot \mathrm{LM}(g)$, wobei "$\cdot$" die kommutative Multiplikation bezeichnet. Induktiv gilt das auch für Produkte mit mehreren Faktoren.

Für $P, Q \in \mathrm{M}(\mathcal{D}); R, R' \in \mathrm{M}(\mathcal{D})$ gilt bezüglich einer ordnungserhaltenden Matrixordnung: $\mathrm{LM}(RPR') = R \cdot P \cdot R'$. Es folgt für $P, Q \in \mathrm{M}(\mathcal{D}); R, R' \in \mathrm{M}(\mathcal{D}) \setminus \{0\}$,

$$P < Q \Rightarrow \mathrm{LM}(RPR') = R \cdot P \cdot R' \overset{*}{<} R \cdot Q \cdot R' = \mathrm{LM}(RQR') \ ,$$

wobei die markierte Ungleichung $\overset{*}{<}$ gilt, da die kommutative Multiplikation einer Vektoraddition der zugehörigen Ordnungsvektoren entspricht, die ordnungserhaltend ist. Damit ist aber die Zulässigkeit gezeigt. □

Bis jetzt wurde gezeigt, daß der direkte Pfad zwischen Matrixordnungen "gangbar" ist. Ein Ergebnis ist also, daß Deformationstechniken wie der Gröbner–Walk grundsätzlich möglich sind.

Als nächstes wird die naive Umsetzung des Gröbner–Walks untersucht. Dabei war das anfangs vorgestellte Paket einerseits sehr hilfreich um den Algorithmus zu implementieren aber auch um die Resultate zu analysieren.

6.3 Gröbner–Walk für Algebren ordnungserhaltender Art

Bei der Adaption des Gröbner–Walks gibt es zunächst keine wesentliche Änderung. Schritt 5, die Zwischenreduktion, wurde weggelassen und so verhindert, daß

nicht liftbare Elemente die bereits gefundenen Polynome entstellen. Der Lift wird mit der vollständigen Basis statt mit der Basis der Initialmonome durchgeführt (die ja keine Gröbnerbasis mehr sein muß).

Die Frage, ob der Lift im Schritt 6 wieder zu einer Gröbnerbasis führt, muß im allgemeinen leider negativ beantwortet werden. Interessant ist allerdings, daß es in vielen Fällen dennoch gut geht. Wie gesagt fand ich ein Gegenbeispiel erst nach erfolgreichen Läufen in einigen Beispielen.

Das gefundene Gegenbeispiel wurde noch stark vereinfacht. Diese minimale Version wird im folgenden Abschnitt diskutiert.

In der bisherigen Realisierung des Walks ist anzumerken, daß nach dem Liften stets die ursprünglichen Basiselemente noch vorhanden sind, da auf das Reduzieren im Schritt 5 verzichtet wird.

Folglich erzeugt die gefundene Basis jedenfalls noch das richtige Ideal. Ein Test, ob eine Gröbnerbasis vorliegt, sichert also zugleich, daß der vorherige Schritt geklappt hat.

So erkennt das Programm zumindest, ob es erfolgreich war — was leider recht aufwendig ist.

6.3.1 Ein Gegenbeispiel

Das hier analysierte Gegenbeispiel "lebt" in einer Ore–Algebra mit nur einem Operator, dem Shift S_n der Variablen n und einer zusätzlichen Variablen x. Wir haben also die Ore–Algebra

$$\mathcal{D} = \mathbb{Q}[x,n]\,\langle S_n, n \mapsto n+1, 0\rangle\,.$$

Start- und Zielordnung bezüglich der Variablen (S_n, n, x) seien gegeben durch die Matrizen

$$\begin{aligned} S &= \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\ Z &= \begin{pmatrix} 2 & 2 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Behauptung 6.5 *Dann ist*

$$G := \{n^2 + n, S_n + nx\}$$

eine reduzierte Gröbnerbasis bezüglich der Startordnung.

Beweis: Eine Gröbnerbasis liegt vor, wenn alle S-Polynome modulo G zu Null reduziert werden. Wir reduzieren also $\mathrm{S}(n^2 + n, S_n + nx)$ nach G:

$$\begin{array}{rcl} \mathrm{S}(n^2 + n, S_n + nx) & = & S_n(n^2 + n) - n^2(S_n + nx) \\ & = & ((n+1)^2 + n + 1)S_n - n^2 S_n - n^3 x \\ & = & 3nS_n + 2S_n - n^3 x \\ & \stackrel{S_n+nx}{\longrightarrow} & -3n^2 x - 2nx - n^3 x \\ & \stackrel{n^2+n}{\longrightarrow} & -2n^2 x - 2nx \stackrel{n^2+n}{\longrightarrow} 0. \end{array}$$

Somit haben wir eine Gröbnerbasis. Die Nicht-Leitterme der Basiselemente sind n, nx. Offenbar ist keiner davon Vielfaches eines der Leitterme S_n, n^2. Also ist die Basis reduziert. □

Die Störterme ergänzen das S-Polynom gerade so, daß der Faktor $(n^2 + n)$ abspaltet. Im kommutativen Fall würde das S-Polynom nicht zu Null reduziert und die Gröbnerbasis hätte ein Element mehr.

Der kritische Punkt für den Walk liegt hier bei $t = \frac{1}{2}$, ab der Leitform

$\frac{1}{2}\begin{pmatrix} 2 & 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 2 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix}$ beginnt sich die Überzahl von n, x in $S_n + nx$ auszuwirken. Im Polynom $n^2 + n$ ändert sich der Leitterm nicht. Damit erhalten wir als Zwischenordnung $M = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Bezüglich dieser neuen Ordnung hat nun G die Darstellung $\{n^2 + n, nx + S_n\}$. Ein Leitpolynom hat sich geändert.

Die Initialpolynome von $g_1 = n^2 + n$ und $g_2 = nx + S_n$ sind nun

$l_1 = \mathrm{in}(g_1, M) = n^2$ und $l_2 = \mathrm{in}(g_2, M) = nx + S_n$.

Das erste S-Polynom ist $s_1 = nl_2 - xl_1 = nS_n$.

Das S-Polynom $s_2' = \mathrm{S}(s_1, n^2) = \mathrm{S}(n^2, nS_n) = 2nS_n + S_n$ reduziert sich mit s_1 zu $s_2 = S_n$.

Das S-Polynom $s'_3 = \mathrm{S}(nx + S_n, s_1) = S_n n^2 - n^2 S_n = 2nS_n + S_n$ wird mit s_2 zu 0 reduziert.

Die Gröbnerbasis der Initialpolynome ist damit: $G_{in} = \{n^2, nx + S_n, nS_n, S_n\}$.

Beim Liften wird S_n zu $S_n + nx$ gehoben, es entsteht also kein neues Element.

Das Problem ist nicht das Liften von s_2, sondern daß s_2 dafür sorgt, daß s'_3 zu 0 reduziert wird!

Das fehlende Basiselement ist S_n^2, die korrekte Gröbnerbasis berechnet sich zu $\{n^2 + n, nx + S_n, nS_n + S_n, S_n^2\}$. Dies wird klarer, wenn der Buchbergeralgorithmus mit den vollen Polynomen ausgeführt wird.

Dabei wird als erstes das S-Polynom $\tilde{s}'_1 = \mathrm{S}(nx + S_n, n^2 + n) = nS_n - nx$ erzeugt, das sich zu $\tilde{s}_1 = nS_n + S_n$ reduziert.

Als nächstes wird das S-Polynom $\tilde{s}'_2 = \mathrm{S}(n^2 + n, \tilde{s}_1) = S_n(n^2 - n) - n(nS_n + Sn) = n^2 S_n + 3nS_n + 2S_n - n^2 S_n - nS_n = 2(nS_n + S_n)$ bestimmt, reduziert sich aber offenbar zu 0, es ist ja gerade $2\tilde{s}_1$.

Es ist $\mathrm{S}(nx + S_n, nS_n + S_n) = S_n^2 + nx^2 + xS_n$ $\tilde{s}'_3 = S_n^2 + x(nx + S_n)$, was zu $\tilde{s}_3 = S_n^2$ reduziert wird.

Diese direkte Gegenüberstellung der Erstellung der Gröbnerbasis für die vollständigen Polynome und der korrespondierenden Rechnungen für die Initialpolynome zeigt, warum S_n^2 nicht gefunden wird:

Die Initialpolynome generieren ein neues Element (s_2), während die entsprechenden Operationen mit den gelifteten Elementen wieder zu einem reduzierbaren Element führen. Das ist zunächst nicht so schlimm: das Resultat wird zu 0 oder zu einem redundanten Element geliftet.

Dann sorgt aber das zusätzliche Element im weiteren Verlauf dafür, daß ein Element, das in der gelifteten Rechnung tatsächlich zu einem neuen Basiselement führt, nun zu Null reduziert wird. Wir haben hier also denselben Effekt, wegen dem der Schritt 5, die Reduktion der Gröbnerbasis der Initialpolynome problematisch war. Hier findet die schädliche Reduktion aber implizit beim Abarbeiten des Buchbergeralgorithmus statt.

6.4 Ausblick

Es gibt hier zwei Ausblicke. Der eine betrifft die Weiterentwicklung der *Mathematica*–Pakete “Ore” und “OreAlgs”, der andere bezieht sich auf den Gröbner–Walk, zu dem auch ein Paket “Groebnerwalk” existiert.

6.4.1 Die *Mathematica*-Pakete

Die Pakete sind sicher eine hilfreiche Unterstützung für den geneigten Anwender. Die Benutzerschnittstelle und die Effizienz der Implementation ist allerdings noch sehr ausbaubar. Auf die Schwächen bin ich im entsprechenden Kapitel eingegangen, diese zu beseitigen wäre sicher ein erstes Ziel.

Ob aber das Grundpaket auch bei optimaler Implementierung in reinem *Mathematica*-Quelltext jemals mit den existierenden Systemen in anderen Computeralgebrasystemen oder Programmiersprachen mithalten kann, ist zu bezweifeln.

Daher wäre es denkbar die Pakete zu Schnittstellen zu Implementationen in anderen Sprachen umzubauen oder zumindest zeitkritische Rechnungen auszulagern.

Eine Fülle von Erweiterungsmöglichkeiten als Versuchsplattform steht auch offen. Insbesondere eine Einführung von Zählern für jegliche Operationen mit Ore-Objekten um hardwareunabhängige Komplexitätsuntersuchungen zu ermöglichen.

Ein berechenbares Kriterium für die Zulässigkeit einer Matrixordnung auch für nicht ordnungserhaltende Algebren wäre wünschenswert.

Bei den höheren Algorithmen könnte man untersuchen, ob nicht auch für spezielle Ideale in Algebren nicht ordnungserhaltender Art eine Gröbnerbasis gefunden werden kann.

6.4.2 Der Gröbner–Walk

Ausgehend von dem gefundenen Gegenbeispiel ist eine Abwandlung des Gröbner–Walks denkbar, bei der die störenden zusätzlichen Terme entschärft werden.

Hierzu müßte in den Buchbergeralgorithmus eingestiegen und die Reduktion nach den bisher gefundenen Basiselementen dort anders behandelt werden. So ist durchaus denkbar, daß man den Effekt aus dem Gegenbeispiel in den Griff bekommt.

Sollte das prinzipiell nicht möglich sein, kann weiterhin versucht werden, den Test, ob der Schritt geklappt hat, effizienter zu machen (bisher werden alle S-Polynome reduziert ohne irgendeine spezielle Information über die Elemente auszunutzen).

Eine weitere, hier nicht weiter verfolgte Idee ist das Untersuchen kommutativer Gröbnerbasen für die Initialmonome (also simples Fortlassen aller Störterme).

A Einführung in die Benutzung der Pakete

Die folgenden Ausführungen sind größtenteils mit *Mathematica* Version 5.1 erstellt und anschließend im TEX-Format abgespeichert worden. Der Text dürfte beweisen, daß die Unterstützung des TEX–Formates in *Mathematica* noch verbesserungsfähig ist[7].

Das Package Ore.m

Das *Mathematica*-Paket "Ore.m" dient dem Umgang mit Ore-Algebren über $\mathbb{Q}$. Es stellt etliche Operatoren als "Black Boxes" zur Verfügung, läßt aber auch beliebige Operatorvariablen unter Angabe der σ und δ zu. Eine Wirkung als Operator kann man definieren, muß es aber nicht.
Ist jedem Operator eine Wirkung zugeordnet, kann man ein Operatorpolynom auf einen Ausdruck "anwenden". Der Ausdruck muß dabei nicht notwendig ein Polynom sein.

Die Grundfunktionen sind:

- Deklaration von Operatoren,
- Addition, Multiplikation,
- Definieren verschiedener Ordnungen
- Vergleichen, Leitterme extrahieren.

Die Grundfunktionen

Wir laden das Paket und schauen die implementierten Funktionen an. *Mathematica* erlaubt es, auf einfache Weise alle Funktionen anzuzeigen (Die Datei Ore.m muß dabei für *Mathematica* auffindbar sein):

```
Needs["Ore`"]
```

Das Ergebnis des Aufrufs "`?Ore`*`" ist eine Tabelle aller implementierten Funktionen:

[7] Bis *Mathematica* Version 5.0 wurden *Mathematica*–eigene Fonts verwendet, was besser aussah. Dies wurde aufgrund der komplizierten Einbindung in Latex aufgegeben.

Ore‘

Delta
NextDegeneration
NextDegenerationVec
OMu
Operation
OPo
Ore
OreApply
OreCoefficientList
OreContent
OreCurrentMatrix
OreDeclareVariables
OreDefineDegLexOrder
OreDefineDiff
OreDefineDummy
OreDefineLexOrder
OreDefineOperator
OreDefineQDifferentiation
OreDefineReverseDegLexOrder
OreDefineReverseLexOrder
OreDefineShift
OreEliminateContent
OreExponent
OreExtendedSPolynomial
OreGetMonomOrder
OreInitialTerm
OreInverse
OreIsMonicQ
OreIsObjectQ
OreIsUnitQ
OreLeadingCoefficient
OreLeadingMonomial
OreLeadingOperatorMonomial
OreLeadingSummand
OreLesserTerms
OreListOfOrderObjects
OreMapOnCoefficients
OreMonomial
OreMultiplication
OreNumberOfObjects
OreNumberOfOperators
OreNumberOfVariables
OreOne
OreOneMonomial
OreOperatorNames
OreOrderData
OreOrderedListOfObjects
OrePolynomial
OrePolynomialsToVectorList
OreRedefineMonomOrder
OreShowStatus
OreSimplify
OreSimplifyCoefficients
OreSPoly
OreSummand
OreTerm
OreVarContent
OreVarEliminateContent
OreVariables
OreVarMonomialQ
OreZero
OreZeroSummand
Sigma
SolvableQ
SummandToVector

An einem Beispiel lernen wir, wie das Paket funktioniert.

Ein exotischerer Kandidat einer ordnungserhaltenden Ore-Algebra

Die Objekte der häufigsten Ore-Algebren dürften die Operatoren in der Tabelle von F. Chyzak & B. Salvy aus [ChySal] sein. Die Klassiker sind

- Differentiation (stetig),
- Shift (diskret) und
- q-Differentiation.

Diese algebraischen Objekte entsprechen (manchmal nicht eindeutig) "wirklichen" Operatoren. Dem algebraischen Objekt D aus der Differentialalgebra k⟨X,D⟩ mit DX=XD+1 entspricht eine lineare Operation (D "wirkt" auf ein p) auf Funktionen:
Diff: f(X)| →f'(X),

Der Shift aus k⟨n,S⟩ mit S n = (n+1)S entspricht der Wirkung:
f[n]→f[n], f(n)| →f(n+1)

Nun finden sich unter den ordnungserhaltenden Ore-Algebren weit mehr als diese "üblichen Kandidaten".

Weder müssen die σ_i nur von einer Variablen abhängen noch homogen sein. Je nach Monomordnung können einige σ_i sogar den absoluten Grad erhöhen.
Es mag dann zwar keine Entsprechung als Operatoren auf geeigneten Funktionenräumen mehr geben, dennoch ist es reizvoll auch diese Exoten zu untersuchen.
Die Ore-Algebra mit folgenden Substitutionshomomorphismen und den zugehörigen inneren Derivationen ist solch ein Beispiel:

σ_1: X1→X1+X2^2, X2→X2
δ_1: P(X_1,X_2)→P(X_1,X_2)-σ_1(P)(X_1,X_2)

σ_2: X1→X1-X2,X2→X2
δ_2: P(X_1,X_2)→P(X_1,X_2)-σ_2(P)(X_1,X_2)

Diese Algebra soll nun mit Hilfe des Paketes Ore.m untersucht werden.

Deklaration der Unbestimmten

Zur Definition wird ein Symbol angegeben, das nur der Darstellung in der Ausgabe dient, und σ und δ werden durch ihre Wirkung auf die Variablen definiert. Da wir hier den Operatoren keine wirkliche Operation zuordnen wollen, wird als letztes Argument eine Dummy-Funktion übergeben.

$D_1 = \mathbf{OreDefineOperator[D1, \{X1 \to X1 + X2\text{^}2, X2 \to X2\},}$
$\mathbf{\{X1 \to X2\text{^}2, X2 \to 0\}, \#\&]}$
$D_2 = \mathbf{OreDefineOperator[D2, \{X1 \to X1 - X2, X2 \to X2\},}$
$\mathbf{\{X1 \to X2, X2 \to 0\}, \#\&]}$

D1

D2

Das Zuweisen der Operatoren an D_1, D_2 ist wichtig, denn die Symbole D1, D2 dienen nur der Ausgabe und sind keine Ore-Polynome. Viele übliche Operatoren sind bereits vordefiniert.
Wenn wir nun eine Liste der Objekte ausgeben lassen, sehen wir, daß die auftauchenden Variablen automatisch als Unbestimmte registriert wurden:

OreOrderedListOfObjects

$\{\mathrm{D1}, \mathrm{D2}, \mathrm{X1}, \mathrm{X2}\}$

Ein Tip: Die automatische Ergänzungsfunktion von *Mathematica* funktioniert auch bei den Befehlen aus den nachgeladenen Paketen. Das erspart einerseits Tipparbeit und bietet obendrein ein Menü der möglichen Ergänzungen.

Eingabe von Ore-Polynomen

Zur Umwandlung gewöhnlicher Variablen in Ore-Polynome dient OrePolynomial, kurz OPo. Dies ist nötig, da die Rechenfunktionen nur mit Ore-Objekten arbeiten.

$\mathbf{Px = OPo[X1 + X2]}$

$\mathrm{X1} + \mathrm{X2}$

Die bei den Operatordefinitionen erhaltenen Objekte sind bereits Ore-Polynome. Polynome baut man nun mittels der Multiplikation und der Addition auf. Die Ad-

dition funktioniert, wie man es sich vorstellt:

Px + D_1

D1 + X1 + X2

Multiplikation

Für die Multiplikation wird der Befehl OreMultiplication, kurz OMu verwendet. Produkte von D_1 Mit den Variablen ergeben das von σ_1 zu erwartende Resultat:

{OMu[D_1, OPo[X1]], OMu[D_1, OPo[X2]]}

$\{\text{X1D1} + \text{X2}^2\text{D1} + \text{X2}^2, \text{X2D1}\}$

Multiplizieren wir an D_1 nacheinander D_2 und (X1+X2), also Px, ergibt das:

P = OMu[D_1, D_2, Px]

$\text{X1}(\text{D1D2}) + \text{X2}^2(\text{D1D2}) + \text{X2D1} + \text{X2}^2\text{D2}$

Die Ausgabe von P ist durch eine eigene Darstellungsfunktion lesbarer gehalten. Die interne Darstellung offenbart der *Mathematica*-Befehl FullForm:

FullForm[P]

OrePolynomial[
OreSummand[X1, OreMonomial[List[1, 1], List[2, 1]]],
OreSummand[Power[X2, 2],
OreMonomial[List[1, 1], List[2, 1]]],
OreSummand[X2, OreMonomial[List[1, 1]]],
OreSummand[Power[X2, 2], OreMonomial[List[2, 1]]]]

Vergleiche, Abrufen und Ändern der Ordnung

Die aktuelle Algebra mit der aktuellen Ordnung ist ordnungserhaltend. Den Test darauf leistet der Befehl "SolvableQ":

SolvableQ

True

Die Defaultordnung auf den Monomen ist lexikografisch, wobei die Operatoren größer als die Variablen sind. Ansonsten gilt die Reihenfolge der Deklaration:

OPo[X2] < OPo[X1]

True

Wir merken uns die aktuelle (defaultmäßig lexikografische) Ordnung:

merk = OreGetMonomOrder[]

$\{\{\mathrm{D1}, \mathrm{D2}, \mathrm{X1}, \mathrm{X2}\},$
$\{\{1,0,0,0\},\{0,1,0,0\},\{0,0,1,0\},\{0,0,0,1\}\}\}$

Nun tauschen wir die Reihenfolge der Variablen in der lexikografischen Ordnung:

OreRedefineMonomOrder[{X2, X1, D_1, D_2}, IdentityMatrix[4]]

Nun ist X2 nicht mehr kleiner als X1:

OPo[X2] < OPo[X1]

False

Und die Algebra ist nun nicht mehr ordnungserhaltend!

SolvableQ

False

Nehmen wir nun eine Monomordnung, die näher an einer Gesamtgrad-Ordnung liegt. Mit etwas Probieren kommt man auf die folgende Matrixordnung:

OreRedefineMonomOrder[{D_1, D_2, X1, X2},
{{3, 3, 3, 1}, {0, 1, 1, 1}, {0, 0, 1, 1}, {0, 0, 0, 1}}]

SolvableQ

True

Die Ordnung spielt für Gröbnerbasen eine wichtige Rolle. Die Algorithmen hierfür befinden sich im Paket "OreAlgorithms.m". Schon jetzt können wir den folgenden Satz aus der Arbeit am vorliegenden Beispiel prüfen:

A,B seien Matrizen zu Ordnungen bezüglich derer unsere Ore-Algebra ordnungserhaltend ist. Dann ist die Leitform von A, verfeinert durch die Matrix B ord-

nungserhaltend (Satz 6.1).

```
A = IdentityMatrix[4];
B = {{3, 3, 3, 1}, {0, 1, 1, 1}, {0, 0, 1, 1}, {0, 0, 0, 1}};
F = Drop[Prepend[B, First[A]], -1]
```

$\{\{1, 0, 0, 0\}, \{3, 3, 3, 1\}, \{0, 1, 1, 1\}, \{0, 0, 1, 1\}\}$

```
OreRedefineMonomOrder[{D1, D2, X1, X2}, F];
SolvableQ
```

True

Und weiter können wir einige Ordnungen zwischen "F" und "B" nehmen, um (6.3) zu testen. Wir betrachten zehn zufällige Leitgewichte zwischen A[[1]] und B[[1]] und ergänzen sie durch B:

```
zuf = Table[Random[], {10}];
gewichte = Map[# * First[A] + (1 - #) * First[B]&, zuf];
Zwi = Map[Drop[Prepend[B, #], -1]&, gewichte];
```

und testen die Solvable-Bedingung für alle Ordnungen:

```
merk = OreGetMonomOrder[]; Ergebnis = True;
For[i = 1, i <= 10, i++,
OreRedefineMonomOrder[{D1, D2, X1, X2}, Zwi[[i]]];
Ergebnis = Ergebnis&&SolvableQ;
]; Ergebnis
OreRedefineMonomOrder[Sequence@@merk];
```

True

Dabei wurde darauf geachtet, die Monomordnung wieder herzustellen. Das vermeidet unliebsame Überaschungen.

Höhere Algorithmen: Das Paket OreAlgorithms

Dieses Paket baut auf Ore.m auf, hier sind komplexere Algorithmen realisiert.

```
Needs["OreAlgorithms`"]
```

Schauen wir uns die hinzugekommenen Funktionen an:

?OreAlgorithms`*

OreAlgorithms`

BaseGenerators	OreLeftGroebner
InfoMessage	OreLeftGroebnerLimited
OreAlgebrasVersion	OreMonomialLCM
OreCheckExtended	OreNormalForm
OreDivision	OrePartDivision
OreDivisionExtended	OreRandomPolynomial
OreExtendedLeftGroebnerLimited	OreReduceExtendedGroebnerBasis
OreExtendedNormalForm	OreReduceGroebnerBasis
OreExtendedSPoly	OreReducePolynomial
OreGCD	OreTopReduce
OreIsLeftGroebner	OreTopReducibleQ

Es geht in erster Linie um Gröbnerbasisberechnungen und Funktionen im Umfeld davon. Für Gröbnerbasisberechnungen wird der (kaum optimierte) Algorithmus von Buchberger verwendet. Für ordnungserhaltende Algebren funktioniert das Verfahren garantiert.
Allerdings gibt es einen etwas ungewohnten Effekt: Eine Menge von Monomen muß nun keine Gröbnerbasis darstellen!
Wir betrachten in unserer exotischen Algebra folgende Monome:

OreRedefineMonomOrder[{D_1, D_2, **X1, X2**},
{{**3, 3, 3, 1**}, {**0, 1, 1, 1**}, {**0, 0, 1, 1**}, {**0, 0, 0, 1**}}]
$P =$ {D_1, **OMu**[**OPo**[**X2**], D_2], **OPo**[**X1^2**]}

$\{\mathrm{D1}, \mathrm{X2D2}, \mathrm{X1}^2\}$

Bilden wir nun die reduzierte Gröbnerbasis ...

$G =$ **OreReduceGroebnerBasis**[**OreLeftGroebner**[P]]

$\{\mathrm{X1}^2, -\mathrm{X1X2}^2, \mathrm{X2D2}, \mathrm{X2}^3, \mathrm{D1}\}$

... so stellen wir fest, daß neue Elemente auftauchen. Das beruht darauf, daß σ_1 hier Potenzen von X2 generiert!

Gehen wir in die lexikografische Ordnung und betrachten ein anderes Ideal:

OreDefineLexOrder;

P1 = {D_1 + **OPo**[**X1**], **OMu**[**OPo**[**X2**], D_2], **OPo**[**X1^2**]}

$\{\mathrm{D1} + \mathrm{X1}, \mathrm{X2D2}, \mathrm{X1}^2\}$

Timing[G = OreReduceGroebnerBasis[OreLeftGroebner[P1]]]

$\{0.146978\mathrm{Second}, \{\mathrm{D1} + \mathrm{X1}, \mathrm{X2D2}, \mathrm{X1}^2, \mathrm{X2}^2\}\}$

OreRedefineMonomOrder[{D_1, D_2, X1, X2},
{{3, 3, 3, 1}, {0, 1, 1, 1}, {0, 0, 1, 1}, {0, 0, 0, 1}}];
P1 = OreSimplify/@P1

$\{\mathrm{X1} + \mathrm{D1}, \mathrm{X2D2}, \mathrm{X1}^2\}$

Das Simplify ist hier wichtig: Es stellt die korrekte interne Darstellung nach einer Ordnungsänderung wieder her. Die Darstellung des ersten Elementes hat sich in der neuen Ordnung verändert!

Timing[G = OreReduceGroebnerBasis[OreLeftGroebner[P1]]]

$\{0.423936\mathrm{Second}, \{(-1)\mathrm{D1}^2, \mathrm{X2D2}, \mathrm{X1} + \mathrm{D1}, \mathrm{X2}^2\}\}$

Hier hat sich mit der Ordnung die Gröbnerbasis geändert.
Erwähnenswert dürfte noch die "extended"-Version der Gröbnerbasisberechnung sein. Sie versorgt uns noch mit einer Darstellung durch die alte Idealbasis. Das Anhängsel "Limited" bedeutet, daß eine obere Schranke angegeben wird, bei der die Berechnung weiterer S-Polynome abgebrochen wird:

EG = OreExtendedLeftGroebnerLimited[P1, 10]

$\{\{\mathrm{X1} + \mathrm{D1}, \{1, 0, 0\}\}, \{\mathrm{X2D2}, \{0, 1, 0\}\}, \{\mathrm{X1}^2, \{0, 0, 1\}\},$
$\{(-1)\mathrm{D1}^2 + (-\mathrm{X2}^2)\mathrm{D1} + -\mathrm{X2}^2, \{\mathrm{X1} + (-1)\mathrm{D1}, 0, -1\}\},$
$\{\mathrm{X2}^2, \{\mathrm{X2D2}, -\mathrm{X1} + (-1)\mathrm{D1} + \mathrm{X2}, 0\}\}\}$

Wir testen es am letzten Element: Der erste Eintrag muß aus den Idealelementen mit den Koeffizienten aus der mitgelieferten Liste hervorgehen:

L = Last[EG][[2]];
OMu[L[[1]], P1[[1]]] + OMu[L[[2]], P1[[2]]]+
OMu[L[[3]], P1[[3]]]

$\mathrm{X2}^2$

Das ist in der Tat der erste Eintrag. Die Extended-Versionen der anderen Befehle arbeiten mit diesen Paaren statt mit einfachen Ore-Polynomen.

Sehen wir nun, wie *Mathematica* -Befehle und die Befehle aus den Paketen beim Rechnen Hand in Hand gehen.

Zusammenarbeit mit *Mathematica*

Problem:
Wir haben Funktionalgleichungen, in denen Shifts und Differenzieren vorkommt und suchen eine Lösungsfunktion. Die Gleichungen sind zum Beispiel:

f(X,n+1)-X f(X,n) = 0 und 3 f(X,n+2)- 4 X^2 f'(X,n) = 0.

Vorgehen:
Durch Angabe einer geeigneten Termordnung lassen wir den Shift in der Gröbnerbasis des Verschwindungsideales bevorzugt eliminieren. So erhalten wir eine DGL, die mit Standardmethoden gelöst wird.
Wir fangen ganz von vorne an:

```
Quit[];

Needs["OreAlgorithms`"];
NShift = OreDefineShift[Sn, n];
Diff = OreDefineDiff[Dx, X];
```

Das sind die gewohnten Operatoren, die auf Ausdrücke anwendbar sind , z.B.

```
{OreApply[NShift, (X * n)^2], OreApply[Diff, (X * n)^2]}
```

$\{(1+n)^2X^2, 2n^2X\}$

Unsere Monomordnung soll den Shift höher gewichten:

```
OreRedefineMonomOrder[{NShift, Diff, n, X},IdentityMatrix[4]]
```

Es folgen unsere Funktionalgleichungen (wobei nach X abzuleiten ist):

```
A1 = NShift - OMu[OPo[X], Diff]
```

Sn + $(-X)$Dx

```
A2 = 3NShift^2 - 4OMu[OPo[X^2], Diff^2]
```

$3\text{Sn}^2 + (-4X^2)\text{Dx}^2$

Berechnen der Gröbnerbasis eliminiert hier den Shift. Mit "verboselevel 1" werden wir über den Fortschritt des Buchbergeralgorithmus unterrichtet. In komplizierteren Beispielen könnte man so eventuell erkennen, daß gar nicht die ganze Basis berechnet werden muß.

GB = OreLeftGroebner[{A1, A2}, 1]

As element 3 added $X^2\text{Dx}^2 + (-3X)\text{Dx}$ =!= 0 to G.

$\{\text{Sn} + (-X)\text{Dx}, 3\text{Sn}^2 + (-4X^2)\text{Dx}^2, X^2\text{Dx}^2 + (-3X)\text{Dx}\}$

Hier sieht man: Die Variable X in dem neuen Basiselement kann man ausklammern. Dieser Linksfaktor ist für die Gleichung P*f=0 unerheblich.
Wir eliminieren ihn mit OreVarEliminateContent:

reduced = OreVarEliminateContent[Last[GB]]

$X\text{Dx}^2 + (-3)\text{Dx}$

Diesem Ore-Polynom entspricht eine Differentialgleichung, die mit Standard-*Mathematica*-Methoden behandelt werden kann: "OreApply" führt uns in *Mathematicas* Welt zurück:

DGL = OreApply[GB[[3]], f[X, n]]

$-3Xf^{(1,0)}[X, n] + X^2 f^{(2,0)}[X, n]$

Solution = DSolve[DGL == 0, f[X, n], X]

$\{\{f[X, n] \to \frac{1}{4}X^4 C[1] + C[2]\}\}$

Hier finden sich zwei Konstanten. Diese hängen natürlich noch von n ab:

Solution = Solution/.{C[t_] → C[t, n]}

$\{\{f[X, n] \to \frac{1}{4}X^4 C[1, n] + C[2, n]\}\}$

Wir wenden nun den Operator A1 an, um diese Konstanten durch Koeffizientenvergleich zu bestimmen:

Collect[OreApply[A1, f[X, n]/.Solution], X]

$\{X^4(-C[1, n] + \frac{1}{4}C[1, 1 + n]) + C[2, 1 + n]\}$

Dies muß identisch Null sein, es folgt C[2,n]=0 sowie c[1,n+1]=4*C[1,n], also induktiv: c[1,n] = c 4^n mit C[1,0]=c.

Damit erhalten wir notwendig als Lösung:

$\{F\} = f[X, n]$/.Solution/.$\{C[2, n] \to 0, C[1, n] \to C4\text{^}n\}$

$\{4^{-1+n} C X^4\}$

Die Probe zeigt, daß unsere Lösung auch stimmt:

{OreApply[A1, F], OreApply[A2, F]}

$\{0, 0\}$

B Implementation des Gröbner–Walks und Beispiele

Das Paket “Gröbner–Walk” selbst kann als Beispiel für eine Anwendung des Paketes “OreAlgorithms” herhalten.

Zunächst wird die Implementation gezeigt.

Ein Beispiel, wo der Walk versagt wurde bereits analysiert. Es folgen einige Fälle, in denen der Walk zum Ziel führt.

Der Gröbner–Walk

Für Ore Algebren, nach "On the Walk",

Sicherstellen der Funktionen aus dem Paket OreAlgorithms

```
Needs["OreAlgorithms`"];
```

Die benötigten Grundfunktionen

RefinedOrder[w_,B_]: Zur Ordnungsmatrix B und Gewichtsvektor w die Matrix der Matrixordnung Ord(w,B) finden.

Hierzu wird aus der Matrix, die sich durch voranstellen von w^T an B ergibt, die unterste Zeile gestrichen, für die so eine nichtsinguläre Matrix entsteht.
Im theoretischen Teil wird einfach eine n $\times$ n+1 – Matrix verwendet. Die Implementation braucht aber eine quadratische, reguläre Matrix.

```
Clear[RefinedOrder];
RefinedOrder[w_/;Or@@(Thread[w!=0]), B_/;Det[B]=!=0]:=
Module[
{l, n, Result},
l = n = Length[B]; (* Number of Line to drop *)
Result = Prepend[Drop[B, -1], w];
While[Det[Result] == 0,
n = n - 1;
Result =
Flatten[{{w}, Take[B, (n - 1)], Drop[B, n]}, 1];
];
Return[Result];
];
```

SortOrePolynomials: Eine Liste von Orepolynom nach Leitmonomen fallend sortieren.

```
Clear[SortOrePolynomials];
SortOrePolynomials[G_List, verbosity_Integer:0]:=
Sort[G, #1 > #2&];
```

TakeInitials[G_List]: Aus einer Liste von Ore Polynomen die Initialterme extrahieren.

```
Clear[TakeInitials];
TakeInitials[G_List]:=
Map[OreInitialTerm[#]&, G];
```

InitGB[G_List]: Die Gröbnerbasis der Initialformen der Ore Polynome aus G bestimmen.

```
Clear[InitGB];
InitGB[G_List, verbosity_Integer:0]:=
Module[{ifo},
ifo = TakeInitials[G]; (* Get initial forms.*)
OreLeftGroebner[ifo]
];
```

Interreduce: (zwischen-) Reduktion einer gegebenen Gröbnerbasis.

```
Clear[Interreduce];
Interreduce[G_List, v_:0]:=OreReduceGroebnerBasis[G, v];
```

Liftup[Gw,G,{order}]: Liften der Elemente aus Gw bzüglich einer reduzierten Gröbnerbasis G zur Matrixordnung {order}.

Hierzu wird bezüglich der Ordnung "order" eine Division mit Rest berechnet. Dieser Rest ist dann zu subtrahieren, um ein Idealelement aus ⟨G⟩ mit dem glei-

chen Leitterm zu erhalten.

```
Clear[LiftUp];
LiftUp[Gw_List, G_List, {order__}, verbosity_Integer:0]:=
Module[{oldorder, ret},
InfoMessage[verbosity, 3, "Entering LiftUp:"];
oldorder = OreOrderData[];
OreRedefineMonomOrder[order];
ret = Map[# - OreDivision[#, OreSimplify/@G][[1]]&,
OreSimplify/@Gw];
OreRedefineMonomOrder[oldorder];
ret = OreSimplify/@ret
];
```

NextDegeneration[G,v]: Eine Funktion aus dem Package Ore.

Diese Funktion bestimmt den Punkt w auf der Geraden von der Leitform der aktuellen Monomordnung zum Zielgewicht v, an dem erstmals neue Monome im w-Initialpolynomen eines Ore-Polynomes g∈G auftauchen. Wegen Zugriff auf Interna ist dies in Ore.m realisiert.

DoTheWalk [G,DestOrder]: Gröbner–Walk zur Zielordnung DestOrder ausführen.
Achtung: Nach erfolgreicher Ausführung ist die Zielordnung die aktuelle!

G muß eine reduzierte Gröbnerbasis bezüglich der aktuellen Ordnung sein.
Defaultmäßig wird getestet, ob der letzte Schritt erfolgreich war, verbosity -1 schaltet dies ab. Da der Walk nicht immer zum Ziel führt, ist das nur zu empfehlen, wenn man sich sicher ist.

```
Clear[DoTheWalk];
DoTheWalk::"B singular" = "Singular destination order matrix ‘1‘.";

DoTheWalk[G_List, DestOrder_/;MatrixQ[DestOrder],
verbosity_Integer:0, maxsteps_:Infinity]:=
Module[{aO, (* aO : Actual ordermatrix here.*)
GG, (* The actual basis on the walk.*)
```

```
Gw, (* The initalforms of the elements of GG.*)
Gwp, (* Sorted Gw/new gröbnerbase.*)
vars, (* Current variable list.*)
A, (* Startorder.*)
B = DestOrder, (* Destination order.*)
v, (* The initialform of the destinationorder.*)
w, (* Initialform of actual order on the walk.*)
t, (* ratio from the last weight to the destination.*)
step = 0(* actual step of the Walk.*)
},
Catch[
If[Det[B] == 0, Message[DoTheWalk::"B singlular", B]; Abort[]];
InfoMessage[verbosity, 3, "Entering DoTheWalk:", G, DestOrder];

(*1.Initialisation *)

{vars, A} = {OreOrderData[]};
w = First[A]; v = First[B];

GG = G; (* Start with the given gröbnerbase.*)
InfoMessage[verbosity, 3, "Initialized: v,w,g=", v, w, GG];

While[True, (* main loop *)
InfoMessage[verbosity, 0,
Print["Check, wether actual GG is still a Gröbner base:
Current Matrix is:", OreCurrentMatrix];
If[OreIsLeftGroebner[GG], "It ist.", Print["No! Exit."];
OreRedefineMonomOrder[vars, A]; Throw[{"Walk failed!", GG}]]];

step = step + 1;

lastorder = OreOrderData[]; (* remember last order *)
aO = RefinedOrder[w, B]; (* Next order : *)
OreRedefineMonomOrder[vars, aO]; (* Setting the new order.*)
InfoMessage[verbosity, 1, "New order matrix is:",
OreCurrentMatrix, "."];
GG = (OreSimplify/@GG); (* Re − sort the objects.*)
```

```
InfoMessage[verbosity, 2, "last order: ",
{lastorder}[[2]]//MatrixForm, "actual order: ",
OreCurrentMatrix//MatrixForm, "Gb now: ", GG];

(*2. get the initialpolynomials : *)
Gw = TakeInitials[GG];
InfoMessage[verbosity, 2, "Step 2, initial forms:", Gw];

(*3. sort according to new order.*)

Gwp = SortOrePolynomials[Gw];
InfoMessage[verbosity, 2, "Step 3: Sorted, now Gwp=", Gwp];

(*4. Calculate groebnerbasis.*)

InfoMessage[verbosity, 2, "Step 4: Calculating GB of initials:"];
Gwp = InitGB[Gwp, verbosity];
InfoMessage[verbosity, 3, "It is:", Gwp];

(* Interreduction leads to failures in noncomm.case;
(*5. interreduction.*)
Gwp = Interreduce[Gwp];
*)

(*6. Lift to a gröbnerbasis.*)
GG = Lift$Up$[Gwp, GG, {lastorder}, verbosity];
InfoMessage[verbosity, 2, "Step 6: Gwp now lifted: GG=", GG];

(*7. interreduction.*)
GG = Interreduce[GG];
InfoMessage[verbosity, 2, "Step 7: Gwp reduced: GG=", GG];

(*8. are we ready?*)
InfoMessage[verbosity, 2, "Step 8: Ready? ", w===v];
If[w===v, Return[GG]];

(*9. search new border.*)
```

```
{w, t} = NextDegeneration[GG, v];

InfoMessage[verbosity, 2, "Step 9: New weight determined."];
InfoMessage[verbosity, 3, "t=", t, " weight=", w];
If[t==2, Return[GG]];

(*10.prepare next iteration.*)
(*w = (1 - t)w + tv; *)
InfoMessage[verbosity, 3, "Next weight vector: w=", w];
If[step>=maxsteps, Return[]];
InfoMessage[verbosity, 2, "Entering iteration ", step + 1];
]; (* While *)
]
];
```

Beispiele, in denen der Gröbner–Walk funktioniert.

In den betrachteten Beispielen handelt es sich um Differential/Shift-Algebren.

```
Needs["GroebnerWalk`"]
Diff = OreDefineDiff[Dx, x];
NShift = OreDefineShift[Sn, n];
{Varlist, Mat} = OreGetMonomOrder[]
```

$$\{\{\mathrm{Dx}, \mathrm{Sn}, x, n\}, \{\{1,0,0,0\}, \{0,1,0,0\}, \{0,0,1,0\}, \{0,0,0,1\}\}\}$$

Ein kleines Beispiel:

Wir definieren ein Ideal, repräsentiert durch zwei erzeugenden Elemente.

```
G1 = OMu[NShift, OPo[x^2]];
G2 = OMu[OPo[x], Diff] + OPo[n^2];
Ideal = {G1, G2}
```

$$\{x^2\mathrm{Sn}, x\mathrm{Dx} + n^2\}$$

Unsere Zielordnung soll die gradlexikografische mit Dx>Sn>x>n sein.
Wir schauen uns vorab das Ergebnis an.

```
C2 = {{1, 1, 1, 1}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
V = {Diff, NShift, x, n};
OreRedefineMonomOrder[V, C2]; (* Gradlex. Ordnung definieren. *);
Ideal = OreSimplify/@Ideal; (* Idealelemente normalisieren. *);
Timing[GBneu = OreLeftGroebnerLimited[Ideal, 11]]
GBneu = OreReduceGroebnerBasis[GBneu]
```

$\{\{1, 1, 1, 1\}, \{0, 1, 0, 0\}, \{0, 0, 1, 0\}, \{0, 0, 0, 1\}\}$

$\{0.282957\text{Second}, \{x^2\text{Sn}, x\text{Dx} + n^2,$
$(-n^2x)\text{Sn} + (-2nx)\text{Sn} + x\text{Sn}, (-n^4)\text{Sn} + (-4n^3)\text{Sn} + (-3n^2)\text{Sn} + (2n)\text{Sn}\}\}$

$\{(-n^4)\text{Sn} + (-4n^3)\text{Sn} + (-3n^2)\text{Sn} + (2n)\text{Sn},$
$(-n^2x)\text{Sn} + (-2nx)\text{Sn} + x\text{Sn}, x^2\text{Sn}, x\text{Dx} + n^2\}$

Die Basis in unserer Zielordnung C2 hat also vier Elemente.
Bezüglich der lexikografischen Ordnung mit n>x>Sn>Dx allerdings ist die Gröbnerbasis ganz einfach, wir wechseln wieder in die Startordnung und lassen sie ausrechnen:

```
C1 = {{0, 0, 0, 1}, {0, 0, 1, 0}, {0, 1, 0, 0}, {1, 0, 0, 0}};
OreRedefineMonomOrder[V, C1];
Ideal = OreSimplify/@Ideal;

Timing[GBneu = OreLeftGroebnerLimited[Ideal, 11]]
```

$\{0.024996\text{Second}, \{x^2\text{Sn}, n^2 + x\text{Dx}\}\}$

In der Tat reduziert sich das einzige S-Polynom zu Null:

```
Spol = OreSPoly[Ideal[[1]], Ideal[[2]]]
```

$(-2nx^2)\text{Sn} + (-x^3)(\text{DxSn}) + (-x^2)\text{Sn}$

```
OreDivision[Spol, Ideal]
```

$\{0, \{2n + x\text{Dx} + -1, 0\}\}$

Nun schauen wir, was der Gröbner–Walk macht. Wir lassen uns Startordnung, Zielordnung und aktuelle Ordnung anzeigen:

```
{C1//MatrixForm, C2//MatrixForm, MatrixForm/@OreGetMonomOrder[],
Ideal}
```

$$\{\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \{\begin{pmatrix} \mathrm{Dx} \\ \mathrm{Sn} \\ x \\ n \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\},$$
$\{x^2\mathrm{Sn}, n^2 + x\mathrm{Dx}\}\}$

Nun gehen wir in die Startordnung und “schreiten” zur Zielordnung. Mit Verbosity 3 lassen wir uns Informationen über den Ablauf des Algorithmus geben.

OreRedefineMonomOrder[*V*, C1];
Ideal = OreSimplify/@Ideal;
GB = DoTheWalk[Ideal, C2, 3]

Entering DoTheWalk:$\{x^2\mathrm{Sn}, n^2 + x\mathrm{Dx}\}$
$\{\{1, 1, 1, 1\}, \{0, 1, 0, 0\}, \{0, 0, 1, 0\}, \{0, 0, 0, 1\}\}$

Initialized: v,w,g=$\{1, 1, 1, 1\}\{0, 0, 0, 1\}\{x^2\mathrm{Sn}, n^2 + x\mathrm{Dx}\}$

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{0, 0, 0, 1\}, \{0, 0, 1, 0\}, \{0, 1, 0, 0\}, \{1, 0, 0, 0\}\}$

It is.

New order matrix is:$\{\{0, 0, 0, 1\}, \{1, 1, 1, 1\}, \{0, 1, 0, 0\}, \{0, 0, 1, 0\}\}$.

last order: $\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$ actual order: $\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ Gb now: $\{x^2\mathrm{Sn}, n^2 +$
$x\mathrm{Dx}\}$

Step 2, Initial forms:$\{x^2\mathrm{Sn}, n^2\}$

Step 3: Sorted, now Gwp=$\{x^2\mathrm{Sn}, n^2\}$

Step 4: Calculating GB of initials:

It is:$\{x^2\mathrm{Sn}, n^2\}$

Entering LiftUp:

Step 6: Gwp now lifted: GG=$\{x^2\mathrm{Sn}, n^2 + x\mathrm{Dx}\}$

Step 7: Gwp reduced: GG=$\{n^2 + x\mathrm{Dx}, x^2\mathrm{Sn}\}$

Step 8: Ready? False

Step 9: New weight determined.

t=1 weight=$\{1,1,1,1\}$

Next weight vector: w=$\{1,1,1,1\}$

Entering iteration 2

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{0,0,0,1\},\{1,1,1,1\},\{0,1,0,0\},\{0,0,1,0\}\}$

It is.

New order matrix is:$\{\{1,1,1,1\},\{0,1,0,0\},\{0,0,1,0\},\{0,0,0,1\}\}$.

last order: $\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ actual order: $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ Gb now: $\{x\text{Dx}+n^2, x^2\text{Sn}\}$

Step 2, Initial forms:$\{x\text{Dx}+n^2, x^2\text{Sn}\}$

Step 3: Sorted, now Gwp=$\{x\text{Dx}+n^2, x^2\text{Sn}\}$

Step 4: Calculating GB of initials:

It is:$\{x\text{Dx}+n^2, x^2\text{Sn},$
$(n^2x)\text{Sn}+(2nx)\text{Sn}+(-x)\text{Sn}, n^4\text{Sn}+(4n^3)\text{Sn}+(3n^2)\text{Sn}+(-2n)\text{Sn}\}$

Entering LiftUp:

Step 6: Gwp now lifted: GG=$\{x\text{Dx}+n^2, x^2\text{Sn},$
$(n^2x)\text{Sn}+(2nx)\text{Sn}+(-x)\text{Sn}, n^4\text{Sn}+(4n^3)\text{Sn}+(3n^2)\text{Sn}+(-2n)\text{Sn}\}$

Step 7: Gwp reduced: GG=$\{n^4\text{Sn}+(4n^3)\text{Sn}+(3n^2)\text{Sn}+(-2n)\text{Sn},$
$(n^2x)\text{Sn}+(2nx)\text{Sn}+(-x)\text{Sn}, x^2\text{Sn}, x\text{Dx}+n^2\}$

Step 8: Ready? True

$\{n^4\text{Sn}+(4n^3)\text{Sn}+(3n^2)\text{Sn}+(-2n)\text{Sn},$
$(n^2x)\text{Sn}+(2nx)\text{Sn}+(-x)\text{Sn}, x^2\text{Sn}, x\text{Dx}+n^2\}$

OreIsLeftGroebner[GB]

True

Ein Blick auf die Daten zeigt, daß wir uns nun in der Zielordnung befinden und die gefundene Basis mit dem vorab bestimmten Ergebnis übereinstimmt:

{MatrixForm/@OreGetMonomOrder[], GB}

$$\{\{\begin{pmatrix} \text{Dx} \\ \text{Sn} \\ x \\ n \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\}, \{n^4\text{Sn} + (4n^3)\text{Sn} + (3n^2)\text{Sn} + (-2n)\text{Sn},$$
$$(n^2x)\text{Sn} + (2nx)\text{Sn} + (-x)\text{Sn}, x^2\text{Sn}, x\text{Dx} + n^2\}\}$$

Wir können also direkt wieder zurücklaufen:

GBzur = DoTheWalk[GB, C1, 3]

Entering DoTheWalk:$\{n^4\text{Sn} + (4n^3)\text{Sn} + (3n^2)\text{Sn} + (-2n)\text{Sn},$
$(n^2x)\text{Sn} + (2nx)\text{Sn} + (-x)\text{Sn}, x^2\text{Sn}, x\text{Dx} + n^2\}$
$\{\{0,0,0,1\},\{0,0,1,0\},\{0,1,0,0\},\{1,0,0,0\}\}$

Initialized: v,w,g=$\{0,0,0,1\}\{1,1,1,1\}\{n^4\text{Sn} + (4n^3)\text{Sn} + (3n^2)\text{Sn} + (-2n)\text{Sn},$
$(n^2x)\text{Sn} + (2nx)\text{Sn} + (-x)\text{Sn}, x^2\text{Sn}, x\text{Dx} + n^2\}$

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{1,1,1,1\},\{0,1,0,0\},\{0,0,1,0\},\{0,0,0,1\}\}$

It is.

New order matrix is:$\{\{1,1,1,1\},\{0,0,0,1\},\{0,0,1,0\},\{0,1,0,0\}\}$.

last order: $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ actual order:

$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$ Gb now: $\{n^4\text{Sn} + (4n^3)\text{Sn} + (3n^2)\text{Sn} + (-2n)\text{Sn},$
$(n^2x)\text{Sn} + (2nx)\text{Sn} + (-x)\text{Sn}, x^2\text{Sn}, n^2 + x\text{Dx}\}$

Step 2, Initial forms:$\{n^4\text{Sn}, (n^2x)\text{Sn}, x^2\text{Sn}, n^2 + x\text{Dx}\}$

Step 3: Sorted, now Gwp=$\{n^4$Sn$, (n^2x)$Sn$, x^2$Sn$, n^2+x$Dx$\}$

Step 4: Calculating GB of initials:

It is:$\{n^4$Sn$, (n^2x)$Sn$, x^2$Sn$, n^2+x$Dx$, (-2nx)$Sn$+x$Sn$, x$Sn$, (-n)$Sn$\}$

Entering LiftUp:

Step 6: Gwp now lifted: GG=$\{n^4$Sn$+(4n^3)$Sn$+(3n^2)$Sn$+(-2n)$Sn$,$
(n^2x)Sn$+(2nx)$Sn$+(-x)$Sn$, x^2$Sn$, n^2+x$Dx$, 0, 0, 0\}$

Step 7: Gwp reduced: GG=$\{x^2$Sn$, n^2+x$Dx$\}$

Step 8: Ready? False

Step 9: New weight determined.

t=1 weight=$\{0, 0, 0, 1\}$

Next weight vector: w=$\{0, 0, 0, 1\}$

Entering iteration 2

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{1, 1, 1, 1\}, \{0, 0, 0, 1\}, \{0, 0, 1, 0\}, \{0, 1, 0, 0\}\}$

It is.

New order matrix is:$\{\{0, 0, 0, 1\}, \{0, 0, 1, 0\}, \{0, 1, 0, 0\}, \{1, 0, 0, 0\}\}$.

last order: $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$ actual order: $\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$ Gb now: $\{x^2$Sn$, n^2+$
xDx$\}$

Step 2, Initial forms:$\{x^2$Sn$, n^2\}$

Step 3: Sorted, now Gwp=$\{x^2$Sn$, n^2\}$

Step 4: Calculating GB of initials:

It is:$\{x^2$Sn$, n^2\}$

Entering LiftUp:

Step 6: Gwp now lifted: GG=$\{x^2$Sn$, n^2+x$Dx$\}$

Step 7: Gwp reduced: GG=$\{n^2 + x\text{Dx}, x^2\text{Sn}\}$

Step 8: Ready? True

$\{n^2 + x\text{Dx}, x^2\text{Sn}\}$

Es hat geklappt! Ein Blick auf Schritt 6 im ersten Durchlauf zeigt, daß drei Elemente der Gröbnerbasis der Initialpolynome nicht geliftet wurden.
Würde Schritt 5 ausgeführt werden, also die Basis der Initialpolynome reduziert werden, hätte das vorletzte Element der Basis nach Schritt 4 das Element $x^2\text{Sn}$ eliminiert. Damit wäre die gefundene Basis unvollständig!

{C1//MatrixForm, C2//MatrixForm, MatrixForm/@OreGetMonomOrder[], GBzur}

$$\{\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \{\begin{pmatrix} \text{Dx} \\ \text{Sn} \\ x \\ n \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\}, \{n^2+x\text{Dx}, x^2\text{Sn}\}\}$$

Hier haben wir also ein Beispiel, wo der Gröbner–Walk korrekt Basen unterschiedlicher Längen ineinander überführt.

Ein weiteres kleines Beispiel:

Das folgende Beispiel könnte sich ergeben, wenn man Funktionen f(x,n) sucht mit f(x,n+1)=x f´(x,n) und n x^2 f´´(x,n)=f(x,n+2).

N1 = OMu[OPo[x], Diff] − NShift;
N2 = OMu[OPo[nx^2], Diff^2] − NShift^2;
Ideal = {N1, N2}

$\{x\text{Dx} + (-1)\text{Sn}, (nx^2)\text{Dx}^2 + (-1)\text{Sn}^2\}$

Wie in der Einführung untersuchen wir das Verschwindungsideal.
Wir berechnen die Gröbnerbasis bezüglich der lexikografischen Ordnung:

OreDefineLexOrder;
{Varlist, C1} = OreGetMonomOrder[]; {MatrixForm[Varlist], MatrixForm[C1]}
Ideal = OreSimplify/@Ideal
Timing[GBneu = OreLeftGroebnerLimited[Ideal, 11]]

GBstart = OreReduceGroebnerBasis[GBneu]

$$\{\begin{pmatrix} \text{Dx} \\ \text{Sn} \\ x \\ n \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\}$$

$$\{x\text{Dx} + (-1)\text{Sn}, (nx^2)\text{Dx}^2 + (-1)\text{Sn}^2\}$$

$$\{0.376943\text{Second}, \{x\text{Dx} + (-1)\text{Sn}, (nx^2)\text{Dx}^2 + (-1)\text{Sn}^2, \\ (-n)\text{Sn}^2 + \text{Sn}^2 + n\text{Sn}, \text{Sn}^3 + (-1)\text{Sn}^2, \text{Sn}^2, n\text{Sn}\}\}$$

$$\{x\text{Dx} + (-1)\text{Sn}, \text{Sn}^2, n\text{Sn}\}$$

Hier sieht man ziemlich leicht, daß nur die Nullfunktion das Ausgangsproblem löst. Aber sehen wir davon einmal ab.
Nun wird so eliminiert, daß möglichst ein Ausdruck in x, Dx übrigbleibt: also, Sn, n bekommen hohe Priorität, x und Dx kommen am Schluß.

C2 = {{0, 1, 0, 1}, {0, 0, 1, 0}, {1, 0, 0, 1}, {1, 0, 0, 0}};
C2//MatrixForm
OreRedefineMonomOrder[*V*, C2];
Ideal = OreSimplify/@Ideal;
Timing[GBneu = OreLeftGroebnerLimited[Ideal, 11]]
GBneu = OreReduceGroebnerBasis[GBneu]

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\{0.678897\text{Second}, \{(-1)\text{Sn} + x\text{Dx}, (-1)\text{Sn}^2 + (nx^2)\text{Dx}^2, \\ (-nx^2)\text{Dx}^2 + x^2\text{Dx}^2 + x\text{Dx}, (-x^3)\text{Dx}^3 + (-2x^2)\text{Dx}^2, x^2\text{Dx}^2 + x\text{Dx}, (nx)\text{Dx}\}\}$$

$$\{(nx)\text{Dx}, (-1)\text{Sn} + x\text{Dx}, x^2\text{Dx}^2 + x\text{Dx}\}$$

Das war die direkte Berechnung. Nun gehen wir zurück zur lexikografischen Ordnung, und probieren den Walk, diesmal werden drei Iterationen gebraucht:

OreDefineLexOrder

Result = DoTheWalk[GBstart, C2, 2, 5]

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{1,0,0,0\},\{0,1,0,0\},\{0,0,1,0\},\{0,0,0,1\}\}$

It is.

New order matrix is:$\{\{1,0,0,0\},\{0,1,0,1\},\{0,0,1,0\},\{1,0,0,1\}\}$.

last order: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ actual order:
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$ Gb now: $\{x\text{Dx} + (-1)\text{Sn}, \text{Sn}^2, n\text{Sn}\}$

Step 2, Initial forms:$\{x\text{Dx}, \text{Sn}^2, n\text{Sn}\}$

Step 3: Sorted, now Gwp=$\{x\text{Dx}, \text{Sn}^2, n\text{Sn}\}$

Step 4: Calculating GB of initials:

Step 6: Gwp now lifted: GG=$\{x\text{Dx} + (-1)\text{Sn}, \text{Sn}^2, n\text{Sn}\}$

Step 7: Gwp reduced: GG=$\{x\text{Dx} + (-1)\text{Sn}, n\text{Sn}, \text{Sn}^2\}$

Step 8: Ready? False

Step 9: New weight determined.

Entering iteration 2

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{1,0,0,0\},\{0,1,0,1\},\{0,0,1,0\},\{1,0,0,1\}\}$

It is.

New order matrix is:$\{\{\frac{1}{2},\frac{1}{2},0,\frac{1}{2}\},\{0,1,0,1\},\{0,0,1,0\},\{1,0,0,1\}\}$.

last order: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$ actual order:

$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$ Gb now: $\{(-1)\text{Sn} + x\text{Dx}, n\text{Sn}, \text{Sn}^2\}$

Step 2, Initial forms:$\{(-1)\text{Sn} + x\text{Dx}, n\text{Sn}, \text{Sn}^2\}$

Step 3: Sorted, now Gwp=$\{(-1)\text{Sn} + x\text{Dx}, n\text{Sn}, \text{Sn}^2\}$

Step 4: Calculating GB of initials:

Step 6: Gwp now lifted: GG=
$\{(-1)\text{Sn} + x\text{Dx}, n\text{Sn}, \text{Sn}^2, (-x^2)\text{Dx}^2 + (-x)\text{Dx}, (nx)\text{Dx}\}$

Step 7: Gwp reduced: GG=$\{(nx)\text{Dx}, (-x^2)\text{Dx}^2 + (-x)\text{Dx}, (-1)\text{Sn} + x\text{Dx}\}$

Step 8: Ready? False

Step 9: New weight determined.

Entering iteration 3

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{\frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2}\}, \{0, 1, 0, 1\}, \{0, 0, 1, 0\}, \{1, 0, 0, 1\}\}$

It is.

New order matrix is:$\{\{0, 1, 0, 1\}, \{0, 0, 1, 0\}, \{1, 0, 0, 1\}, \{1, 0, 0, 0\}\}$.

last order: $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$ actual order: $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
Gb now: $\{(nx)\text{Dx}, (-x^2)\text{Dx}^2 + (-x)\text{Dx}, (-1)\text{Sn} + x\text{Dx}\}$

Step 2, Initial forms:$\{(nx)\text{Dx}, (-x^2)\text{Dx}^2 + (-x)\text{Dx}, (-1)\text{Sn}\}$

Step 3: Sorted, now Gwp=$\{(nx)\text{Dx}, (-x^2)\text{Dx}^2 + (-x)\text{Dx}, (-1)\text{Sn}\}$

Step 4: Calculating GB of initials:

Step 6: Gwp now lifted: GG=$\{(nx)\text{Dx}, (-x^2)\text{Dx}^2 + (-x)\text{Dx}, (-1)\text{Sn} + x\text{Dx}\}$

Step 7: Gwp reduced: GG=$\{(nx)\text{Dx}, (-1)\text{Sn} + x\text{Dx}, (-x^2)\text{Dx}^2 + (-x)\text{Dx}\}$

Step 8: Ready? True

$\{(nx)\text{Dx}, (-1)\text{Sn} + x\text{Dx}, (-x^2)\text{Dx}^2 + (-x)\text{Dx}\}$

Probehalber laufen wir nochmals zurück:

DoTheWalk[Result, C1]

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{0, 1, 0, 1\}, \{0, 0, 1, 0\}, \{1, 0, 0, 1\}, \{1, 0, 0, 0\}\}$

It is.

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{0, 1, 0, 1\}, \{1, 0, 0, 0\}, \{0, 1, 0, 0\}, \{0, 0, 1, 0\}\}$

It is.

Check, whether actual GG is still a Gröbner base: Current matrix is:
$\{\{\frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2}\}, \{1, 0, 0, 0\}, \{0, 1, 0, 0\}, \{0, 0, 1, 0\}\}$

It is.

$\{x\text{Dx} + (-1)\text{Sn}, \text{Sn}^2, n\text{Sn}\}$

Und wir sind wieder zurückgelangt!

Literatur

[AmGlKü] *Beatrice Amrhein, Oliver Gloor, Wolfgang Küchlin*: On the Walk, *Theoretical Computer Science*, **187** (1997), 179-202

[BeWpf] *Thomas Becker, Volker Weispfenning*: Gröbner Bases, A Computational Approach to Commutative Algebra, Springer Verlag, Berlin, 1993

[ChySal] *Frédéric Chyzak, Bruno Salvy*: Non-commutative Elimination in Ore Algebras Proves Multivariate Identities, *J. Symbolic Computation*, **11** (1996), 1-41

[CoKaMa] Stéphane Collart, Michael Kalkbrenner, Daniel Mall: Converting Bases with the Gröbner Walk, *J. Symbolic Computation*, **24** (1997), 465-469

[GoWa] *Kenneth R. Goodearl, Robert B. Warfield*: An introduction to noncommutative Noetherian Rings, Volume 16 of London Mathematical Society Student Texts, Cambridge University Press, Cambridge, 1989.

[Keller] *Benjamin J. Keller*: Algorithms and Orders for Finding Noncommutative Groebner Bases, Virginia Polytechnic Institute and State University, 1997.

[KRW] *A. Kandri-Rody, V. Weispfenning*: Non-commutative Gröbner Bases in Algebras of Solvable Type, *J. Symbolic Computation*, **9** (1990), 1-26

[Levand] *Viktor Levandovskyy*: Non–commutative Computer Algebra for polynomial algebras: Gröbner bases, applications and implementation, Dissertation an der Universität Kaiserslautern, 2005

[Mora86] *Ferdinando Mora*: Groebner Bases for Non-Commutative Polynomial Rings, *AAECC-3 Lecture Notes in Computer Science*, **229**, 353-362, Springer Verlag, Berlin Heidelberg, 1986

[MoRo] *Teo Mora, Lorenzo Robbiano*: The Gröbner Fan of an Ideal, *J. Symbolic Computation*, **6** (1988),183-208

[Mora94] *Teo Mora*: An introduction to commutative and noncommutative Gröbner bases, *Theoretical Computer Science*, **134** (1994), 131-173

[Pesch] *Michael Pesch*: Gröbner Bases in Polynomial Rings, Shaker Verlag, 1998

[Robb] *Lorenzo Robbiano*: Term Orderings on the Polynomial Ring, *Proceedings of EUROCAL 85, Lecture Notes in Computer Science*, **204** (1985), 513-517.

[Roda] *Giovanna Roda*: Algorithms for change of orderings in the theory of Gröbner bases, Dissertation am Institut für Symbolisches Rechnen, Linz, 2004

[WPF] *Volker Weispfenning*: Admissible Orders And Linear Forms, *ACM Sigsam Bull*, **27** (1987), 16-18

Index

Hiermit versichere ich, daß ich die vorliegende Dissertation selbständig und ohne unerlaubte Hilfe angefertigt und andere als die in der Dissertation angegebenen Hilfsmittel nicht benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, habe ich als solche kenntlich gemacht. Kein Teil dieser Arbeit ist in einem anderen Promotions- oder Habilitationsverfahren verwendet worden.