Universität Leipzig
Fakultät für Mathematik und Informatik
Mathematisches Institut

Lösen von Gleichungssystemen mit kontinuierlichen Symmetrien

Diplomarbeit

Leipzig
im Februar 1996

vorgelegt von
Ralf Hemmecke

Inhaltsverzeichnis

1 Einleitung **4**

2 Ausgangsposition **8**

2.1 Problemstellung 8

2.2 Erste Überlegungen 8

2.3 Konventionen 9

3 Gröbnerbasen **10**

3.1 Definitionen 10

3.2 Eigenschaften einer Gröbnerbasis 14

4 Grundlegende Algorithmen **16**

4.1 Pseudo-Normalform-Algorithmus 16

4.2 Buchbergers Algorithmus 19

4.3 Minimale Gröbnerbasis 20

4.4 Reduzierte Gröbnerbasis 21

4.5 Variationen des Buchberger-Algorithmus 22

5 Gröbnerbasen und Spezialisierung **24**

6 Ein neuer Algorithmus **27**

6.1 Herkunft und Entstehung 27

6.2 Der Algorithmus gamma 28

6.3 Verallgemeinerung 32

7 Beispiele **34**

7.1 Automorphismen von Lie-Algebren 34

7.2 Hilfsmittel 38

7.3 Die Lie-Algebra $A_{4,7}$ 38

7.4 Die Lie-Algebra $A_{5,2}$ 44

7.5 Effizienzuntersuchungen 49

8 Zusammenfassung und Ausblick **52**

8.1 Kurzzusammenfassung 52

8.2 Ausblick auf weitere Untersuchungen 52

A Verwendung von REDUCE **55**

B Implementation von gamma **59**

C Notation **63**

Literaturverzeichnis **64**

Erklärung **66**

Kapitel 1

Einleitung

Viele Probleme in Physik, Chemie und anderen Wissenschaften oder Wissenschaftszweigen, z. B. Robotik, können auf Probleme mit Idealen transformiert werden, die von einer endlichen Anzahl von Polynomen erzeugt werden, etwa die Bestimmung der gemeinsamen Nullstellen von Polynomen eines Ideals oder die Frage, ob ein gegebenes Polynom in einem vorgegebenen Ideal liegt.

B. Buchberger zeigte, daß viele Probleme in diesem Zusammenhang gelöst werden können, wenn man zu einem Erzeugendensystem des gegebenen Ideals übergeht, das gewisse weitere Eigenschaften hat. Solche Erzeugendensysteme sind heute unter dem Namen Gröbnerbasis oder Standardbasis bekannt. Zur Angabe einer Gröbnerbasis gehört immer auch die Angabe einer Ordnung auf den Potenzprodukten der vorkommenden Variablen, d. h. einer Termordnung.

Buchberger gab außerdem einen Algorithmus an, der ein gegebenes Erzeugendensystem in eine Gröbnerbasis transformiert, weiterhin auch ein algorithmisch entscheidbares Kriterium, ob eine Menge von Polynomen eine Gröbnerbasis ist oder nicht.

Hierbei kommt auch einer Verallgemeinerung der bekannten Division mit Rest eine verstärkte Bedeutung zu. In der Theorie der Gröbnerbasen wird dies dann nicht Division, sondern Reduktion eines Polynoms bezüglich einer vorgegebenen Menge von Polynomen genannt. Dies liefert eine Reduktionsrelation.

Gröbnerbasen haben die positive Eigenschaft, daß der Rest eines Polynoms bei einer vollständigen Reduktion (das ist eine solche, bei der das entstandene Polynom nicht weiter reduziert werden kann) eindeutig bestimmt ist, d. h. nicht davon abhängt, in welcher Weise die Reduktion ausgeführt wird. Mit anderen Worten heißt das: Bezüglich einer Gröbnerbasis besitzt die oben genannte Reduktionsrelation die Church-Rosser-Eigenschaft.

Die Frage, ob ein vorgegebenes Polynom in einem vorgegebenen Ideal liegt, das sogenannte Enthaltenseinsproblem, kann algorithmisch beantwortet werden, indem zuerst mit dem Buchberger-Algorithmus eine Gröbnerbasis G dieses Ideals (bzgl. einer beliebigen Termordnung) bestimmt wird und dann dieses Polynom bezüglich dieser Gröbnerbasis reduziert wird. Das Polynom ist genau dann bzgl. G zu Null reduzierbar, wenn es im Ideal liegt.

Ebenso können ähnliche Probleme gelöst werden, etwa die Frage, ob zwei verschiedene Mengen von Polynomen dasselbe Ideal erzeugen, oder ob zwei Polynome modulo eines gewissen Ideals kongruent sind. Von Interesse ist auch die Beantwortung der Frage nach der Existenz gemeinsamer Nullstellen eines Systems von Polynomen oder nach der Endlichkeit oder Unendlichkeit der Lösungsmenge, die sich in einfacher Weise aus der zugehörigen Gröbnerbasis ablesen läßt. Eine ausführlichere Auflistung von Problemen im Zusammenhang mit Gröbnerbasen und Methoden zu ihrer Behandlung ist in [2] zu finden.

Buchbergers Algorithmus kann als Verallgemeinerung des Gaußschen Algorithmus angesehen werden. Im Falle eines Gleichungssystems mit ausschließlich linearen Polynomen überführt er, in genau derselben Weise wie der Gaußsche Algorithmus, dieses Gleichungssystem in eine Dreiecksgestalt. Mit dem Buchberger-Algorithmus können aber auch Polynome höheren als ersten Grades behandelt werden. Hierbei kommt allerdings der Termordnung eine größere Bedeutung zu, da sie wesentlich die Laufzeit und die Form des Ergebnisses beeinflußt.

Für die Suche nach gemeinsamen Nullstellen eines gegebenen Systems von Polynomen ist eine Gröbnerbasis bezüglich der lexikographischen Termordnung von Vorteil, da diese eine „Dreiecksgestalt“ aufweist und aus ihr schrittweise die Variablen bestimmt werden können. Der Nachteil bei Verwendung dieser Ordnung liegt in der Tatsache begründet, daß in vielen Fällen die Berechnung einer Gröbnerbasis nicht in annehmbarer Zeit ausführbar ist und somit versucht werden muß, eine Gröbnerbasis bezüglich einer anderen Termordnung zu bestimmen und aus dieser Aussagen über die Nullstellen zu extrahieren.

Obwohl die Methode der Gröbnerbasen bereits in Buchbergers Doktorarbeit von 1965 zu finden ist, blieb sie relativ lange Zeit unbeachtet. Heute jedoch gibt es zahlreiche Forschungsbereiche, die direkt oder indirekt mit Gröbnerbasen arbeiten.

Mit der Entwicklung von Computeralgebrasystemen und deren Verbreitung wurde auch Buchbergers Algorithmus implementiert. Obwohl dieser Algorithmus im schlechtesten Fall doppelt-exponentiell von der Anzahl der auftretenden Variablen abhängt, sind für viele in der Praxis vorkommenden Probleme die Rechnungen ausführbar. Somit wurde Buchbergers Algorithmus ein wichtiges Hilfsmittel bei der Lösung vieler praktischer Probleme, die sich auf Probleme mit Polynomen transformieren lassen.

Im Laufe der Zeit wurden verstärkt Anstrengungen unternommen, diesen Algorithmus zu verfeinern. Es entstanden Selektionsstrategien für die Auswahl der zu behandelnden S-Polynome. Der Einfluß der Termordnung auf das Laufzeitverhalten wurde untersucht. Heute existieren auch Verfeinerungen, die eine Zerlegung auftretender Polynome in mehrere Faktoren ausnutzen, um für gewisse Klassen von Eingaben einen Geschwindigkeitsgewinn zu erzielen.

In dieser Arbeit wird ein Verfahren untersucht, das den Buchberger-Algorithmus dahingehend verfeinert, daß auch gewisse Symmetrien, die bei manchen praktischen Problemen vorhanden sind, ausgenutzt werden können, um einen eventuellen Rechenvorteil zu erlangen.

Über eine Verbindung von Symmetrien mit dem Buchberger-Algorithmus sind nur wenige Veröffentlichungen bekannt. Gatermann z. B. benutzt in [6] diskrete Symmetrien, um dann zu zeigen, daß bei der Abarbeitung des Buchberger-Algorithmus Polynome entstehen, die faktorisiert werden können, und sich damit das Problem in kleinere Teilprobleme zerlegen läßt.

Das hier vorgestellte Verfahren verwendet Symmetrien, die von einer Anzahl von Parametern abhängen, um das Ausgangsproblem in „kleinere“ Probleme zu zerlegen. „Kleiner“ heißt dann, daß Nullstellenmengen geringerer Dimension beschrieben werden, aus denen die Gesamtlösung rekonstruiert werden kann.

Die Arbeit gliedert sich wie folgt:
Im Kapitel 2 wird die Problemstellung formuliert und ein erster Ansatz für eine Lösung gegeben.

Kapitel 3 führt in die Theorie der Gröbnerbasen ein und präsentiert grundlegende Aussagen für die weiteren Untersuchungen.

Im Kapitel 4 werden grundlegende Algorithmen beschrieben, die für die Konstruktion einer Gröbnerbasis nötig sind. Dabei werden diese Algorithmen gleich in einer „nennerfreien“ Form angegeben.

Eine wesentliche Aussage, die zu einer einfacheren Formulierung des Algorithmus **gamma** führt, wird im Kapitel 5 bewiesen.

Kapitel 6 präsentiert den Algorithmus **gamma** und gibt einen Beweis für dessen Korrektheit.

Im Kapitel 7 wird die Wirkungsweise von **gamma** an einigen Beispielen demonstriert.

Das Kapitel 8 gibt eine Kurzzusammenfassung und einen Ausblick auf Untersuchungsrichtungen, die bei einer zukünftigen Betrachtung berücksichtigt werden sollten.

Eine kurze Abhandlung über die Verwendung des Computeralgebrasystems REDUCE sowie eine Implementation des Algorithmus **gamma** in REDUCE sind im Anhang zu finden.

An dieser Stelle möchte ich besonders Herrn Dr. H.-G. Gräbe danken, der bei der Erstellung der Arbeit meine Aufmerksamkeit oft auf die wesentlichen Punkte der Untersuchung gelenkt und mich auf diese Weise hervorragend betreut hat. Außerdem war er mir bei der Literaturrecherche behilflich und gab mir wertvolle Hinweise für die Verwendung von Routinen aus dem REDUCE-Paket CALI.

Äußerst dankbar bin ich meinem Bruder Raymond Hemmecke für seine Hinweise zur Korrektheit und Verständlichkeit der Darstellungen in dieser Arbeit und den damit verbundenen anregenden Gesprächen.

Cornelia Opitz und Bernd Schwennicke danke ich für ihre Hilfe bei der Durchsicht des Manuskripts.

Kapitel 2

Ausgangsposition

2.1 Problemstellung

Gegeben sei ein algebraisch abgeschlossener Körper K und eine Teilmenge $B = \{b_1, \ldots, b_p\}$ des Polynomrings $K[A] := K[A_1, \ldots, A_n]$ in den Unbestimmten $A_1, \ldots, A_n$.

Sei $\mathcal{A} := K^n$ ein affiner Raum. Γ sei eine durch $\mathcal{X} := K^m$ parametrisierte Gruppe, d. h., ein Element γ von Γ läßt sich schreiben in der Form $\gamma = \gamma(x)$ mit $x = (x_1, \ldots, x_m) \in \mathcal{X}$. Γ möge auf $\mathcal{A}$ operieren, und diese Operation sei durch Polynome $\gamma_1, \ldots, \gamma_n \in K[X_1, \ldots, X_m, A_1, \ldots, A_n]$ vermittelt, so daß für alle $x \in \mathcal{X}$ und $a \in \mathcal{A}$ gilt:

$$\gamma(x) \cdot a = (\gamma_1(x, a), \ldots, \gamma_n(x, a)) \in \mathcal{A}.$$

Wir fordern weiterhin, daß die Anwendung von Γ die gemeinsame **Nullstellenmenge**

$$\mathcal{Z}(B) := \{a \in \mathcal{A} \mid b_1(a) = \ldots = b_p(a) = 0\}$$

(der Polynome) von B invariant läßt.

Unter diesen Voraussetzungen ist eine Beschreibung der Nullstellenmenge $\mathcal{Z}(B)$ gesucht.

2.2 Erste Überlegungen

Ist nun $z \in \mathcal{Z}(B) \subseteq \mathcal{A}$ irgendeine gemeinsame Nullstelle von B, so gilt auch

$$\mathrm{Orb}(z) := \{\gamma \cdot z \mid \gamma \in \Gamma\} = \{\gamma(x) \cdot z \mid x \in \mathcal{X}\} \subseteq \mathcal{Z}(B),$$

d. h., die Bahn (auch der Orbit genannt) einer Nullstelle z von B bzgl. Γ enthält nur Nullstellen von B.

Ein Ansatz, das System B zu vereinfachen, ist die folgende Vorgehensweise. Wir versuchen, mit Hilfe von Γ in $\mathcal{Z}(B)$ auf eine Menge $Z \subseteq \mathcal{Z}(B)$ zu schließen, die nur Elemente enthält, für die gewisse gemeinsame Koordinaten verschwinden. Wenn $i_1, \ldots, i_l$ die Indizes jener Koordinaten sind, so bedeutet dies, daß

$$Z \subseteq \{a \in \mathcal{A} \mid a_{i_1} = \cdots = a_{i_l} = 0\} \cap \mathcal{Z}(B).$$

Außerdem möge Z so beschaffen sein, daß die Menge

$$\mathrm{Orb}(Z) := \{\mathrm{Orb}(z) | z \in Z\}$$

einen möglichst „großen" Teil von $\mathcal{Z}(B)$ überdeckt. Geometrisch ausgedrückt, heißt dies, wir geben eine Menge von Koordinaten-Hyperebenen in $\mathcal{A}$ vor und suchen unter allen Bahnen von $\mathcal{Z}(B)$ diejenigen heraus, von denen wir in einer effizienten Weise entscheiden können, daß sie einen Schnittpunkt mit dem Durchschnitt aller dieser Hyperebenen haben. Die Menge Z wird dann gerade von den Punkten dieser Bahnen gebildet, die im Durchschnitt der Hyperebenen liegen.

Offensichtlich hängt ein solches Z stark von der Lage der Nullstellenmenge $\mathcal{Z}(B)$ in $\mathcal{A}$ und der Wahl der Hyperebenen ab. Somit ist es natürlich auch möglich, daß nur $Z = \emptyset$ obige Bedingungen erfüllt. Dies heißt aber nur, daß für eine solche Problemstellung die in dieser Arbeit untersuchte Problemreduzierung keine Anwendung findet.

Im allgemeinen ist auch nicht zu erwarten, daß $\mathrm{Orb}(Z) = \mathcal{Z}(B)$ gilt. Wir müssen dann die Nullstellen von B, die sich nicht auf diese Weise erhalten lassen, auf eine andere Art bestimmen.

In dieser Arbeit werden wir obigen Ansatz genauer untersuchen. Dabei werden wir einen Algorithmus angeben, der ein solches Z und die „Ausnahmemenge" $\mathcal{Z}(B) \setminus \mathrm{Orb}(Z)$ berechnet. Es wird sich jedoch zeigen, daß die Zusatzbedingungen, die wir aus der Bestimmung von Z für die Berechnung der Ausnahmemenge erhalten, im allgemeinen nicht ausreichen, um auch für diesen Teil der Nullstellenmenge eine akzeptable Problemreduktion zu erhalten. Dies zeigt jedoch Wege, die bei der weiteren Untersuchung der Anwendung kontinuierlicher Symmetrien auf das Lösen polynomialer Gleichungssysteme beschritten werden können.

2.3 Konventionen

In der gesamten Arbeit ist R ein Ring (stets kommutativ und unitär) und K ein Körper. Außerdem schreiben wir $\mathcal{A}$ für K^n und $\mathcal{X}$ für K^m. $K[A]$ steht stets als Abkürzung für den Polynomring $K[A_1, \ldots, A_n]$ in den Unbestimmten $A_1, \ldots, A_n$ und $R[X]$ für $R[X_1, \ldots, X_m]$. Eine Übersicht über die Verwendung der Symbole kann im Anhang C gefunden werden.

Kapitel 3

Gröbnerbasen

In diesem Kapitel präsentieren wir einige Definitionen und Aussagen aus der Theorie der Gröbnerbasen. Eine umfassende Monographie zu Gröbnerbasen ist das Buch [1] von Becker und Weispfenning. Daraus sind die meisten Aussagen und Definitionen entnommen, die wir hier angeben. Wir beschränken uns bei der Darstellung der Theorie der Gröbnerbasen darauf, nur die Definitionen und Theoreme anzugeben, die für die weitere Arbeit wesentlich sind. Der Begriff der Reduzierbarkeit wurde nicht aus [1] übernommen, sondern statt dessen die Pseudoreduzierbarkeit eingeführt, da diese eine wesentliche Rolle bei den weiteren Betrachtungen spielt.

3.1 Definitionen

Die Bezeichnung gewisser Begriffe ist nicht immer einheitlich in der Literatur. Wir geben deshalb zuerst einige Definitionen an, um eine klare Begriffsverwendung für diese Arbeit zu schaffen.

Definition 3.1. Es sei R ein Ring und $R[X]$ der Polynomring in den Unbestimmten $X_1, \ldots, X_m$. Ein **Term** ist ein Potenzprodukt der Form

$$X_1^{i_1} \cdot \cdots \cdot X_m^{i_m}$$

mit nichtnegativen ganzen Zahlen $i_1, \ldots, i_m$. Durch $T(X_1, \ldots, X_m)$ oder kürzer $T(X)$ wird die Menge aller Terme in diesen Unbestimmten bezeichnet. $T(X)$ bildet offenbar zusammen mit der Multiplikation von $R[X]$ ein kommutatives Monoid, dessen Einselement 1 formal dem Term $X_1^0 \cdot \cdots \cdot X_m^0$ entspricht.

Ebenso haben wir in $T(X)$ die übliche **Teilbarkeitsrelation** $|$, d. h. für $s, t \in T(X)$:

$$s|t \Leftrightarrow \exists t' \in T(X) : s \cdot t' = t.$$

Ein **Monom** in den Unbestimmten $X_1, \ldots, X_m$ über R ist ein Produkt der Form $m = rt$ mit $r \in R$ und $t \in T(X)$.

Bemerkung. Die Begriffe Term und Monom treten in der mathematischen Literatur meist in vertauschter Bedeutung auf. Unsere Definition hält sich dagegen an die heutzutage übliche Verwendung dieser Begriffe in der Theorie der Gröbnerbasen.

Definition 3.2. Für ein Polynom

$$0 \neq f = \sum_{i=1}^{l} r_i t_i \qquad (0 \neq r_i \in R,\ t_i \in T(X) \text{ für } i = 1, \ldots, l),$$

bei dem alle t_i paarweise verschieden sind, bezeichne

$$C(f) = \{r_1, \ldots, r_l\}$$

die Menge der Koeffizienten von f,

$$T(f) = \{t_1, \ldots, t_l\}$$

die Menge der Terme von f und

$$M(f) = \{r_1 t_1, \ldots, r_l t_l\}$$

die Menge der Monome von f. Diese Mengen sind leer, falls $f = 0$.

Definition 3.3. Sei $\leq$ eine Totalordnung auf $T(X)$ und $f \neq 0$. Wenn $m = rt \in M(f)$ so gewählt ist, daß $t = \max T(f)$, so heißt $\mathrm{LT}(f) := t$ der **Leitterm** von f, $\mathrm{LC}(f) := r \in R$ der **Leitkoeffizient** von f und $\mathrm{LM}(f) := m$ das **Leitmonom** von f.

Sei $F \subseteq R[X]$. Wir bezeichnen mit $\mathrm{LT}(F) := \{\mathrm{LT}(f) \mid 0 \neq f \in F\}$ die Menge der Leitterme von F.

Definition 3.4. Sei $B = \{b_1, \ldots, b_p\}$ eine endliche Teilmenge des Polynomrings $R[X]$. Durch $\mathcal{I}_{R[X]}(B) := \mathcal{I}_{R[X]}(b_1, \ldots, b_p)$ werde das Ideal bezeichnet, das von den Polynomen aus B in $R[X]$ erzeugt wird.

Wenn aus dem Kontext hervorgeht, in welchem Polynomring das Ideal von B zu betrachten ist, schreiben wir kürzer $\mathcal{I}(B)$.

Die Konstruktion einer Gröbnerbasis steht in engem Zusammenhang mit dem Euklidischen Algorithmus für Polynome in einer Unbestimmten und stellt dessen Verallgemeinerung im Sinne der Anwendung auf Polynome in mehreren Unbestimmten dar. Bei Polynomen in einer Unbestimmten hat man die natürliche Gradordnung auf den Termen. Dies ist nicht mehr der Fall, wenn mehrere Unbestimmte auftreten. Im Begriff der Termordnung sind Eigenschaften genannt, die für die Konstruktion einer Gröbnerbasis nötig sind.

Definition 3.5. Eine **Termordnung** auf einem Monoid T ist eine Totalordnung $\leq$ auf T, die folgende Bedingungen erfüllt.

(i) $\forall t \in T : 1 \leq t$

(ii) $\forall s, t_1, t_2 \in T : t_1 \leq t_2 \Rightarrow t_1 \cdot s \leq t_2 \cdot s$

Die erste Bedingung impliziert gerade, daß eine Termordnung eine Wohlordnung ist. Wir schreiben auch $t_1 < t_2$ und meinen damit, daß t_1 und t_2 in der Relation $\leq$ stehen, aber nicht gleich sind.

Bemerkung. Da wir keine Untersuchungen bezüglich verschiedener Termordnungen vornehmen werden, sei im folgenden für eine Termmenge der Form $T(X_1, \ldots, X_m)$ eine beliebige, aber feste Termordnung gewählt. Auf diese beziehen sich dann auch LT, LC und LM.

Bemerkung. In einem euklidischen Ring R hat man die Begriffe eines **kleinsten gemeinsamen Vielfachen** und eines **größten gemeinsamen Teilers** zweier Elemente zur Verfügung. Allerdings sind diese nur bis auf einen in R invertierbaren Faktor eindeutig bestimmt. Wir schreiben trotzdem $r = \gcd(r', r'')$ (bzw. $r = \operatorname{lcm}(r', r'')$) und meinen damit, daß r ein größter gemeinsamer Teiler (bzw. kleinstes gemeinsames Vielfaches) von $r', r'' \in R$ ist. Eine analoge Aussage gilt für „den" Inhalt $\operatorname{content}(f)$ eines Polynoms $0 \neq f \in R[X]$. „Das" zu f gehörige primitive Polynom sei $\operatorname{primitivePart}(f) = f / \operatorname{content}(f)$.

Definition 3.6. Eine Teilmenge $G \subset K[X]$ heißt **Gröbnerbasis** (bzgl. der Termordnung $\leq$), wenn G endlich ist, $0 \notin G$ und für jedes $s \in \operatorname{LT}(\mathcal{I}(G))$ ein $t \in \operatorname{LT}(G)$ mit $t|s$ existiert.

Sei $I \subseteq K[X]$ ein Ideal. G heißt dann **Gröbnerbasis von** I, wenn G Gröbnerbasis ist und $I = \mathcal{I}(G)$ gilt.

Anstelle der Division mit Rest im Euklidischen Algorithmus tritt nun der allgemeinere Begriff der Reduktion eines Polynoms bezüglich einer Menge von Polynomen. Eine solche Reduktion ist wesentlicher Bestandteil des von Buchberger angegebenen Algorithmus.

Definition 3.7. Sei R ein euklidischer Ring und seien $f, \tilde{f}, f^*, g \in R[X]$ und $G = \{g_1, \ldots, g_k\} \subset R[X]$.

(i) f heißt **modulo** g **zu** $\tilde{f}$ **pseudo-reduzierbar** (in Zeichen: $f \to_g \tilde{f}$), falls $f, g \neq 0$, $f = \sum_{i=1}^{l} r_i t_i$, ($r_i \neq 0$ für $i = 1, \ldots, l$, alle t_i paarweise verschieden) und

$$r_0 \in C(f),\ t_0 \in T(f) \text{ mit } r_0 t_0 \in M(f)$$

existieren, so daß $\mathrm{LT}(g)|t_0$ und mit

$$\begin{aligned} \varrho &= \gcd(r_0, \mathrm{LC}\, g), \\ \tilde{r} &= \frac{\mathrm{LC}\, g}{\varrho}, \\ \tilde{r}_0 &= \frac{r_0}{\varrho} \qquad \text{und} \\ t &= \frac{t_0}{\mathrm{LT}\, g} \end{aligned}$$

außerdem gilt

$$\tilde{f} = \tilde{r}f - \tilde{r}_0 t g.$$

(ii) f heißt **modulo G zu $\tilde{f}$ pseudo-reduzierbar** (in Zeichen:$f \rightarrow_G \tilde{f}$), falls $f \rightarrow_g \tilde{f}$ für ein $g \in G$.

(iii) f heißt **pseudo-reduzierbar modulo G**, falls ein $\tilde{f} \in R[X]$ existiert, so daß f modulo G zu $\tilde{f}$ pseudo-reduzierbar ist.

(iv) f^* heißt genau dann eine **Pseudo-Normalform von f modulo G**, wenn f^* nicht pseudo-reduzierbar modulo G ist und $f \rightarrow_G^* f^*$ erfüllt.

Durch $\rightarrow_G^*$ werde der reflexive transitive Abschluß von $\rightarrow_G$ bezeichnet.

Bemerkung. Falls $f \rightarrow_g \tilde{f}$, so erkennen wir aus obiger Definition, daß ein Monom von f ersetzt wird durch ein Polynom, dessen Terme in der gegebenen Termordnung stets echt kleiner sind als der Term des entfernten Monoms. Wird bei der Reduktion das Leitmonom von f ersetzt, so können wir diese Aussage kürzer durch $\mathrm{LT}\, \tilde{f} < \mathrm{LT}\, f$ ausdrücken.

In jedem Falle gilt aber die Ungleichung $\mathrm{LT}\, \tilde{f} \leq \mathrm{LT}\, f$.

Bemerkung. In [1] werden analoge Begriffe ohne die Vorsilbe *pseudo* eingeführt. Dort wird aber statt eines euklidischen Rings R ein Körper K als Koeffizientenbereich vorausgesetzt.

Vergleichen wir diese Begriffe mit denen in unserer Definition unter der Voraussetzung, daß $R = K$ ein Körper ist, so stellen wir fest, daß sich die Polynome, die bei einer Pseudo-Reduktion bzw. Reduktion modulo G aus demselben Polynom entstehen, nur um einen konstanten Faktor (aus R) unterscheiden. Insbesondere fallen die Eigenschaften, reduzierbar bzw. pseudo-reduzierbar modulo G zu sein, zusammen. Ebenso gilt: 0 ist Normalform eines Polynoms f genau dann, wenn 0 Pseudo-Normalform von f modulo G ist.

Definition 3.8. Sei R ein Ring und Q sein Quotientenkörper. Eine Gröbnerbasis G eines Ideals I von $Q[X_1, \ldots, X_m]$ heißt **nennerfrei** (bzgl. R), wenn $G \subset R[X_1, \ldots, X_m]$.

Bemerkung. Der Begriff „nennerfrei“ taucht im Zusammenhang mit Gröbnerbasen nicht in der klassischen Literatur auf. Wir verwenden ihn hier nur, um auf einfache Weise sagen können, daß die Koeffizienten der Polynome in R liegen.

3.2 Eigenschaften einer Gröbnerbasis

In der Literatur sind verschiedene Definitionen des Begriffs Gröbnerbasis bekannt. In [1] sind für eine Menge von Polynomen äquivalente Bedingungen angegeben, Gröbnerbasis zu sein. Wir geben hier nur eine Auswahl an und verweisen für den Beweis auf [1].

Theorem 3.9. *Sei I ein Ideal von $K[X]$ und G eine endliche Teilmenge von I mit $0 \notin G$. Die folgende Aussagen sind äquivalent.*

(i) G ist eine Gröbnerbasis von I.

(ii) $f \to_G^* 0$ für alle $f \in I$.

(iii) Jedes $f \in I \setminus \{0\}$ ist reduzierbar modulo G.

(iv) Für jedes $s \in \mathrm{LT}(I)$ existiert ein $t \in \mathrm{LT}(G)$ mit $t|s$.

Beweis. Das Theorem ist eine Teilaussage von Proposition 5.38 in [1]. □

In der Theorie der Gröbnerbasen spielt der Begriff des S-Polynoms eine fundamentale Rolle. Die Definition dieses Begriffs ist in der Literatur nicht einheitlich und variiert von Autor zu Autor. Jedoch unterscheiden sich die nach verschiedenen Definitionen gebildeten S-Polynome nur um einen Faktor aus dem Koeffizientenbereich. Wir geben hier (wie schon bei der Reduzierbarkeit) eine „nennerfreie“ Formulierung an und bemerken, daß es beim S-Polynom nur darauf ankommt, zwei Polynome so zu addieren, daß die Summe beider Leitmonome verschwindet.

Definition 3.10. Sei R ein Ring und seien $g_1, g_2 \in R[X_1, \ldots, X_m]$ zwei Polynome. Das **S-Polynom** von $g_1 \neq 0$ und $g_2 \neq 0$ ist definiert als

$$\mathrm{spol}(g_1, g_2) := r_1\, t_1\, g_1 - r_2\, t_2\, g_2 \in R[X],$$

wobei $r_1 = \mathrm{LC}(g_2), r_2 = \mathrm{LC}(g_1) \in R$ und $t_1, t_2 \in T(X)$ so gewählt sind, daß

$$t_1\, \mathrm{LT}(g_1) = t_2\, \mathrm{LT}(g_2) = \mathrm{lcm}(\mathrm{LT}\, g_1, \mathrm{LT}\, g_2).$$

Ist $g_1 = 0$ oder $g_2 = 0$, so sei auch das S-Polynom gleich 0.

Für die algorithmische Entscheidbarkeit, ob eine vorgegebene Menge von Polynomen eine Gröbnerbasis ist, ist das folgende Theorem von fundamentaler Bedeutung.

Theorem 3.11. *Sei G eine endliche Teilmenge von $K[X]$ mit $0 \notin G$. Die folgenden Aussagen sind äquivalent:*

(i) *G ist eine Gröbnerbasis.*

(ii) *Wenn $g_1, g_2 \in G$ und $g^* \in K[X]$ eine (Pseudo-)Normalform von $g = \mathrm{spol}(g_1, g_2)$ ist, so ist $g^* = 0$.*

(iii) $\mathrm{spol}(g_1, g_2) \rightarrow_G^* 0$ *für alle $g_1, g_2 \in G$.*

Beweis. Dieses Theorem ist in [1] als Theorem 5.48 zu finden. Zusammen mit einer Bemerkung über Pseudo-Reduzierbarkeit, die wir im Anschluß an die Definition 3.7 gaben, können wir den Beweis aus [1] übernehmen. $\square$

Kapitel 4

Grundlegende Algorithmen

Auf Grund des Zieles, das wir in dieser Arbeit verfolgen, ist es von Vorteil, einen Normalformalgorithmus und einen Algorithmus zur Berechnung einer Gröbnerbasis in einer „nennerfreien“ Form zu präsentieren. Die Formulierung und der Beweis von Theorem 5.2 und dessen Anwendung beim Algorithmus **gamma** beruhen wesentlich auf der Tatsache, daß eine solche nennerfreie Form möglich ist.

Eine derartige nennerfreie Herangehensweise ist nicht neu bei der Berechnung einer Gröbnerbasis und wird bereits in vielen Computeralgebrasystemen aus Effizienzgründen eingesetzt. Für uns ist hier allerdings nur die Existenz solcher Algorithmen wesentlich.

In diesem Kapitel sei R ein euklidischer Ring, Q sein Quotientenkörper und $R[X] = R[X_1, \ldots, X_m]$ ein Polynomring.

4.1 Pseudo-Normalform-Algorithmus

Für die Konstruktion einer Gröbnerbasis ist eine Verallgemeinerung der Division mit Rest auf Polynome in mehreren Unbestimmten erforderlich. Dies führt auf den Begriff der Reduktion modulo einer Menge von Polynomen, den wir in der Definition 3.7 angegeben haben.

Lemma 4.1. *Der Algorithmus* **pseudoNormalForm** *terminiert und erfüllt seine Spezifikation.*

Beweis. Die Termination folgt aus der ersten Bemerkung zur Definition 3.7 und der Tatsache, daß eine Termordnung eine Wohlordnung ist.

Algorithmus pseudoNormalForm
Eingabe:

- $f \in R[X]$
- $G = \{g_1, \ldots, g_k\} \subset R[X]$

Ausgabe:

- $f^* \in R[X]$, so daß f^* eine Pseudo-Normalform von f modulo G ist.
- $r \in R$ Faktor, mit dem f multipliziert wurde, um Nenner zu vermeiden
- $H = \{h_1, \ldots, h_k\} \subset R[X]$

Für alle $1 \leq i \leq k$ gilt $\mathrm{LT}(h_i g_i) \leq \mathrm{LT}\, f$, wenn die entsprechende Bildung des Leitterms definiert ist und es gilt die Beziehung

$$rf = f^* + \sum_{i=1}^{k} h_i g_i.$$

<u>begin</u>
$f^* := f$
$r := 1$
Setze $h_i := 0$ für alle $i = 1, \ldots, k$.
<u>while</u> f^* (pseudo-)reduzierbar modulo G <u>do</u>
    Für einen gewissen Index $1 \leq i_0 \leq k$ gilt $f^* \to_g \tilde{f}$ mit $g = g_{i_0}$.
    Sei $r_0 t_0 \in M(f^*)$ das Monom, das ersetzt wird.
    Wähle $\tilde{r}$, $\tilde{r}_0$, t entsprechend Definition 3.7, d. h.
    $\varrho := \gcd(r_0, \mathrm{LC}\, g)$
    $\tilde{r} := \mathrm{LC}(g)/\varrho$
    $\tilde{r}_0 := r_0/\varrho$
    $t := t_0/\mathrm{LT}(g)$
    $\tilde{f} := \tilde{r} f^* - \tilde{r}_0 t g$
    $h_{i_0} := \tilde{r} h_{i_0} + \tilde{r}_0 t$ (H1)
    $h_i := \tilde{r} h_i$ für $1 \leq i \leq k$ mit $i \neq i_0$ (H2)
    $f^* := \tilde{f}$
    $r := \tilde{r} r$
<u>end while</u>
<u>return</u> (f^*, r, H)
<u>end pseudoNormalForm</u>

Tabelle 4.1: Algorithmus **pseudoNormalForm**

Der Algorithmus ist korrekt. Wir zeigen, daß die Relationen

$$rf = f^* + \sum_{i=1}^{k} h_i g_i, \tag{4.1}$$

$$h_1, \dots, h_k \in R[X], \tag{4.2}$$

$$r \in R, \tag{4.3}$$

$$f, f^* \neq 0 \Rightarrow \mathrm{LT}(f^*) \leq \mathrm{LT}(f), \tag{4.4}$$

$$\forall i \in \{1, \dots, k\} : (h_i g_i, f \neq 0 \Rightarrow \mathrm{LT}(h_i g_i) \leq \mathrm{LT}(f)) \tag{4.5}$$

Invarianten der while-Schleife bilden und somit auch am Ende des Algorithmus gelten.

Offenbar gelten diese Relationen zu Beginn der while-Schleife. Kennzeichnen wir zur besseren Unterscheidung die Variablen nach einer while-Iteration mit einem Strich, so ist wegen $\tilde{r} \in R$ natürlich auch $r' = \tilde{r}r \in R$. Da außerdem $\tilde{r}_0 \in R$ und $t \in T(X)$, folgt mit (H1) und (H2) die Relation (4.2). Weiterhin gilt

$$\begin{aligned}
rf &= f^* + \sum_{i=1}^{k} h_i g_i \\
\tilde{r}rf &= (\tilde{r}f^* - \tilde{r}_0 tg) + \tilde{r}_0 tg + \tilde{r}\sum_{i=1}^{k} h_i \\
r'f &= f^{*\prime} + \sum_{i=1}^{k} h_i' g_i,
\end{aligned}$$

womit wir auch (4.1) als Invariante bestätigen.

Für die folgenden Überlegungen nehmen wir an, daß alle auftretenden Leittermbildungen definiert sind, andernfalls ist nichts zu zeigen.

Mit obiger Vereinbarung gilt am Ende der while-Schleife $f^* \to_g f^{*\prime}$. Die Relation (4.4) bleibt somit wegen der ersten Bemerkung zur Definition 3.7 erhalten, denn $\mathrm{LT}(f^{*\prime}) \leq \mathrm{LT}(f^*) \leq \mathrm{LT}(f)$.

Nach Wahl von t im Algorithmus ist $t\,\mathrm{LT}(g_{i_0}) \in T(f^*)$, also

$$t\,\mathrm{LT}(g_{i_0}) \leq \mathrm{LT}(f^*) \leq \mathrm{LT}(f).$$

Um die Invarianz der Relation (4.5) zu zeigen, reicht es, $i = i_0$ zu betrachten, da anderenfalls $\mathrm{LT}(h_i' g_i) = \mathrm{LT}(h_i g_i) \leq \mathrm{LT}(f)$ gilt. Es ist aber

$$\begin{aligned}
\mathrm{LT}(h_{i_0}' g_{i_0}) &\leq \max\{\mathrm{LT}(\tilde{r} h_{i_0} g_{i_0}), \mathrm{LT}(\tilde{r}_0 t g_{i_0})\} \\
&= \max\{\mathrm{LT}(h_{i_0} g_{i_0}), t\,\mathrm{LT}(g_{i_0})\} \\
&\leq \mathrm{LT}(f).
\end{aligned}$$

Damit erweist sich auch die letzte Relation als invariant. □

Algorithmus buchberger
Eingabe: B = endliche Teilmenge von $R[X]$
Ausgabe: G = endliche Teilmenge von $R[X]$, so daß G eine nennerfreie Gröbnerbasis von $\mathcal{I}(B)$ in $Q[X]$ ist

```
begin
G := B
P := {{g1, g2} | g1, g2 ∈ G, g1 ≠ g2}
while P ≠ ∅ do
    wähle {g1, g2} aus P
    P := P \ {{g1, g2}}
    g := spol(g1, g2) (gemäß Definition 3.10)
    (g, r, H) := pseudoNormalForm(g, G)
    if g ≠ 0 then
        g := primitivePart(g) ∈ R[X]
        P := P ∪ {{ḡ, g} | ḡ ∈ G}
        G := G ∪ {g}                                    (G1)
    end if
end while
return G
end buchberger
```

Tabelle 4.2: Algorithmus **buchberger**

4.2 Buchbergers Algorithmus

Von Buchberger wurde ein Algorithmus entwickelt, der ein endliches Erzeugendensystem eines Ideals in eine Gröbnerbasis desselben Ideals transformiert. Der hier vorgestellte Algorithmus **buchberger** unterscheidet sich nur in unwesentlichen Punkten von dem von Buchberger in [2] angegebenen Algorithmus 6.2 bzw. dem Algorithmus GRÖBNER in [1].

Lemma 4.2. *Der Algorithmus* **buchberger** *terminiert und erfüllt seine Spezifikation.*

Beweis. Der Beweis der Korrektheit beruht wesentlich auf Theorem 3.11. Für einen vollständigen Beweis der Termination und der Korrektheit des Algorithmus verweisen wir auf Theorem 5.53 in [1]. Es bleibt nur anzumerken, daß im gesamten Algorithmus niemals eine Division auftritt, bei der der Ring R verlassen werden müßte. Offenbar sind also die Elemente von G am Ende des Algorithmus nennerfrei. □

Bemerkung. Ein Element der im Algorithmus auftretenden Menge P ist allgemein unter dem Namen **kritisches Paar** bekannt.

Algorithmus minimalBasis
Eingabe: B = endliche Teilmenge von $R[X]$, die nennerfreie Gröbnerbasis ist
Ausgabe: G = endliche Teilmenge von $R[X]$, so daß G eine minimale nennerfreie Gröbnerbasis von $\mathcal{I}(B)$ in $Q[X]$ ist

<u>begin</u>
$F := G := B$
<u>while</u> $F \neq \emptyset$ <u>do</u>
    wähle $f \in F$
    $F := F \setminus \{f\}$
    <u>if</u> $\exists g \in G \setminus \{f\}$, so daß $\mathrm{LT}(g) | \mathrm{LT}(f)$ <u>then</u>
        $G := G \setminus \{f\}$ (G2)
    <u>end if</u>
<u>end while</u>
<u>return</u> G
<u>end minimalBasis</u>

Tabelle 4.3: Algorithmus **minimalBasis**

4.3 Minimale Gröbnerbasis

Definition 4.3. Eine **minimale** Gröbnerbasis G ist eine Gröbnerbasis, die das Nullpolynom nicht enthält und für die gilt:

$$\forall g \in G \; \forall t \in \mathrm{LT}(G \setminus \{g\}) : \; t \nmid \mathrm{LT}\, g.$$

Wie wir im vorhergehenden Abschnitt gesehen haben, gibt es einen Algorithmus, der eine gegebene Menge von Polynomen zu einer Gröbnerbasis macht, indem zu dieser Menge gewisse weitere Polynome (nämlich die (Pseudo-) Normalformen von S-Polynomen) hinzugenommen werden. Offenbar erhalten wir dadurch eine Gröbnerbasis, in der gewisse Polynome überflüssig sein können. Der Algorithmus **minimalBasis** transformiert eine gegebene Gröbnerbasis in eine minimale Gröbnerbasis, indem redundante Polynome einfach entfernt werden.

Lemma 4.4. *Der Algorithmus* **minimalBasis** *terminiert und erfüllt seine Spezifikation.*

Beweis. Die Termination ist klar, da B und damit F endlich ist und sich in jeder <u>while</u>-Iteration die Kardinalität von F verringert. Wegen $G \subseteq B$ ist G nennerfrei. Setzen wir $I := \mathcal{I}(B)$, so ist Punkt (iv) aus Theorem 3.9 eine Invariante der <u>while</u>-Schleife, d. h.,

$$\text{für jedes } s \in \mathrm{LT}(I) \text{ existiert ein } t \in \mathrm{LT}(G) \text{ mit } t|s.$$

Zu Beginn der while-Schleife gilt diese Relation auf Grund unserer Definition einer Gröbnerbasis. Wenn an der Stelle (G2) ein Polynom aus G entfernt wird, so ist klar, daß wegen Transitivität der Teilbarkeitsrelation obige Relation erhalten bleibt. Zusammen mit Theorem 3.9 folgt, daß der Algorithmus eine Gröbnerbasis zurückliefert.

Offenbar gibt es am Ende des Algorithmus keine zwei Polynome $g_1, g_2 \in G$, so daß $\mathrm{LT}(g_1) | \mathrm{LT}(g_2)$. Die Menge G ist also eine minimale Gröbnerbasis. □

Der Algorithmus **groebner** sei die Nacheinanderausführung der Algorithmen **buchberger** und **minimalBasis**.

4.4 Reduzierte Gröbnerbasis

Im folgenden sei Q der Quotientenkörper des euklidischen Rings R.

Da die Elemente einer Gröbnerbasis nicht eindeutig bestimmt sind, wird der Begriff einer reduzierten Gröbnerbasis eingeführt. Eine reduzierte Gröbnerbasis enthält nur Polynome, deren Leitkoeffizienten gleich 1 sind und die bezüglich der restlichen Polynome der Basis nicht weiter reduziert werden können. Wir wollen diesen Begriff nicht weiter präzisieren und geben statt dessen die folgende Definition.

Definition 4.5. Eine endliche Teilmenge G von $Q[X]$ heißt **nennerfrei-reduziert**, wenn sie folgende Bedingungen erfüllt:

(i) G ist nennerfrei, d. h. $G \subset R[X]$,

(ii) G enthält nur primitive Polynome, insbesondere enthält G nicht das Nullpolynom, und

(iii) kein $g \in G$ ist pseudo-reduzierbar modulo $G \setminus \{g\}$.

Ist G eine Gröbnerbasis, so heißt sie eine **nennerfrei-reduzierte Gröbnerbasis**.

Bemerkung. Im Gegensatz zu einer reduzierten Gröbnerbasis sind die Elemente einer nennerfrei-reduzierten Gröbnerbasis nur bis auf Einheiten von R eindeutig bestimmt. Eine reduzierte Gröbnerbasis im Sinne von [1] kann erhalten werden, indem jedes Polynom einer nennerfrei-reduzierten Gröbnerbasis durch seinen Leitkoeffizienten geteilt wird. Es ist klar, daß die so erhaltene Menge dasselbe Ideal in $Q[X]$ erzeugt, da die Leitkoeffizienten Einheiten dieses Ringes sind.

In der Literatur treten für reduziert auch die Begriffe autoreduziert und interreduziert auf.

Algorithmus autoReduce
Eingabe: B = endliche Teilmenge von $R[X]$
Ausgabe: G = endliche Teilmenge von $R[X]$, so daß G nennerfrei-reduziert und $\mathcal{I}(G) = \mathcal{I}(B)$ in $Q[X]$ ist

begin
$G := \{b \in B \mid b \neq 0\}$
while es gibt ein $g \in G$, das pseudo-reduzierbar ist modulo $G \setminus \{g\}$ do
    wähle $g \in G$, das pseudo-reduzierbar ist modulo $G \setminus \{g\}$
    $G := G \setminus \{g\}$
    $(g, r, H) :=$ **pseudoNormalForm**(g, G)
    if $g \neq 0$ then $G := G \cup \{g\}$ end if
end while
$G := \{\text{primitivePart}(g) \mid g \in G\}$
return G
end autoReduce

Tabelle 4.4: Algorithmus **autoReduce**

Wir geben mit **autoReduce** einen Algorithmus an, der aus einer gegebenen Basis $B \subset R[X]$ eine nennerfrei-reduzierte Menge G derart bestimmt, daß $\mathcal{I}(B) = \mathcal{I}(G)$.

Lemma 4.6. *Der Algorithmus* **autoReduce** *terminiert und erfüllt seine Spezifikation.*

Beweis. Der Algorithmus **autoReduce** ist eine nennerfreie Formulierung des Algorithmus REDUCTION in [1]. Termination und Korrektheit folgen mit analogen Argumenten wie in [1]. □

Wenden wir **autoReduce** auf eine Gröbnerbasis an, so erhalten wir eine nennerfrei-reduzierte Gröbnerbasis.

4.5 Variationen des Buchberger-Algorithmus

Der Buchberger-Algorithmus wurde von uns in einer einfachen Form präsentiert. Für theoretische Zwecke ist dies ausreichend. Ebenso reicht diese Form, um das Prinzip einer Problemreduzierung, die wir später vorstellen wollen, aufzuzeigen. Für die Praxis ist ein günstiges Laufzeitverhalten von Interesse.

Mit der Implementierung des Buchberger-Algorithmus in verschiedensten Programmen wurde versucht, immer größere Probleme in Angriff zu nehmen. Doch bereits bei relativ kleinen Eingabegrößen, d. h. wenigen Polynomen in wenigen Unbestimmten, wurden Grenzen hinsichtlich vorhandener Rechnerkapazitäten erreicht (siehe etwa [4]). Dies liegt am exponentiellen Laufzeit- und Speicherplatzverhalten des Algorithmus in Abhängigkeit von den Eingabegrößen. Im schlechtesten Fall ist er sogar doppelt-exponentiell in Abhängigkeit von der Anzahl der Variablen. Aus diesem Grunde wurden verstärkt Untersuchungen betrieben, um trotz dieser Schwierigkeiten, größere Probleme lösen zu können. Bereits Buchberger gab in [2] Kriterien an, mit deren Hilfe man, ohne eine „teure“ Reduktion eines entstandenen S-Polynoms auszuführen, entscheiden kann, ob dieses Polynom zu Null reduzierbar ist und somit nicht weiter betrachtet werden muß.

Viele Anstrengungen wurden unternommen, um eine optimale Auswahlstrategie der kritischen Paare zu erreichen, da festgestellt wurde, daß man damit die Laufzeit des Algorithmus entscheidend beeinflussen kann. Neben vielen anderen ist hier vor allem die Arbeit [7] von Giovini zu nennen.

Eine andere Herangehensweise ist die Aufteilung des Ausgangsproblems in „kleinere“ Teilprobleme. Eine solche Teile-und-Herrsche-Strategie kann, wie etwa in [14] beschrieben, durch Faktorisierung oder arithmetische Nebenbedingungen (etwa: Lösungen sollen positiv sein) erreicht werden.

Wir konzentrieren uns auf die Problemaufteilung mittels Faktorisierung. Gräbe stellt in [8] eine Form des Buchberger-Algorithmus mit Faktorisierung vor, den wir hier benutzen und ebenfalls mit **FGB** bezeichnen.

Sei K ein algebraisch abgeschlossener Körper. Vorgegeben seien eine Menge $B = \{b_1, \ldots, b_p\} \subset K[A]$ von Polynomen in den Unbestimmten $A_1, \ldots, A_n$ und eine Menge $C = \{c_1, \ldots, c_q\} \subset K[A]$ von „Einschränkungen“. Der Algorithmus **FGB** bestimmt die Nullstellenmenge $\mathcal{Z}(B, C)$, indem er eine gewisse Anzahl von Gröbnerbasen B_i und zugehörige „Einschränkungen“ C_i zurückliefert, so daß

$$\mathcal{Z}(B, C) = \bigcup_i \mathcal{Z}(B_i, C_i)$$

gilt. Dabei bezeichne

$$\mathcal{Z}(B, C) = \mathcal{Z}(B) \setminus \mathcal{Z}(c) \qquad (\subseteq \mathcal{A})$$

mit

$$c = \prod_{f \in C} f$$

die **relative Nullstellenmenge von** B **bzgl.** C **in** $\mathcal{A} = K^n$.

Bemerkung. Bezeichne c das Produkt aller Polynome von C. In geometrischer Sprechweise beschreibt $\mathcal{Z}(B, C)$ den Durchschnitt der „algebraischen Menge“ $\mathcal{Z}(B)$ mit der „offenen Hauptmenge“ $\mathcal{D}(c)$ von $\mathcal{A}$. Dies verdeutlicht, warum wir hier den Begriff „Einschränkung“ verwenden.

Kapitel 5

Gröbnerbasen und Spezialisierung

Im folgenden sei K einen algebraisch abgeschlossenen Körper. $R = K[A]$ bezeichne den Polynomring in den Unbestimmten $A_1, \ldots, A_n$ und Q dessen Quotientenkörper.

Sei $\mathfrak{m}$ ein maximales Ideal von R. $R/\mathfrak{m}$ ist isomorph zu K und wird vermöge dieses Isomorphismus mit K identifiziert. Weiterhin sei $\sigma : R \to R/\mathfrak{m} \cong K$ die natürliche Projektion auf den Restklassenkörper, die wir **Spezialisierung** nennen. Mit

$$a_i := \sigma(A_i) \qquad (i = 1, \ldots, n)$$

schreiben wir für ein $f = f(A_1, \ldots, A_n) \in R$ statt $\sigma(f(A_1, \ldots, A_n))$ auch einfach $f(a_1, \ldots, a_n)$ oder kürzer $f(a)$.

σ läßt sich kanonisch zu einer Abbildung $\bar{\sigma} : R[X] \to K[X]$ erweitern. Für ein Polynom

$$f = \sum_{i=1}^{l} r_i(A) t_i \in R[X]$$

sei

$$f^\sigma := \bar{\sigma}(f) := \sum_{i=1}^{l} \sigma(r_i(A)) t_i = \sum_{i=1}^{l} r_i(a) t_i.$$

Wenn $B = \{b_1, \ldots, b_p\}$, so bezeichne B^σ die Menge $\{b_1^\sigma, \ldots, b_p^\sigma\}$.

Die Aussagen in diesem Kapitel sind grundlegend für die Formulierung eines Algorithmus, der im nächsten Kapitel vorgestellt wird. Auf Grund von Theorem 5.2 und der Folgerung 5.4 können nämlich gewisse Anweisungen vermieden werden, die bei einer naiven Formulierung jenes Algorithmus auftreten würden.

Um den Beweis des Theorems 5.2 zu vereinfachen, stellen wir ein Lemma voran.

Lemma 5.1. *Sei $G = \{g_1, \ldots, g_k\} \subset R[X]$ eine nennerfreie Gröbnerbasis von $\mathcal{I}(G)$ in $Q[X]$, $\sigma : R \to K$ eine Spezialisierung und $f \in \mathcal{I}(G) \cap R[X]$. Wenn für alle $g \in G$ gilt:* $\mathrm{LC}(g)^\sigma \neq 0$, *so ist $f^\sigma \in \mathcal{I}(G^\sigma)$.*

Beweis. Da $f \in \mathcal{I}(G)$ und G eine Gröbnerbasis ist, folgt $f \to_G^* 0$. Betrachten wir nun den Algorithmus **pseudoNormalForm** mit der Eingabe f und G, so erhalten wir eine Relation

$$rf = \sum_{i=1}^{k} h_i g_i, \tag{5.1}$$

mit $h_1, \ldots, h_k \in R[X]$ und $r \in R$.

Für diese Abarbeitung des Algorithmus zeigen wir, daß die Relation $r^\sigma \neq 0$ eine Invariante der while-Schleife ist. Offenbar gilt auf Grund der Zuweisung $r := 1$ obige Relation zu Beginn der while-Schleife. Betrachten wir im Schleifenkörper nun die Wahl von $\tilde{r}$, so erkennen wir gemäß Definition 3.7, daß $\tilde{r} | \mathrm{LC}(g)$ für ein gewisses $g \in G$. Nutzen wir die Voraussetzung $\mathrm{LC}(g)^\sigma \neq 0$, so folgt $\tilde{r}^\sigma \neq 0$ und damit auch $\sigma(\tilde{r}r) \neq 0$.

Damit liefert die Relation (5.1) eine Darstellung

$$f^\sigma = \frac{1}{r^\sigma} \left(\sum_{i=1}^{k} {h_i}^\sigma {g_i}^\sigma \right).$$

Folglich gilt $f^\sigma \in \mathcal{I}(G^\sigma)$, und das Lemma ist bewiesen. □

Theorem 5.2. *Sei $B = \{b_1, \ldots, b_p\} \subset R[X]$ und $\sigma : R \to K$ eine Spezialisierung. Erzeuge B in $Q[X]$ das echte Ideal $\mathcal{I}(B)$ und sei $G = \{g_1, \ldots, g_k\}$ eine minimale nennerfreie Gröbnerbasis von $\mathcal{I}(B)$.*

Wenn $\mathrm{LC}(g_i)^\sigma \neq 0$, *d. h.* $\mathrm{LC}(g_i)^\sigma = \mathrm{LC}({g_i}^\sigma)$ *und* $\mathrm{LT}(g_i) = \mathrm{LT}({g_i}^\sigma)$ *für alle $i = 1, \ldots, k$, dann erzeugt G^σ ein echtes Ideal in $K[X]$ und es gilt*

$$\mathcal{I}(B^\sigma) \subseteq \mathcal{I}(G^\sigma) \qquad (\subset K[X]). \tag{5.2}$$

Beweis. $\mathcal{I}(G^\sigma)$ ist ein echtes Ideal von $K[X]$, da $\mathrm{LT}(G) = \mathrm{LT}(G^\sigma)$ und G eine minimale Gröbnerbasis ist.

Für jedes $b \in B$ gilt $b \in \mathcal{I}(G)$, da G Gröbnerbasis von $\mathcal{I}(B)$ ist. Nach Lemma 5.1 folgt dann $b^\sigma \in \mathcal{I}(G^\sigma)$ und somit auch $\mathcal{I}(B^\sigma) \subseteq \mathcal{I}(G^\sigma)$. □

Definition 5.3. Das **Nullstellengebilde eines Ideals** $I \subseteq K[X]$ ist die Menge

$$\mathcal{Z}(I) := \{x \in \mathcal{X} \mid f(x) = 0 \text{ für alle } f \in I\}.$$

Der HILBERTsche Basissatz sichert, daß jedes Ideal $I \subseteq K[X]$ ein endliches Erzeugendensystem B besitzt. Offenbar gilt $\mathcal{Z}(I) = \mathcal{Z}(B)$, da jedes Element von I eine $K[X]$-Linearkombination von Elementen von B ist.

Als einfache Folgerung aus dem HILBERTschen Nullstellensatz haben wir: Wenn I ein echtes Ideal in $K[X]$ ist, so gilt

$$\emptyset \neq \mathcal{Z}(I) \subseteq \mathcal{X} = K^m.$$

Damit erhalten wir eine Folgerung aus dem letzten Theorem.

Folgerung 5.4. *Mit den gleichen Voraussetzungen wie im Theorem 5.2 gilt*

$$\emptyset \neq \mathcal{Z}(G^\sigma) \subseteq \mathcal{Z}(B^\sigma) \subseteq \mathcal{X},$$

d. h., die Nullstellenmenge von B^σ ist nicht leer.

Kapitel 6

Ein neuer Algorithmus

In diesem Kapitel beschreiben wir zuerst die Motivation, die zur Untersuchung eines neuen Verfahrens geführt hat, geben anschließend den Algorithmus **gamma** an und verallgemeinern am Ende seine Spezfikation, um einen größeren Anwendungsbereich abdecken zu können.

Der Körper K sei in diesem Kapitel algebraisch abgeschlossen.

6.1 Herkunft und Entstehung

Ausgangspunkt der hier vorgestellten Untersuchung ist die vollständige Lösung der konstanten Quanten-Yang-Baxter-Gleichung im zweidimensionalen Fall von Hietarinta in [11]. Die dort verwendete Lösungsmethode wird hier teilweise formalisiert und verallgemeinert und gibt schließlich Anlaß zu einem Algorithmus, der sich dann auch auf andere Probleme anwenden läßt.

Um zu dieser vollständigen Lösung zu gelangen, wurde in [11] verstärkt die Eigenschaft ausgenutzt, daß sich unter gewissen Voraussetzungen einige der auftretenden Polynome in Faktoren zerlegen lassen. Diese Voraussetzungen konnten erfüllt werden, indem zusätzlich bereits bekannte Symmetrien ausgenutzt wurden, bei denen sich die Lösungsmenge nicht ändert.

Hietarinta verwendete „diskrete“ Symmetrien, die darin bestanden, daß gewisse Variablen mit anderen vertauscht werden konnten. Außerdem benutzte er eine von Parametern abhängige Transformation, die ebenfalls die Lösungsmenge invariant ließ. Im Laufe der Rechnung wurden diese freien Parameter auf eine Weise fixiert, die es ermöglichte, (durch Faktorisierung gewisser Polynome) das Ausgangsproblem in eine Reihe von „einfacheren“ Problemen zu zerlegen.

Hietarintas Idee bestand darin, anstatt nach der vollen Lösungsmenge nur nach „leicht zu findenden“ Repräsentanten der Symmetrieklassen zu suchen und dann mit Hilfe der Symmetrietransformationen alle Lösungen darzustellen.

Unser Ansatz nimmt die obige Idee als Grundlage. Zunächst denken wir uns Polynome gegeben, die Symmetrien der gesuchten Lösungsmenge beschreiben. (Für die genaue Problemformulierung verweisen wir auf den Abschnitt 2.1.) „Leicht zu finden“ wird bei uns bedeuten, daß wir mit Hilfe der Berechnung einer Gröbnerbasis solche Repräsentanten aus der Lösungsmenge wählen, für die gewisse Koordinaten verschwinden.

6.2 Der Algorithmus gamma

In diesem Abschnitt geben wir einen Algorithmus an, der eine gewisse Gruppenaktion ausnutzt, um das Ausgangsproblem, die Menge $\mathcal{Z}(B)$ der Nullstellen von B zu beschreiben, in mehrere „einfachere“ Probleme zu zerlegen.

Bevor wir zum Beweis der Korrektheit des Algorithmus **gamma** kommen, werden wir erst einige Bemerkungen zu seinem Aufbau machen.

Die Menge T beschreibt die Koordinaten, die zum Verschwinden gebracht werden. Es erscheint günstig, l so groß wie möglich zu wählen. Da wir m Parameter haben, nämlich $x_1, \dots, x_m$, sollte möglichst $l = m$ gewählt werden. Allerdings mag das nicht immer möglich sein.

Die Menge $\{\delta_1, \dots, \delta_l\}$ drückt gerade die Gruppenaktion auf den Koordinaten aus, die zum Verschwinden gebracht werden sollen. Für jedes $a \in \mathcal{A}$ untersuchen wir, ob ein $x \in \mathcal{X}$ existiert, so daß

$$\delta_1(x, a) = \cdots = \delta_l(x, a) = 0$$

gilt. Ist dies der Fall, so läßt sich dieses a mittels der Gruppenaktion aus einem Element von $\mathcal{A}$ konstruieren, dessen zu $\delta_1, \dots, \delta_l$ gehörigen Koordinaten verschwinden. Für ein festes a existiert ein solches x genau dann, wenn

$$\Delta(a) := \{\delta_1(X, a), \dots, \delta_l(X, a)\}$$

Erzeugendensystem eines echten Ideals von $K[X]$ ist. Die Echtheit dieses Ideals kann einfach festgestellt werden, wenn dessen Gröbnerbasis G (bzgl. einer beliebigen Termordnung) bekannt ist. Wenn $1 \in G$, so liegt kein echtes Ideal vor, und umgekehrt.

Um die Gröbnerbasisberechnung nicht für jedes a gesondert ausführen zu müssen, bestimmen wir eine Gröbnerbasis des von $\Delta := \{\delta_1, \dots, \delta_l\}$ in $Q[X]$ erzeugten Ideals, wobei Q den Quotientenkörper von $R = K[A]$ bezeichnet. Das Theorem 5.2 gibt hinreichende Bedingungen dafür an, daß nach einer Spezialisierung σ von Δ die Menge $\Delta^\sigma = \Delta(a)$ ein echtes Ideal in $K[X]$ erzeugt.

Algorithmus gamma
Eingabe:

- $B = \{b_1, \ldots, b_p\}$ endliche Teilmenge von $R = K[A]$
- $T = \{(\delta_1, d_1), \ldots, (\delta_l, d_l)\} \subseteq \{(\gamma_1, A_1), \ldots, (\gamma_n, A_n)\} \subseteq R[X] \times R$. Transformationspolynome zusammen mit den zugehörigen Variablen, die dann „verschwinden".
- Γ durch $\mathcal{X} = K^m$ parametrisierte Gruppe, die auf $\mathcal{A} = K^n$ operiert und dabei $\mathcal{Z}(B)$ invariant läßt. Die Gruppenaktion sei vermittelt durch $\gamma_1, \ldots, \gamma_n$, d. h., für alle $x \in \mathcal{X}$ und $a \in \mathcal{A}$ gilt:

$$\gamma(x) \cdot a = (\gamma_1(x, a), \ldots, \gamma_n(x, a)) \in \mathcal{A}.$$

Ausgabe:

- $S_\nu = \{(B_i, C_i)\}_{i \in I_\nu}$ Gröbnerbasen mit zugehörigen Einschränkungen für gewisse endliche Indexmengen I_0 und I_1 mit $I_0 \cap I_1 = \emptyset$, so daß

$$\mathcal{Z}(B) = \bigcup_{i \in I_0} \mathcal{Z}(B_i, C_i) \cup \bigcup_{i \in I_1} \operatorname{Orb}(\mathcal{Z}(B_i, C_i)).$$

  Dabei sind alle B_i und C_i endliche Teilmengen von $K[A]$, und der Orbit sei bezüglich Γ gebildet.

<u>begin</u>
$G :=$ **groebner**$(\{\delta_1, \ldots, \delta_l\})$, (*verwende* $R = K[A]$)
 (G ist jetzt eine minimale nennerfreie Gröbnerbasis.)
<u>if</u> $\mathcal{I}(G) = Q[X]$ <u>then</u>
 (Schlechte Wahl von T, d. h. keine Verwendung der)
 (Gruppenaktion und damit keine Problemreduktion.)
 $S_0 :=$ **FGB**$(B, \emptyset)$
 $S_1 := \emptyset$
<u>else</u>
 $r_1^{\nu_1} \cdot \cdots \cdot r_u^{\nu_u} = \prod_{g \in G} \mathrm{LC}(g)$ (Z1)
 (Zerlegung in (irreduzible) Faktoren in $R = K[A]$*)*
 $S_0 := \bigcup_{i=1}^{u}$ **FGB**$(B \cup \{r_i\}, \{r_1, \ldots, r_{i-1}\})$
 $S_1 :=$ **FGB**$(B \cup \{d_1, \ldots, d_l\}, \{r_1, \ldots, r_u\})$
<u>end if</u>
<u>return</u> (S_0, S_1)
<u>end gamma</u>

Tabelle 6.1: Algorithmus **gamma**

Theorem 5.2 liefert aber sogar noch mehr. Ein naiver Algorithmus müßte nämlich alle Leitkoeffizienten der Polynome aufsammeln, die während der Gröbnerbasisberechnung von $\{\delta_1, \ldots, \delta_l\}$ entstehen, damit gesichert ist, daß im Algorithmus **buchberger** beim Übergang von einem Polynom g zu seinem primitiven Teil primitivePart(g) keine Informationen verlorengehen und keine falsche Entscheidung über die Existenz eines x für ein gewisses a getroffen wird. Das Theorem 5.2 gestattet, ein solches Aufsammeln zu vermeiden, und zeigt, daß es sogar ausreicht, nur die Leitkoeffizienten einer minimalen Gröbnerbasis zu betrachten.

An der Stelle (Z1) wird das Produkt der Leitkoeffizienten, welches ein Element von $K[A]$ ist, in (verschiedene) irreduzible Faktoren zerlegt. Dies ist jedoch eine ineffiziente Formulierung. Bei einer Implementation von **gamma** ist es vorteilhafter, die Leitkoeffizienten einzeln über $K[A]$ zu faktorisieren (nicht deren Produkt) und aus den einzelnen Faktoren die gewünschte Zerlegung zu konstruieren.

Mit der Menge S_1 wird (zusammen mit der Gruppenaktion von Γ) gerade der Teil der gesuchten Nullstellenmenge $\mathcal{Z}(B)$ beschrieben, für den kein Leitkoeffizient von G verschwindet. Mit anderen Worten, wenn $(B_0, C_0) \in S_1$, $(a_1, \ldots, a_n) \in \mathcal{Z}(B_0, C_0)$ und $\sigma : K[A] \to K$ die Spezialisierung ist, die A_i in a_i $(i = 1, \ldots, n)$ überführt, so gilt $\mathrm{LC}(g)^\sigma \neq 0$ für alle $g \in G$. In diesem Fall kann Γ erfolgreich zur Problemreduzierung genutzt werden.

Andernfalls können wir die Ausgangsmenge B jeweils um ein weiteres Polynom ergänzen. Im Algorithmus sind dies die Polynome $r_1, \ldots, r_u$. Geometrisch beschreibt $\mathcal{Z}(B \cup \{r_i\})$ den Durchschnitt von $\mathcal{Z}(B)$ mit der Hyperfläche, die durch $r_i = 0$ in $\mathcal{A}$ gegeben ist.

Ebenso wie der Buchberger-Algorithmus mit Faktorisierung beschreibt der Algorithmus **gamma** nur eine Heuristik. In gewissen Fällen werden dadurch Verbesserungen erreicht, in anderen Fällen kann die zusätzliche Gröbnerbasisberechnung der Transformationspolynome jedoch auch einen Mehraufwand bedeuten.

Lemma 6.1. *Der Algorithmus* **gamma** *terminiert und erfüllt seine Spezifikation.*

Beweis. Die Termination von **gamma** ist offensichtlich. Die Korrektheit folgt aus der von **FGB**. Wird nämlich der <u>then</u>-Zweig durchlaufen, so ist **gamma** im wesentlichen mit **FGB** identisch. Betrachten wir nun den <u>else</u>-Zweig. S_ν ist eine Menge von Paaren. Sei etwa $S_\nu = \{(B_i, C_i)\}_{i \in I_\nu}$ $(\nu = 0, 1)$ für gewisse endliche Indexmengen I_0 und I_1 mit $I_0 \cap I_1 = \emptyset$. Wir müssen

$$\mathcal{Z}(B) = \bigcup_{i \in I_0} \mathcal{Z}(B_i, C_i) \cup \bigcup_{i \in I_1} \mathrm{Orb}(\mathcal{Z}(B_i, C_i)) \tag{6.1}$$

beweisen.

Nach Spezifikation von **FGB** haben wir die Identitäten

$$\bigcup_{i \in I_0} \mathcal{Z}(B_i, C_i) = \bigcup_{i=1}^{u} \mathcal{Z}(B \cup \{r_i\}, \{r_1, \ldots, r_{i-1}\}), \tag{6.2}$$

$$\bigcup_{i \in I_1} \mathcal{Z}(B_i, C_i) = \mathcal{Z}(B \cup \{d_1, \ldots, d_l\}, \{r_1, \ldots, r_u\}), \tag{6.3}$$

die den letzten beiden Zeilen im <u>else</u>-Zweig entsprechen.

Die Inklusion „$\supseteq$" in (6.1) folgt aus obigen Gleichungen zusammen mit Eigenschaften von Nullstellengebilden und der Tatsache, daß $\mathcal{Z}(B)$ invariant unter der Gruppenaktion von Γ ist. Wir haben

$$\begin{aligned} \mathcal{Z}(B) &\supseteq \bigcup_{i=1}^{u} \mathcal{Z}(B \cup \{r_i\}) \\ &\supseteq \bigcup_{i=1}^{u} \mathcal{Z}(B \cup \{r_i\}, \{r_1, \ldots, r_{i-1}\}) \\ &= \bigcup_{i \in I_1} \mathcal{Z}(B_i, C_i) \end{aligned}$$

und

$$\begin{aligned} \mathcal{Z}(B) &\supseteq \mathcal{Z}(B \cup \{d_1, \ldots, d_l\}) \\ &\supseteq \mathcal{Z}(B \cup \{d_1, \ldots, d_l\}, \{r_1, \ldots, r_u\}) \\ &= \bigcup_{i \in I_0} \mathcal{Z}(B_i, C_i). \end{aligned}$$

Um die umgekehrte Inklusion zu zeigen, müssen wir angeben, wie sich ein Element $a = (a_1, \ldots, a_n) \in \mathcal{Z}(B)$ durch S_0 oder S_1 beschreiben läßt.

Gilt $r_j(a) = 0$ für ein gewisses $j \in \{1, \ldots, u\}$, so haben wir

$$a \in \bigcup_{i=1}^{u} \mathcal{Z}(B \cup \{r_i\}, \{r_1, \ldots, r_{i-1}\})$$

und wegen (6.2) wird dieses a durch S_0 erfaßt.

Betrachten wir im Algorithmus die Stelle (Z1) und nehmen nun an, daß $r_i(a) \neq 0$ für alle $i = 1, \ldots, u$ ist, so heißt dies für die Spezialisierung $\sigma : K[A] \to K$, bei der A_i in a_i $(i = 1, \ldots, n)$ überführt wird, daß $\mathrm{LC}(g)^\sigma \neq 0$ für alle $g \in G$ gilt.

Damit können wir Theorem 5.2 anwenden und erhalten

$$\mathcal{I}(\delta_1^\sigma, \ldots, \delta_l^\sigma) \subseteq \mathcal{I}(G^\sigma) \subset K[X].$$

Die Folgerung 5.4 sagt dann aus, daß es ein $x \in \mathcal{X}$ gibt mit

$$\delta_1(x, a) = \cdots = \delta_l(x, a) = 0.$$

Dies heißt aber, daß in $\mathrm{Orb}(a)$ ein Element liegt, bei dem die Koordinaten verschwinden, die zu $\delta_1, \ldots, \delta_l$ gehören. Es reicht also, ein solches Element zu bestimmen, da mit einem Element aus dem Orbit der ganze Orbit bestimmt ist. Diese verschwindenden Koordinaten werden aber gerade durch $d_1, \ldots, d_l$ repräsentiert. Folglich wird a durch die Menge S_1 beschrieben. □

6.3 Verallgemeinerung

Betrachten wir den Algorithmus **gamma** genauer, so können wir feststellen, daß wir die Invertierbarkeit eines Elements von Γ nur wirklich benötigen, um von der Menge S_1 auf die vollständige Lösung zu schließen.

In der von uns eingeführten Terminologie heißt das, daß es zu jedem Gruppenelement $\gamma(x)$ ein inverses Element $\gamma(x')$ gibt ($x, x' \in \mathcal{X}$), so daß für jedes $a \in \mathcal{Z}(B)$

$$\gamma(x') \cdot \gamma(x) \cdot a = a$$

gilt. Benutzen wir die zu den Transformationen gehörigen Polynome, so geht obige Gleichung über in

$$\gamma_i(x', \gamma(x, a)) = a_i \qquad (i = 1, \ldots, n).$$

Mit dieser Vorüberlegung läßt sich die Voraussetzung bezüglich Γ etwas verallgemeinern. In der ursprünglichen Aufgabenstellung verlangen wir nur noch die Existenz von Polynomen $\gamma_1, \ldots, \gamma_n \in K[X_1, \ldots, X_m, A_1, \ldots, A_n]$, so daß für alle $x \in \mathcal{X}$ und $a \in \mathcal{Z}(B) \subseteq \mathcal{A}$ gilt:

$$\gamma(x, a) := (\gamma_1(x, a), \ldots, \gamma_n(x, a)) \in \mathcal{Z}(B).$$

Diese Polynome werden genauso wie früher benutzt.

Um nun von der erhaltenen Lösung S_1 wieder auf den gesamten Lösungsraum schließen zu können, müssen wir zusätzlich die Existenz von Polynomen $\gamma'_1, \ldots, \gamma'_n \in K[X_1, \ldots, X_m, A_1, \ldots, A_n]$ fordern, die die Transformation der γ_i wieder „rückgängig" machen, d. h., für alle $x \in \mathcal{X}$ und $a \in \mathcal{Z}(B)$ und für $1 \leq i \leq n$ gelte:

$$\gamma'_i(x, \gamma(x, a)) = a_i. \tag{6.4}$$

Wir ersetzen nun in der Spezifikation des Algorithmus **gamma** den Eingabeparameter Γ durch:

- $\Gamma = (\gamma_1, \ldots, \gamma_n) \subset K[X, A]$, n-Tupel von Polynomen, so daß für alle $x \in \mathcal{X}$ und $a \in \mathcal{Z}(B) \subseteq \mathcal{A}$ gilt:

$$\gamma(x, a) := (\gamma_1(x, a), \ldots, \gamma_n(x, a)) \in \mathcal{Z}(B).$$

- $\Gamma' = (\gamma'_1, \ldots, \gamma'_n) \subset K[X, A]$, n-Tupel von Polynomen, so daß für alle $x \in \mathcal{X}$ und $a \in \mathcal{Z}(B)$ und für $1 \leq i \leq n$ gilt:

$$\gamma'_i(x, \gamma(x, a)) = a_i.$$

Die Ausgabespezifikation werde ersetzt durch:

- $S_\nu = \{(B_i, C_i)\}_{i \in I_\nu}$ Gröbnerbasen mit zugehörigen Einschränkungen für gewisse endliche Indexmengen I_0 und I_1 mit $I_0 \cap I_1 = \emptyset$, so daß

$$\mathcal{Z}(B) = \bigcup_{i \in I_0} \mathcal{Z}(B_i, C_i) \cup \bigcup_{i \in I_1} (\mathcal{Z}(B_i, C_i))^{\Gamma'}.$$

  Dabei sind alle B_i und C_i endliche Teilmengen von $K[A]$, und für eine Teilmenge $Z \subseteq \mathcal{A}$ sei

$$Z^{\Gamma'} := \{(\gamma'_1(x, a), \ldots, \gamma'_n(x, a)) \mid x \in \mathcal{X}, a \in Z\}.$$

Wir bezeichnen den Algorithmus, der durch diese Spezifikationsänderung entsteht, ebenso mit **gamma**. Die Korrektheit dieses Algorithmus ergibt sich aus dem Beweis von Lemma 6.1, wenn dort der Bezug auf die Gruppenaktion durch die Transformationen Γ und Γ' ersetzt wird.

Kapitel 7

Beispiele

In diesem Kapitel stellen wir dar, wie bei der Frage nach allen Automorphismen einer vorgegebenen Lie-Algebra ein Kontext entsteht, der eine Anwendung des Algorithmus **gamma** ermöglicht, und untersuchen einige solcher Beispiele.

7.1 Automorphismen von Lie-Algebren

Zur besseren Lesbarkeit werden wir in diesem Abschnitt die Einsteinsche Summenkonvention benutzen, d. h. Summation über doppelt auftretende Indizes von 1 bis N.

Außerdem werden wir Begriffe aus der Theorie der Lie-Algebren verwenden, wie sie etwa in [12] oder [13] gefunden werden können.

Sei V ein N-dimensionaler Vektorraum über einem algebraisch abgeschlossenen Körper K und sei $\{e_1, \ldots, e_N\}$ eine Basis von V. Weiterhin sei in V eine Lie-Klammer $[?, ?]$ durch die Angabe von Strukturkonstanten $c_{jk}^l \in K$ $(1 \leq j, k, l \leq N)$ gegeben, so daß

$$[e_j, e_k] = c_{jk}^l e_l \qquad (1 \leq j, k \leq N) \tag{7.1}$$

gilt und V damit zu einer Lie-Algebra wird. Ein Endomorphismus dieser Lie-Algebra ist eine lineare Transformation $\hat{A} : V \to V$, die mit der Lie-Klammer verträglich ist, d. h.

$$\forall v, v' \in V : \ \hat{A}[v, v'] = [\hat{A}v, \hat{A}v']. \tag{7.2}$$

Einem solchen Endomorphismus $\hat{A}$ läßt sich bezüglich der gegebenen Basis eineindeutig eine Matrix

$$A = \begin{pmatrix} A_1^1 & \ldots & A_N^1 \\ \vdots & & \vdots \\ A_1^N & \ldots & A_N^N \end{pmatrix}$$

zuordnen, so daß für $j = 1, \ldots, N$ gilt:

$$\hat{A}e_j = A_j^k e_k.$$

Die Relation (7.2) gilt auf Grund der Bilinearität der Lie-Klammer genau dann, wenn sie für alle möglichen Paare von Basisvektoren (e_j, e_k) gilt. Aus $\hat{A}[e_j, e_k] = [\hat{A}e_j, \hat{A}e_k]$ und den Relationen

$$\hat{A}[e_j, e_k] = \hat{A}(c_{jk}^m e_m) = c_{jk}^m \hat{A}e_m = A_m^l c_{jk}^m e_l,$$
$$[\hat{A}e_j, \hat{A}e_k] = [A_j^p e_p, A_k^q e_q] = A_j^p A_k^q [e_p, e_q] = A_j^p A_k^q c_{pq}^l e_l$$

erhalten wir durch Koeffizientenvergleich die Gleichungen

$$A_m^l c_{jk}^m = A_j^p A_k^q c_{pq}^l \qquad (1 \leq j, k, l \leq N). \tag{7.3}$$

Diese beschreiben die zu Endomorphismen gehörenden Matrizen. Wegen der Antisymmetrie der Lie-Klammer reicht es, $j < k$ zu betrachten.

Wir möchten alle Lie-Algebra-Automorphismen von V bestimmen. Dies erreichen wir, indem wir das Gleichungssystem (7.3) lösen und alle Lösungen verwerfen, deren zugehörige Determinante verschwindet.

An dieser Stelle wird die Gröbnerbasistechnik eingesetzt. Die A_k^j werden als Unbestimmte betrachtet, und in (7.3) wird alles auf die linke Seite gebracht. Diese linken Seiten liefern eine Menge

$$F := \{A_m^l c_{jk}^m - A_j^p A_k^q c_{pq}^l \mid 1 \leq j, k, l \leq N, j < k\} \tag{7.4}$$

von Polynomen aus $K[A] := K[A_k^j : 1 \leq j, k \leq N]$, deren Nullstellenmenge $\mathcal{Z}(F)$ die gesuchten Endomorphismen beschreibt. Um Aussagen über $\mathcal{Z}(F)$ zu erhalten, ist die Berechnung einer Gröbnerbasis von $\mathcal{I}(F)$ bzgl. einer gewissen Termordnung von Vorteil.

Zur Lösung des Gleichungssystems (7.3) können wir aber zusätzlich noch die Tatsache benutzen, daß die Hintereinanderausführung eines Endo- und eines Automorphismus wieder ein Endomorphismus ist und daß sich der ursprüngliche Endomorphismus auf eine ebensolche Weise wiedergewinnen läßt.

Beschränken wir uns auf den Fall $\operatorname{char} K = 0$, so können wir folgendes Argument aus [13, S. 9f] verwenden und damit eine Menge von Lie-Algebra-Automorphismen konstruieren.

Ist $D : V \to V$ eine Derivation von V, d. h. eine K-lineare Abbildung, so daß für alle $v, v' \in V$ gilt

$$D[v, v'] = [Dv, v'] + [v, Dv'],$$

und ist D außerdem noch nilpotent, d. h. $D^s = 0$ für eine gewisse positive natürliche Zahl s, so ist

$$\exp(D) := \sum_{k=0}^{s} \frac{D^k}{k!}$$

ein Automorphismus von V. Für den einfachen Beweis dieser Aussage verweisen wir auf [13, S. 9f].

Die Zahl s und damit auch die Anzahl der Glieder, die in der Exponentialreihe zu berücksichtigen sind, hängt natürlich von D ab.

Bezüglich der gewählten Basis lassen sich den Derivationen von V (die ja insbesondere K-lineare Abbildungen sind) Matrizen zuordnen.

Aus der Theorie der Lie-Algebren ist bekannt, daß für jedes $v \in V$ die Abbildung $\mathrm{ad}(v) : V \to V$, $v' \mapsto [v, v']$ eine Derivation ist. Sei $\tilde{v}$ die zu $\mathrm{ad}(v)$ gehörige Matrix und sei $\mathrm{ad}(v)$ (und damit auch $\tilde{v}$) nilpotent, etwa $\mathrm{ad}(v)^s = 0$ für $s > 0$. Für jedes $x \in K$ läßt sich dann

$$\exp(x\tilde{v}) := \sum_{k=0}^{s} \frac{(x\tilde{v})^k}{k!}$$

bestimmen.

Da die Matrizen $-x\tilde{v}$ und $x\tilde{v}$ kommutieren, folgt durch explizites Einsetzen und Nachrechnen, daß $\exp(-x\tilde{v})\exp(x\tilde{v}) = \exp(0)$ die Einheitsmatrix ist.

Unter den Basisvektoren e_i von V mögen m Stück derart sein, daß $\mathrm{ad}(e_i)$ nilpotent ist. Wir können ohne Beschränkung der Allgemeinheit annehmen, daß dies $e_1, \ldots, e_m$ sind. Für $m = 0$ liefern die folgenden Betrachtungen nichts Neues, sei deshalb $m \geq 1$.

Für $i = 1, \ldots, m$ bezeichnen wir die Matrix, die zu $\exp(X_i \,\mathrm{ad}(e_i))$ gehört, mit $\Gamma_i(X_i)$. Offenbar ist nach obiger Bemerkung $\Gamma_i(X_i) \cdot \Gamma_i(-X_i)$ die Einheitsmatrix.

Die Matrix Γ sei definiert als

$$\Gamma := \begin{pmatrix} \gamma_1^1 & \cdots & \gamma_N^1 \\ \vdots & & \vdots \\ \gamma_1^N & \cdots & \gamma_N^N \end{pmatrix} := A \cdot \Gamma_1(X_1) \cdot \cdots \cdot \Gamma_m(X_m).$$

Die Einträge von Γ sind Polynome in den A_j^i und X_k, d. h. Elemente von $K[A][X_1, \ldots, X_m]$.

Wir definieren

$$\begin{aligned}\Gamma' &:= \begin{pmatrix} \gamma'^1_1 & \dots & \gamma'^1_N \\ \vdots & & \vdots \\ \gamma'^N_1 & \dots & \gamma'^N_N \end{pmatrix} \\ &:= A \cdot (\Gamma_1(X_1) \cdot \dots \cdot \Gamma_m(X_m))^{-1} \\ &= A \cdot \Gamma_m(-X_m) \cdot \dots \cdot \Gamma_1(-X_1).\end{aligned}$$

Setzen wir noch $n := N^2$ und legen eine Reihenfolge der Indexpaare fest, so erhalten wir Polynome $\gamma_1, \dots, \gamma_n$ und $\gamma'_1, \dots, \gamma'_n$ in den Unbestimmten $A_1, \dots, A_n, X_1, \dots, X_m$. Es ist auf Grund der Konstruktion dieser Polynome klar, daß die Relation (6.4) erfüllt ist.

Patera gibt in [15] eine Liste niedrig-dimensionaler Lie-Algebren an. Wir greifen einige davon heraus und demonstrieren an diesen Beispielen die Wirkungsweise des Algorithmus **gamma**.

Obwohl es prinzipiell möglich ist, bei der Gröbnerbasisberechnung jede beliebige Termordnung zu benutzen, hat die lexikographische für die Bestimmung der Nullstellen gewisse Vorteile. Die Gröbnerbasis bzgl. der lexikographischen Termordnung weist eine „Dreiecksgestalt“ in dem Sinne auf, daß die Anzahl der auftretenden Variablen von Polynom zu Polynom abnimmt. Somit können die Nullstellen in analoger Weise zur Nullstellenbestimmung eines lineare Gleichungssystems, das in Dreiecksgestalt vorliegt, gefunden werden, nur daß in unserem Falle im allgemeinen keine linearen Polynome auftreten.

Da sich alle von uns untersuchten Beispiele mit der lexikographischen Termordnung berechnen ließen, werden wir diese Termordnung hier auch zur Beschreibung benutzen.

Definition 7.1. Seien $X^\alpha, X^\beta \in T(X_1, \dots, X_m)$, wobei $\alpha = (\alpha_1, \dots, \alpha_m)$ und $\beta = (\beta_1, \dots, \beta_m)$ Multiindizes sind. Wir nennen X^α **lexikographisch größer** als X^β und schreiben $X^\alpha \geq_{\text{lex}} X^\beta$ genau dann, wenn $\alpha = \beta$ oder es ein $1 \leq i \leq m$ gibt mit $\alpha_j = \beta_j$ für alle $1 \leq j < i$ und $\alpha_i > \beta_i$.

Um für die Unbestimmten A^i_j eine Reihenfolge festzulegen, führen wir hilfsweise eine Kleiner-Relation $\prec$ ein und ordnen dann die Unbestimmten an, indem wir mit der größten (bzgl. $\prec$) beginnen.

Für $i, j, k, l = 1, \dots, N$ gelte $A^i_j \prec A^k_l$ genau dann, wenn eine der folgenden Bedingungen erfüllt ist.

(i) $|i - j| < |k - l|$

(ii) $|i - j| = |k - l|$ und $i + j < k + l$

(iii) $|i - j| = |k - l|$ und $i + j = k + l$ und $i < k$

Dies entspricht der Absicht, die Indizes in einer solchen Weise anzuordnen, daß die Unbestimmten auf der Hauptdiagonale kleiner sind als alle weiteren.

Wir benutzen obige Reihenfolge, um auf den Termen $T(A^i_j : 1 \leq i, j \leq N)$ die lexikographische Termordnung zu definieren, und werden von nun ab stets eine solche Termordnung verwenden.

7.2 Hilfsmittel

Im allgemeinen sind Berechnungen von Gröbnerbasen so aufwendig, daß es sinnvoll ist, einen Computer zu Hilfe zu nehmen. Dies trifft auch für die Beispiele zu, die wir im folgenden untersuchen werden.

In den letzten Jahren wurde die Entwicklung von Computeralgebrasystemen stark vorangetrieben. In den meisten dieser Systeme ist eine Variante des Buchberger-Algorithmus implementiert, womit die Möglichkeit einer Gröbnerbasisberechnung geboten wird. Unsere Berechnungen werden wir im Computeralgebrasystem REDUCE[1] ausführen. Für die Gröbnerbasisberechnung nutzen wir die Routinen, die in dem REDUCE-Paket CALI[2] implementiert sind. Unter Verwendung einiger Routinen von CALI haben wir den Algorithmus **gamma** in REDUCE implementiert.

7.3 Die Lie-Algebra $A_{4,7}$

K ist in diesem Abschnitt der Körper der komplexen Zahlen. Wir betrachten die 4-dimensionale komplexe Lie-Algebra V, die gegeben ist durch die Relationen

$$\begin{aligned}
[e_2, e_3] &= e_1, \\
[e_1, e_4] &= 2e_1, \\
[e_2, e_4] &= e_2, \\
[e_3, e_4] &= e_2 + e_3
\end{aligned}$$

zwischen den Basisvektoren. Alle anderen Ausdrücke dieser Art, die nicht auf Grund der Antikommutativität und Bilinearität der Lie-Klammer aus obigen Relationen gebildet werden können, mögen verschwinden. In der Notation von Patera in [15] ist dies die Komplexifizierung der reellen Lie-Algebra $A_{4,7}$.

Die Menge F, die wir hier gemäß (7.4) erhalten, ist

[1]Zur Beschreibung der von uns benutzten Version 3.4.1 siehe etwa [10].

[2]Wir verwenden Version 2.2.1. Siehe [9]

$$
\begin{aligned}
F := \{&2A_1^4A_2^1 - 2A_2^4A_1^1 + A_1^3A_2^2 - A_2^3A_1^2,\\
&A_1^4A_2^3 + A_1^4A_2^2 - A_2^4A_1^3 - A_2^4A_1^2,\\
&A_1^4A_2^3 - A_2^4A_1^3,\\
&2A_1^4A_3^1 + A_1^3A_3^2 - 2A_3^4A_1^1 - A_1^2A_3^3,\\
&A_1^4A_3^2 + A_1^4A_3^3 - A_1^3A_3^4 - A_3^4A_1^2,\\
&A_1^4A_3^3 - A_1^3A_3^4,\\
&2A_2^4A_3^1 - 2A_3^4A_2^1 + A_2^3A_3^2 - A_3^3A_2^2 + A_1^1,\\
&A_2^4A_3^2 + A_2^4A_3^3 - A_3^4A_2^3 - A_3^4A_2^2 + A_1^2,\\
&A_2^4A_3^3 + A_1^3 - A_3^4A_2^3,\\
&A_1^4,\\
&2A_1^4A_4^1 + A_4^2A_1^3 - A_4^3A_1^2 - 2A_4^4A_1^1 + 2A_1^1,\\
&A_1^4A_4^2 + A_1^4A_4^3 - A_1^3A_4^4 - A_1^2A_4^4 + 2A_1^2,\\
&A_1^4A_4^3 - A_1^3A_4^4 + 2A_1^3,\\
&2A_1^4,\\
&2A_4^1A_2^4 + A_4^2A_2^3 - A_4^3A_2^2 - 2A_2^1A_4^4 + A_2^1,\\
&A_2^4A_4^2 + A_2^4A_4^3 - A_2^3A_4^4 - A_4^4A_2^2 + A_2^2,\\
&A_2^4A_4^3 - A_2^3A_4^4 + A_2^3,\\
&A_2^4,\\
&2A_4^1A_3^4 + A_4^2A_3^3 - 2A_3^1A_4^4 + A_3^1 - A_4^3A_3^2 + A_2^1,\\
&A_4^2A_3^4 + A_3^4A_4^3 - A_3^2A_4^4 + A_3^2 - A_4^4A_3^3 + A_2^2,\\
&A_3^4A_4^3 + A_2^3 - A_4^4A_3^3 + A_3^3,\\
&A_2^4 + A_3^4\}.
\end{aligned}
$$

Wir benutzen die lexikographische Ordnung auf den Termen mit der Variablenreihenfolge aus dem Abschnitt 7.1.

Bevor wir eine Gröbnerbasisberechnung starten, versuchen wir, die Polynome in einem gewissen Sinne zu vereinfachen und eventuell „überflüssige“ Variablen zu entfernen.

Durch Anwendung des Algorithmus **autoReduce** auf die Menge F erhalten wir die Menge F'. Sie beschreibt dieselbe Nullstellenmenge wie F und stellt somit eine erste Vereinfachung des Problems dar, da wir weniger Polynome zu untersuchen haben.

$$\begin{aligned}F' := \{&A_1^2, A_1^3, A_3^4, A_1^4, A_2^4,\\ &A_4^4A_1^1 - A_1^1,\\ &A_2^3A_4^4 - A_2^3,\\ &A_2^3 + A_4^4A_2^2 - A_2^2,\\ &A_2^3A_3^2 - A_3^3A_2^2 + A_1^1,\\ &-A_2^3 + A_4^4A_3^3 - A_3^3,\\ &A_4^2A_2^3 - A_4^3A_2^2 - 2A_2^1A_4^4 + A_2^1,\\ &A_2^3 + A_3^2A_4^4 - A_3^2 + A_3^3 - A_2^2,\\ &-A_4^2A_3^3 + 2A_3^1A_4^4 - A_3^1 + A_4^3A_3^2 - A_2^1\}\end{aligned}$$

Es fällt weiterhin auf, daß die Unbestimmte A_4^1 in den Elementen von F' nicht mehr vorkommt. Wir können ebenfalls die Variablen $A_1^2, A_1^3, A_3^4, A_1^4, A_2^4$ aus unserer Untersuchung streichen, da bereits jetzt schon aus F' geschlossen werden kann, daß die ihnen zugeordneten Koordinaten verschwinden. Es reicht also, die Nullstellenmenge von

$$B := F' \setminus \{A_1^2, A_3^4, A_1^4, A_1^3, A_2^4\}$$

zu beschreiben.

Entsprechend der Vorgehensweise im letzten Abschnitt versuchen wir nun, aus der Lie-Algebra-Struktur von V Polynome zu bestimmen, die wir dann im Algorithmus **gamma** für die Menge T benutzen können.

Die Derivationen $\mathrm{ad}(e_1)$, $\mathrm{ad}(e_2)$ und $\mathrm{ad}(e_3)$ sind nilpotent und geben Anlaß zu den Matrizen

$$M_1(x) := \begin{pmatrix} 1 & 0 & 0 & 2x\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix},$$

$$M_2(x) := \begin{pmatrix} 1 & 0 & x & 0\\ 0 & 1 & 0 & x\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix},$$

$$M_3(x) := \begin{pmatrix} 1 & -x & 0 & -\frac{1}{2}x^2\\ 0 & 1 & 0 & x\\ 0 & 0 & 1 & x\\ 0 & 0 & 0 & 1\end{pmatrix},$$

wobei $M_i(x)$ von $x \in K$ abhängt und die zu $\exp(x \,\mathrm{ad}\, e_i)$ gehörige Matrix ist.

Da in B die Variable A_4^1 nicht vorkommt, bringt es keinen Vorteil, die Koordinate, die dieser Unbestimmten entspricht, weiter zu untersuchen. Die

Variable x tritt jedoch in der Matrix $A \cdot M_1(x)$ nur in der Position $(1,4)$ auf. Aus diesem Grund ist die Matrix $M_1(x)$ nicht verwendbar.

Wir setzen $\Gamma_1(X_1) := M_2(X_1)$ und $\Gamma_2(X_2) := M_3(X_2)$ und definieren mit $m = 2$ die Matrizen Γ und Γ' wie im vorhergehenden Abschnitt, d. h.

$$\begin{aligned}\Gamma &:= A \cdot \Gamma_1(X_1) \cdot \Gamma_2(X_2),\\ \Gamma' &:= A \cdot \Gamma_2(-X_2) \cdot \Gamma_1(-X_1).\end{aligned}$$

Dies liefert für Γ die Einträge $(i = 1, \ldots, 4)$:

$$\begin{aligned}\gamma_1^i &= A_1^i,\\ \gamma_2^i &= -A_1^i X_2 + A_2^i,\\ \gamma_3^i &= A_1^i X_1 + A_3^i,\\ \gamma_4^i &= A_1^i X_1 X_2 - \frac{1}{2} A_1^i X_2^2 + A_2^i X_1 + A_2^i X_2 + A_3^i X_2 + A_4^i.\end{aligned}$$

Für die Einträge in Γ' erhalten wir $(i = 1, \ldots, 4)$:

$$\begin{aligned}\gamma'^i_1 &= A_1^i,\\ \gamma'^i_2 &= A_1^i X_2 + A_2^i,\\ \gamma'^i_3 &= -A_1^i X_1 + A_3^i,\\ \gamma'^i_4 &= -A_1^i X_1 X_2 - \frac{1}{2} A_1^i X_2^2 - A_2^i X_1 - A_2^i X_2 - A_3^i X_2 + A_4^i.\end{aligned}$$

Nutzen wir noch die Tatsache aus, daß gewisse Koordinaten verschwinden, nämlich die, die zu den Unbestimmten A_1^2, A_1^3, A_3^4, A_1^4 und A_2^4 gehören, so erhalten wir aus Γ die Matrix

$$\begin{pmatrix} A_1^1 & -A_1^1 X_2 + A_2^1 & A_1^1 X_1 + A_3^1 & \gamma_4^1 \\ 0 & A_2^2 & A_3^2 & A_2^2 X_1 + (A_2^2 + A_3^2) X_2 + A_4^2 \\ 0 & A_2^3 & A_3^3 & A_2^3 X_1 + (A_2^3 + A_3^3) X_2 + A_4^3 \\ 0 & 0 & 0 & A_4^4 \end{pmatrix},$$

indem wir obige Unbestimmte durch 0 ersetzen.

Da wir 2 Parameter haben, nämlich X_1 und X_2, können wir 2 der Polynome γ_j^i auswählen und diese für unsere Transformation benutzen. Wir wählen aus der ersten Zeile von Γ die Einträge der zweiten und dritten Spalte und setzen

$$T := \{(\gamma_2^1, A_2^1), (\gamma_3^1, A_3^1)\} = \{(-A_1^1 X_2 + A_2^1, A_2^1), (A_1^1 X_1 + A_3^1, A_3^1)\}. \qquad (7.5)$$

Um vergleichen zu können, was sich bei der Benutzung der Transformation ändert, beschreiben wir erst die Nullstellen von B nach der herkömmlichen Gröbnerbasistechnik, d. h., wir berechnen eine Gröbnerbasis von B.

Da wir nicht an der Gröbnerbasis, sondern an einer Beschreibung der Nullstellen interessiert sind, benutzen wir den Algorithmus **FGB**, d. h. den Buchberger-Algorithmus mit Faktorisierung. In dem REDUCE-Paket CALI ist dieser Algorithmus durch die Prozedur `groebfactor` implementiert. Sie liefert für die Eingabe B folgende Mengen:

$$\begin{aligned}
B_1 &:= \{A_1^1, A_2^3, A_2^2, A_3^3, A_2^1, A_3^2, A_3^1\},\\
B_2 &:= \{A_1^1, A_2^3, A_2^2, A_3^3, A_2^1, A_3^2, 2A_4^4 - 1\},\\
B_3 &:= \{A_2^3,\\
&\qquad (A_2^2)^2 - A_1^1,\\
&\qquad A_3^3 - A_2^2,\\
&\qquad A_4^4 - 1,\\
&\qquad A_4^3 A_2^2 + A_2^1,\\
&\qquad A_4^3 A_1^1 + A_2^1 A_2^2,\\
&\qquad A_4^2 A_2^2 - A_3^1 - A_4^3 A_3^2 + A_2^1,\\
&\qquad A_4^2 A_1^1 - A_3^1 A_2^2 + A_3^2 A_2^1 + A_2^1 A_2^2,\\
&\qquad A_4^2 A_2^1 + A_3^1 A_4^3 + (A_4^3)^2 A_3^2 - A_4^3 A_2^1\}.
\end{aligned}$$

Nach Spezifikation von `groebfactor` in [9] sind B_1, B_2 und B_3 Gröbnerbasen, und es gilt

$$\mathcal{Z}(B) = \mathcal{Z}(B_1) \cup \mathcal{Z}(B_2) \cup \mathcal{Z}(B_3).$$

Jede der drei Mengen steht für die linken Seiten eines Gleichungssystems, dessen rechte Seiten verschwinden. Zusammen mit den bereits untersuchten Variablen liefern B_1 und B_2 folgende Matrizen:

$$\begin{pmatrix} 0 & 0 & 0 & a_1 \\ 0 & 0 & 0 & a_2 \\ 0 & 0 & 0 & a_3 \\ 0 & 0 & 0 & a_4 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & a_1 & a_2 \\ 0 & 0 & 0 & a_3 \\ 0 & 0 & 0 & a_4 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} \qquad (a_1, \ldots, a_4 \in K).$$

Diese Lösungen beschreiben aber keine Automorphismen von V, da ihre Determinanten verschwinden. Aus der Menge B_3 lesen wir folgende Lösung ab:

$$\begin{pmatrix} a_1^2 & -a_1 a_5 & a_1 a_4 - a_1 a_5 - a_2 a_5 & a_3 \\ 0 & a_1 & a_2 & a_4 \\ 0 & 0 & a_1 & a_5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (a_1, \ldots, a_5 \in K,\ a_1 \neq 0).$$

Es ist klar, wie Γ und Γ' in eine Form gebracht werden können, um sie als Eingabe für den Algorithmus **gamma** zu verwenden. Wird dieser Algorithmus mit den Mengen B, T aus (7.5), Γ und Γ' als Argumenten gestartet, so erfolgt zuerst eine Gröbnerbasisberechnung der übergebenen Transformationspolynome im Ring $Q[X_1, X_2]$, wobei $R = K[A]$ der Polynomring in den noch zu betrachtenden Unbestimmten ist und Q dessen Quotientenkörper bezeichnet. Die minimale nennerfreie Gröbnerbasis, die bei dieser Berechnung entsteht, ist

$$\{A_1^1 X_2 - A_2^1, \quad A_1^1 X_1 + A_3^1\}.$$

Mit den Bezeichnungen aus dem Algorithmus **gamma** ist nun $u = 1, r_1 = A_1^1$ und $\nu_1 = 2$. Dies bedeutet, daß für alle $a \in \mathcal{Z}(B)$, deren entsprechende Koordinate a_1^1 nicht verschwindet, $x_1, x_2 \in K$ existieren, so daß

$$\gamma_2^1(x_1, x_2; a) = \gamma_3^1(x_1, x_2; a) = 0$$

gilt. Um ein solches a zu erfassen, genügt es also, nur nach den Elementen von $\mathcal{Z}(B)$ zu suchen, für die die Koordinaten mit den Indizes (1,2) und (1,3) verschwinden, und danach die Transformation, die durch Γ' gegeben ist, anzuwenden. Auf diese Weise wird die Menge S_1 bestimmt.

Von $\mathcal{Z}(B)$ sind nun nur noch die Elemente a zu beschreiben, für die $a_1^1 = 0$ ist. Diese Bedingung fügen wir in Form des Polynoms A_1^1 zur Menge B hinzu und bestimmen davon eine Gröbnerbasis. Wir haben damit auch in diesem Fall eine Problemreduzierung erreicht, da praktisch eine Gröbnerbasisberechnung mit einer kleineren Variablenanzahl erfolgt. Auf diese Weise wird die Menge S_0 berechnet.

Da Lie-Algebra-Automorphismen zu beschreiben sind und zuvor bereits erkannt wurde, daß außer a_1^1 alle anderen Koordinaten der ersten Spalte in der Matrixdarstellung verschwinden, könnte die weitere Berechnung bereits hier abgebrochen werden, da mit der Bedingung $a_1^1 = 0$ kein Automorphismus entstehen kann. Eine Erweiterung des Algorithmus **gamma**, die diese Tatsache berücksichtigt, ist prinzipiell möglich, hätte aber keinen Vorteil, wenn nach allen Endomorphismen gesucht wird.

Das Ergebnis, das von **gamma** geliefert wird, sind die folgenden Mengen:

$$\begin{aligned}
S_0 = \{&(\{A_1^1, A_2^2, A_2^3, A_2^1, A_3^3, A_3^2, 2A_4^4 - 1\}, \{A_3^1, A_4^4 - 1, A_4^4\}),\\
&(\{A_3^3, A_1^1, A_2^2, A_2^3, A_2^1, A_3^1 + A_4^3 A_3^2, A_4^4 - 1\}, \{A_4^4\}),\\
&(\{A_1^1, A_2^2, A_2^3, A_2^1, A_3^3, A_3^2 A_3^1\}, \emptyset)\},\\
S_1 = \{&(\{A_2^1, A_3^1, A_2^3, A_4^3, A_4^2, (A_2^2)^2 - A_1^1, A_3^3 - A_2^2, A_4^4 - 1\}, \{A_1^1\})\}.
\end{aligned}$$

Die Elemente von S_0 geben nur Anlaß zu nicht-invertierbaren Matrizen und beschreiben somit keine Automorphismen.

S_1 hingegen führt auf die Matrix

$$\begin{pmatrix} a_1^2 & 0 & 0 & a_3 \\ 0 & a_1 & a_2 & 0 \\ 0 & 0 & a_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (a_1, a_2, a_3 \in K,\ a_1 \neq 0).$$

Benutzen wir nun Γ', um zur vollständigen Lösung zu kommen, so erhalten wir die Matrix

$$\begin{pmatrix} a_1^2 & a_1^2 x_2 & -a_1^2 x_1 & -a_1^2 x_1 x_2 - \frac{1}{2} a_1^2 x_2^2 + a_3 \\ 0 & a_1 & a_2 & -a_1 x_1 - a_1 x_2 - a_2 x_2 \\ 0 & 0 & a_1 & -a_1 x_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

in Abhängigkeit von $a_1, a_2, a_3, x_1, x_2 \in K$ mit $a_1 \neq 0$.

Obwohl wir hier eine andere Darstellung der Lösungsmenge erhalten haben als bei der ersten Methode, lassen sich beide Darstellungen durch entsprechende Transformation der Parameter ineinander überführen.

Beim Vergleich beider Darstellungen fällt auf, daß bei der ersten Methode vier Parameter, nämlich a_1, a_2, a_4, a_5, aus der Menge B_3 bestimmt werden mußten. Bei der zweiten Methode ergeben sich zwei Parameter, nämlich x_1 und x_2, in natürlicher Weise aus der Transformation, die wir auf die Nullstellenmenge anwenden konnten. Den Parameter a_3 erhalten wir aus der Tatsache, daß die Unbestimmte A_4^1 nicht in der Menge F' vorkommt.

Dieser Unterschied bei der Bestimmung der Parameter ist bei Benutzung der lexikographischen Termordnung nicht wesentlich. Läßt sich aber die Gröbnerbasis bzgl. dieser Termordnung auf Grund von Zeit- oder Speicherplatzproblemen nicht berechnen und muß deshalb eine andere gewählt werden, kann dies ein Vorteil sein.

7.4 Die Lie-Algebra $A_{5,2}$

Mit K bezeichnen wir hier wieder den Körper der komplexen Zahlen. Wir betrachten die 5-dimensionale komplexe Lie-Algebra V, die gegeben ist durch die Relationen

$$\begin{aligned} [e_2, e_5] &= e_1 \\ [e_3, e_5] &= e_2 \\ [e_4, e_5] &= e_3 \end{aligned}$$

zwischen den Basisvektoren. In der Notation von Patera in [15] ist dies die Komplexifizierung der reellen Lie-Algebra $A_{5,2}$.

Die Menge F, die wir gemäß (7.4) erhalten, besteht aus 36 Polynomen. F läßt sich leicht mit Hilfe eines Computeralgebrasystems aus den obigen Kommutatorrelationen berechnen. Wir werden im Anhang A beschreiben, wie dies erreicht werden kann. Wir verwenden auch hier die lexikographische Termordnung und erhalten aus F nach Anwendung von **autoReduce** die Menge

$$\begin{aligned}F' := \{&A_1^2, A_2^3, A_1^3, A_1^4, A_1^5, A_2^4, A_2^5, A_3^4, A_3^5,\\ &A_4^5 A_3^3,\\ &A_4^5 A_3^2,\\ &A_4^5 A_2^2,\\ &A_5^5 A_3^3 - A_2^2,\\ &A_3^2 A_5^5 - A_2^1,\\ &A_5^5 A_2^2 - A_1^1,\\ &A_5^2 A_4^5 - A_4^2 A_5^5 + A_3^1,\\ &A_5^3 A_4^5 - A_4^3 A_5^5 + A_3^2,\\ &A_4^5 A_5^4 - A_5^5 A_4^4 + A_3^3\}.\end{aligned}$$

Die Unbestimmten A_4^1 und A_5^1 treten in den Polynomen von F' nicht mehr auf. Wir können ebenfalls die Variablen $A_1^2, A_2^3, A_1^3, A_1^4, A_1^5, A_2^4, A_2^5, A_3^4, A_3^5$ aus unserer Untersuchung streichen, da bereits jetzt schon aus F' geschlossen werden kann, daß die ihnen zugeordneten Koordinaten verschwinden. Es reicht also, die Nullstellenmenge von

$$B := F' \setminus \{A_1^2, A_2^3, A_1^3, A_1^4, A_1^5, A_2^4, A_2^5, A_3^4, A_3^5\}$$

zu beschreiben.

Genauso wie beim ersten Beispiel erhalten wir nun Matrizen

$$M_2(x) := \begin{pmatrix} 1 & 0 & 0 & 0 & x \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_3(x) := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & x \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_4(x) := \begin{pmatrix} 1 & 0 & x & 0 & 0 \\ 0 & 1 & 0 & 0 & x \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_5(x) := \begin{pmatrix} 1 & -x & \frac{1}{2}x^2 & -\frac{1}{5}x^3 & 0 \\ 0 & 1 & -x & \frac{1}{2}x^2 & 0 \\ 0 & 0 & 1 & -x & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

die zu $\exp(x \operatorname{ad} e_i)$ $(i = 2, 3, 4, 5)$ gehören.

Zu $\exp(x \operatorname{ad} e_1)$ gehört die Einheitsmatrix, die nicht von x abhängt und somit auch nicht verwendet werden kann. Auf Grund der Dreiecksgestalt aller Matrizen und der Tatsache, daß A_5^1 in F' nicht vorkommt, ist auch $M_2(x)$ nicht verwendbar.

Wir setzen $\Gamma_1(X_1) := M_3(X_1)$, $\Gamma_2(X_2) := M_4(X_2)$ und $\Gamma_3(X_3) := M_5(X_3)$. Die Matrizen Γ und Γ' seien wie folgt definiert:

$$\begin{aligned}
\Gamma &:= A \cdot \Gamma_1(X_1) \cdot \Gamma_2(X_2) \cdot \Gamma_3(X_3), \\
\Gamma' &:= A \cdot \Gamma_1(-X_2) \cdot \Gamma_2(-X_1) \cdot \Gamma_3(-X_3).
\end{aligned}$$

Die Einträge von Γ sind dann die folgenden $(i = 1, \ldots, 4)$:

$$\begin{aligned}
\gamma_1^i &= A_1^i, \\
\gamma_2^i &= -A_1^i X_3 + A_2^i, \\
\gamma_3^i &= \frac{1}{2} A_1^i X_3^2 - A_2^i X_3 + A_3^i, \\
\gamma_4^i &= -\frac{1}{6} A_1^i X_3^3 + \frac{1}{2} A_2^i X_3^2 - A_3^i X_3 + A_4^i), \\
\gamma_5^i &= A_2^i X_1 + A_3^i X_2 + A_5^i.
\end{aligned}$$

Für die Einträge in Γ' erhalten wir $(i = 1, \ldots, 4)$:

$$\begin{aligned}
{\gamma'}_1^i &= A_1^i, \\
{\gamma'}_2^i &= A_1^i X_3 + A_2^i, \\
{\gamma'}_3^i &= \frac{1}{2} A_1^i X_3^2 + A_2^i X_3 + A_3^i, \\
{\gamma'}_4^i &= \frac{1}{6} A_1^i X_3^3 + \frac{1}{2} A_2^i X_3^2 + A_3^i X_3 + A_4^i, \\
{\gamma'}_5^i &= -A_1^i X_1 X_3 - \frac{1}{2} A_1^i X_2 X_3^2 - A_2^i X_1 - A_2^i X_2 X_3 - A_3^i X_2 + A_5^i.
\end{aligned}$$

Nutzen wir noch die Tatsache aus, daß gewisse Koordinaten verschwinden, nämlich die, die zu den Unbestimmten A_1^2, A_1^3, A_2^3, A_1^4, A_2^4, A_3^4, A_1^5, A_2^5 und A_3^5 gehören, so erhalten wir aus Γ die Matrix

$$\begin{pmatrix}
A_1^1 & \gamma_2^1 & \gamma_3^1 & \gamma_4^1 & \gamma_5^1 \\
0 & A_2^2 & -A_2^2 X_3 + A_3^2 & \frac{1}{2} A_2^2 X_3^2 - A_3^2 X_3 + A_4^2 & A_2^2 X_1 + A_3^1 X_2 + A_5^1 \\
0 & 0 & A_3^3 & -A_3^3 X_3 + A_4^3 & A_3^3 X_2 + A_5^3 \\
0 & 0 & 0 & A_4^4 & A_5^4 \\
0 & 0 & 0 & A_4^5 & A_5^5
\end{pmatrix},$$

indem wir obige Unbestimmte durch 0 ersetzen.

Da wir 3 Parameter haben, nämlich X_1, X_2 und X_3, können wir aus obiger Matrix 3 Einträge auswählen und diese für unsere Transformation nutzen. Wir wählen die Einträge an den Positionen (2,5), (3,4) und (3,5) und setzen

$$T := \{(A_2^2 X_1 + A_3^1 X_2 + A_5^1, A_5^2), (-A_3^3 X_3 + A_4^3, A_4^3), (A_3^3 X_2 + A_5^3, A_5^3)\}. \quad (7.6)$$

Statt des Elements an der Stelle (3,4) hätten wir auch jenes an der Position (1,2) wählen können. Eine Berechnung mit diesen Werten hat aber auf das Resultat keinen Einfluß, d. h. liefert dieselben Mengen S_0 und S_1. Sogar die Berechnungsdauer weicht in beiden Fällen nur unwesentlich voneinander ab.

Zum Vergleich benutzen wir, wie schon im letzten Beispiel, die Prozedur `groebfactor` von CALI, um eine Beschreibung der Nullstellen nach der herkömmlichen Methode zu erhalten. Diese Prozedur liefert für B zwei Mengen B_1 und B_2, wobei die zweite keine Automorphismen beschreibt, da die zugehörige Determinate verschwindet. Die erste Menge hat folgende Gestalt:

$$\begin{aligned} B_1 = \{&A_4^5, \quad A_5^5 A_4^4 - A_3^3, \quad A_4^3 A_5^5 - A_3^2, \quad A_4^2 A_5^5 - A_3^1, \quad A_5^5 A_3^3 - A_2^2, \\ &A_3^2 A_5^5 - A_2^1, \quad A_5^5 A_2^2 - A_1^1, \quad A_3^3 A_1^1 - (A_2^2)^2, \quad A_4^4 A_1^1 - A_3^3 A_2^2, \\ &A_4^4 A_2^2 - (A_3^3)^2, \quad A_3^2 A_1^1 - A_2^1 A_2^2, \quad A_3^2 A_2^2 - A_2^1 A_3^3, \quad A_3^2 A_3^3 - A_2^1 A_4^4, \\ &A_4^3 A_1^1 - A_2^1 A_3^3, \quad A_4^3 A_2^2 - A_2^1 A_4^4, \quad A_4^3 A_3^3 - A_3^2 A_4^4, \quad A_4^3 A_2^1 - (A_3^2)^2, \\ &A_4^2 A_1^1 - A_3^1 A_2^2, \quad A_4^2 A_2^2 - A_3^1 A_3^3, \quad A_4^2 A_3^3 - A_3^1 A_4^4, \quad A_4^2 A_2^1 - A_3^1 A_3^2, \\ &A_4^2 A_3^2 - A_3^1 A_4^3\}. \end{aligned}$$

Sie führt auf die Lösung

$$\begin{pmatrix} a_1^3 a_2 & a_1^2 a_2 & a_1 a_4 & a_5 & a_6 \\ 0 & a_1 a_2 & a_1 a_3 & a_4 & a_7 \\ 0 & 0 & a_1 a_2 & a_3 & a_8 \\ 0 & 0 & 0 & a_2 & a_9 \\ 0 & 0 & 0 & 0 & a_1 \end{pmatrix} \qquad (a_1, \ldots, a_9 \in K, a_1, a_2 \neq 0). \quad (7.7)$$

Analog zum ersten Beispiel führt die Anwendung des Algorithmus **gamma** auf die Mengen B, T, Γ und Γ' zuerst auf eine Bestimmung einer nennerfreien Gröbnerbasis der übergebenen Transformationspolynome. Hier erfolgt die Rechnung im Ring $Q[X_1, X_2, X_3]$, wobei $R = K[A]$ der Polynomring in den noch zu betrachtenden Unbestimmten ist und Q dessen Quotientenkörper bezeichnet. Diese Gröbnerbasis ist die folgende Menge:

$$\{A_2^2 A_3^3\, X_1 - A_3^2 A_5^3 - A_5^2 A_3^3, \quad A_3^3\, X_3 - A_4^3, \quad A_3^3\, X_2 + A_5^3\}.$$

Mit den Bezeichnungen aus dem Algorithmus **gamma** haben wir nun $u = 2$, $r_1 = A_2^2$, $r_2 = A_3^3$, $\nu_1 = 1$ und $\nu_2 = 3$. Dies bedeutet, daß für alle $a \in \mathcal{Z}(B)$, deren entsprechende Koordinaten a_2^2 und a_3^3 nicht verschwinden, Zahlen

$x_1, x_2, x_3 \in K$ existieren, so daß die Auswertung der verwendeten Transformationspolynome aus (7.6) an der Stelle $(x_1, x_2, x_3; a)$ jeweils 0 liefert.

Um solche a zu beschreiben, reicht es offensichtlich, nur nach den Elementen von $\mathcal{Z}(B)$ zu suchen, für die die Koordinaten mit den Indizes (2,5), (3,4) und (3,5) verschwinden, und danach die Transformation Γ' anzuwenden. Auf diese Weise wird die Menge S_1 bestimmt.

Für Elemente a aus dem Teil von $\mathcal{Z}(B)$, der nun noch nicht erfaßt wurde, gilt $a_2^2 = 0$ oder $a_3^3 = 0$. Dies führt somit auf eine Aufteilung in zwei Teilprobleme, bei denen praktisch jeweils eine Variable weniger auftritt.

Insgesamt liefert aber nur die Menge S_1 eine Lösung für die Automorphismen, da S_0 nur zu Lösungen mit verschwindender Determinante Anlaß gibt.

$$
\begin{aligned}
S_1 = \{(& A_2^1, \quad A_3^2, \quad A_5^2, \quad A_4^3, \quad A_5^3, \quad A_4^5, \\
& A_5^5 A_4^4 - A_3^3, \quad A_4^2 A_5^5 - A_3^1, \quad A_5^5 A_3^3 - A_2^2, \quad A_5^5 A_2^2 - A_1^1, \\
& A_3^3 A_1^1 - (A_2^2)^2, \quad A_4^4 A_1^1 - A_3^3 A_2^2, \quad A_4^4 A_2^2 - (A_3^3)^2, \quad A_4^2 A_1^1 - A_3^1 A_2^2, \\
& A_4^2 A_2^2 - A_3^1 A_3^3, \quad A_4^2 A_3^3 - A_3^1 A_4^4\}, \\
& \{A_2^2, A_3^3\})\}
\end{aligned}
$$

Aus S_1 können wir dann eine zu Automorphismen unserer Lie-Algebra gehörende Matrix bestimmen. Sie hat die Form

$$
\begin{pmatrix}
a_1^3 a_2 & 0 & a_1 a_3 & a_4 & a_5 \\
0 & a_1^2 a_2 & 0 & a_3 & 0 \\
0 & 0 & a_1 a_2 & 0 & 0 \\
0 & 0 & 0 & a_2 & a_6 \\
0 & 0 & 0 & 0 & a_1
\end{pmatrix}
\qquad (a_1, \ldots, a_6 \in K, a_1, a_2 \neq 0). \tag{7.8}
$$

Um zur vollen Beschreibung der Automorphismen zu kommen, müssen wir noch Γ' benutzen. Wir erhalten dann

$$
\begin{pmatrix}
a_1^3 a_2 & -a_1^3 a_2 x_3 & \frac{1}{2} a_1^3 a_2 x_3^2 + a_1 a_3 & a_4' & a_5' \\
0 & a_1^2 a_2 & -a_1^2 a_2 x_3 & \frac{1}{2} a_1^2 a_2 x_3^2 + a_3 & a_1^2 a_2 (x_1 - x_2 x_3) \\
0 & 0 & a_1 a_2 & -a_1 a_2 x_3 & a_1 a_2 x_2 \\
0 & 0 & 0 & a_2 & a_6 \\
0 & 0 & 0 & 0 & a_1
\end{pmatrix}
$$

in Abhängigkeit von $a_1, \ldots, a_6, x_1, x_2, x_3 \in K$ mit $a_1, a_2 \neq 0$. Hierbei ist

$$
\begin{aligned}
a_4' &= -\frac{1}{6} a_1^3 a_2 x_3^3 + a_1 a_3 x_3 + a_4, \\
a_5' &= -a_1^3 a_2 x_1 x_3 - \frac{1}{2} a_1^3 a_2 x_2 x_3^2 + a_1 a_3 x_2 + a_5.
\end{aligned}
$$

Da a_4 und a_5 beliebig gewählt werden können, können wir auch die gestrichenen Variablen als freie Parameter betrachten.

Durch entsprechende Transformation der Parameter lassen sich die Lösungen, die wir durch zwei verschiedene Lösungswege erhalten haben, ineinander überführen.

Wie schon im letzten Beispiel fällt auch hier auf, daß nach der zweiten Methode, d. h. bei Anwendung gewisser Symmetrien, nur 6 Parameter aus der partiellen Lösung S_1 bestimmt werden müssen und sich 3 weitere auf eine natürliche Weise aus den Transformationen ergeben.

Diese Tatsache ist erst dann von größerer Bedeutung, wenn sich keine lexikographische Termordnung verwenden läßt. Bei einer anderen Termordnung lassen sich im allgemeinen die Parameter nicht ohne weiteres aus der Gröbnerbasis bestimmen. In jenem Fall ist es günstig, bereits einige Parameter zu kennen, um mit geringerem Aufwand die restlichen Parameter bestimmen zu können.

7.5 Effizienzuntersuchungen

In diesem Abschnitt werden wir weitere Beispiele untersuchen und uns dabei aber hauptsächlich den Abarbeitungszeiten für verschiedene Eingaben widmen. Wir untersuchen weiterhin die Klasse niedrigdimensionaler Lie-Algebren und übernehmen die Notation von Patera in [15]. Allerdings betrachten wir die Komplexifizierung dieser Lie-Algebren, um alle Berechnungen über einem algebraisch abgeschlossenen Körper ausführen zu können.

Die Lie-Algebren $A_{4,7}$ und $A_{5,2}$ sind bereits aus den letzten beiden Beispielen bekannt. Für alle weiteren geben wir nicht-verschwindende Kommutatorrelationen an, die diese Lie-Algebren beschreiben.

$A_{4,8}$:	$[e_2, e_3] = e_1$	$[e_2, e_4] = e_2$	$[e_3, e_4] = -e_3$	
$A_{4,10}$:	$[e_2, e_3] = e_1,$	$[e_2, e_4] = -e_3,$	$[e_3, e_4] = e_2$	
$A_{4,12}$:	$[e_1, e_3] = e_1,$	$[e_2, e_3] = e_2,$	$[e_1, e_4] = -e_2,$	$[e_2, e_4] = e_1$
$A_{5,3}$:	$[e_3, e_4] = e_2,$	$[e_3, e_5] = e_1,$	$[e_4, e_5] = e_3$	
$A_{5,5}$:	$[e_3, e_4] = e_1,$	$[e_2, e_5] = e_1,$	$[e_3, e_5] = e_2$	
$A_{5,6}$:	$[e_3, e_4] = e_1,$	$[e_2, e_5] = e_1,$	$[e_3, e_5] = e_2,$	$[e_4, e_5] = e_3$
$A_{5,22}$:	$[e_2, e_3] = e_1,$	$[e_2, e_5] = e_3,$	$[e_4, e_5] = e_4$	
$A_{5,37}$:	$[e_2, e_3] = e_1,$	$[e_1, e_4] = 2e_1,$	$[e_2, e_4] = e_2,$	$[e_3, e_4] = e_3,$
	$[e_2, e_5] = -e_3,$	$[e_3, e_5] = e_2$		
$A_{5,40}$:	$[e_1, e_2] = 2e_1,$	$[e_1, e_3] = -e_2,$	$[e_2, e_3] = 2e_3$	$[e_1, e_4] = e_5,$
	$[e_2, e_4] = e_4,$	$[e_2, e_5] = -e_5$	$[e_3, e_5] = e_4$	
$A_{6,1}$:	$[e_1, e_2] = e_3,$	$[e_1, e_3] = e_4,$	$[e_1, e_5] = e_6$	
$A_{6,2}$:	$[e_1, e_2] = e_3,$	$[e_1, e_3] = e_4,$	$[e_1, e_4] = e_5$	$[e_1, e_5] = e_6$
$A_{6,22}$:	$[e_1, e_2] = e_3,$	$[e_1, e_3] = e_5,$	$[e_1, e_5] = e_6$	$[e_2, e_3] = e_4,$

$$[e_2, e_4] = e_5, \qquad [e_3, e_6] = e_6$$

Um den Algorithmus wie in den vorangehenden Beispielen anwenden zu können, müssen wir noch festlegen, welche Transformationspolynome zu verwenden sind. Die Matrizen Γ und Γ' werden für jede Lie-Algebra in derselben Weise konstruiert wie früher. Es reicht also, die Zeilen- und Spaltenindizes der Einträge von Γ anzugeben, die die Transformationspolynome bilden.

Wir verwenden:

$A_{4,7}$:	(1,2),	(1,3)		
$A'_{4,7}$:	(2,4),	(3,4)		
$A_{4,8}$:	(1,2),	(1,3)		
$A_{4,10}$:	(1,2),	(1,3)		
$A_{4,12}$:	(1,3),	(1,4)		
$A_{5,2}$:	(2,5),	(3,4),	(3,5)	
$A'_{5,2}$:	(1,2),	(2,5),	(3,5)	
$A_{5,3}$:	(3,4),	(3,5)		
$A_{5,5}$:	(2,3),	(2,5)		
$A'_{5,5}$:	(1,2),	(2,5)		
$A_{5,6}$:	(1,2),	(2,5)		
$A_{5,22}$:	(1,2),	(2,5)		
$A_{5,37}$:	(1,2),	(1,3),	(1,4)	
$A_{5,40}$:	(5,4),	(4,2),	(4,3),	(4,5)
$A_{6,1}$:	(3,1),	(3,2)		
$A_{6,2}$:	(3,1),	(3,2),	(4,1),	(5,1)
$A_{6,22}$:	(3,1),	(3,2),	(4,2),	(5,2)

Manche der Lie-Algebren sind mit einem Strich versehen, um auszudrücken, daß zwar die entsprechenden Lie-Algebren aber unterschiedliche Transformationspolynome benutzt werden.

REDUCE bietet die Möglichkeit, die Ausführungszeit einer Routine in Millisekunden anzuzeigen. Die Zeiten für die Berechnung der Mengen S_0 und S_1 durch den in REDUCE implementierten Algorithmus **gamma** und die Zeit für die Lösung desselben Problems mit Hilfe der Prozedur `groebfactor` von CALI sind in der folgenden Tabelle zusammengefaßt.

$A_{4,7}$	40	360	330	730	450
$A'_{4,7}$	70	340	150	560	440
$A_{4,8}$	30	380	210	620	260
$A_{4,10}$	40	310	630	980	1080
$A_{4,12}$	70	6060	4810	10940	11460
$A_{5,2}$	50	600	250	900	2620
$A'_{5,2}$	50	630	320	1000	2650
$A_{5,3}$	40	90	530	660	4100
$A_{5,5}$	50	1170	460	1680	2110
$A'_{5,5}$	60	1590	340	1990	2100
$A_{5,6}$	60	1600	330	1990	2110
$A_{5,22}$	60	1570	330	1960	2100
$A_{5,37}$	160	166570	2590	169320	175730
$A_{5,40}$	860	199610	7150	207620	224850
$A_{6,1}$	80	22290	700	23070	21830
$A_{6,2}$	140	4340	390	4870	2730
$A_{6,22}$	180	10600	850	11630	12760

Dabei beziehen sich die 2. bis 5. Spalte auf den Algorithmus **gamma**, und die 6. Spalte enthält die Ausführungszeit von `groebfactor`.

Die 2. Spalte beinhaltet die Zeit für die Entscheidung, ob die ausgewählten Transformationspolynome zur Problemreduzierung benutzt werden können oder nicht. Diese Zeit ist verhältnismäßig klein gegenüber der Gesamtberechnungsdauer, könnte aber Bedeutung erlangen, wenn die Anzahl der Parameter größer wird als in den betrachteten Beispielen.

Die 3. Spalte gibt die Berechnungszeit für die Menge S_0 an, die 4. Spalte jene für S_1.

In der 5. Spalte ist die Gesamtausführungszeit von **gamma** eingetragen, also die Summe der Spalten 2, 3 und 4.

Vergleichen wir die letzten beiden Spalten, so erkennen wir, daß in vielen Fällen der Algorithmus **gamma** schneller ist. Doch selbst wenn die Berechnungszeit nicht wesentlich besser ist, liefert **gamma** einen gewissen Vorteil. Wird nämlich nach einer Beschreibung des Nullstellengebildes durch Parameter gesucht, so liefert die Anwendbarkeit der gegebenen Transformation bereits einen Teil der verwendbaren Parameter in natürlicher Weise.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Kurzzusammenfassung

Die vorliegende Arbeit untersucht eine Verallgemeinerung des Buchberger-Algorithmus und führt zu einem Algorithmus, der zusätzlich kontinuierliche Symmetrien berücksichtigt, die bei manchen Problemen in natürlicher Weise vorhanden sind. Zu Beginn wird in die Theorie der Gröbnerbasen eingeführt. Dies geschieht allerdings nur in dem Umfang, wie es für die weitere Abhandlung notwendig ist. Außerdem werden die Definitionen gleich in einer für die Arbeit geeigneteren Form angegeben. Besonderen Wert wird auf die nennerfreie Präsentation der Algorithmen gelegt, die zur Konstruktion einer Gröbnerbasis benötigt werden. Schließlich wird der neue Algorithmus **gamma** angegeben und seine Korrektheit gezeigt. An Beispielen wird dann die Wirkungsweise dieses Algorithmus demonstriert.

8.2 Ausblick auf weitere Untersuchungen

Ohne dies stets anzumerken, verwenden wir in diesem Abschnitt Bezeichnungen in derselben Bedeutung wie in den vorhergehenden Kapiteln.

Der Algorithmus **gamma** läßt sich so erweitern, daß zusätzlich eine Menge von Polynomen berücksichtigt wird, die als „Einschränkungen“ der Nullstellenmenge im vorn beschriebenen Sinne dienen. Eine Untersuchung der gewählten Beispiele mit einem solchen Algorithmus ergab zwar eine Verringerung der Gesamtrechenzeit, jedoch war kein Rechenvorteil gegenüber `groebfactor` sichtbar, wenn auch diese Prozedur mit jenem zusätzlichen Parameter aufgerufen wurde. Daher und auf Grund der Tatsache, daß sich diese Arbeit mit der prinzipiellen Anwendbarkeit kontinuierlicher Symmetrien bei der Lösung von Gleichungssystemen beschäftigt, wurde auf die Angabe dieses erweiterten Algorithmus verzichtet und **gamma** in einer einfachen Form präsentiert, aus der die Anwendung der Transformation deutlicher zu entnehmen ist.

In manchen Fällen treten Transformationen auf, die nicht in polynomialer Weise von Parametern abhängen. Für einen Teil dieser Transformationen läßt sich trotzdem eine Problemaufteilung erreichen. Die Idee ist, „Parameter“ zu untersuchen, die selbst polynomialen Bedingungen unterworfen sind. Unter diesem Blickwinkel wird versucht, die Anzahl der anwendbaren Parameter zu erhöhen. Läßt sich nämlich z. B. die Transformation durch Polynome in $\sin x$ und $\cos x$ beschreiben, so ist dies in Abhängigkeit vom Parameter x keine polynomiale Situation. Der Parameter x könnte nicht benutzt werden. Nach Ersetzen von $\sin x$ durch s, $\cos x$ durch c, wobei s und c als Parameter betrachtet werden, sowie Hinzunehmen der Bedingung $c^2+s^2=1$ ist wieder alles polynomial. Das Polynom c^2+s^2-1 muß aber bei der Berechnung der Gröbnerbasis der Transformationspolynome $\{\delta_1,\ldots,\delta_l\}$ hinzugenommen werden, da s und c keine freien Parameter sind.

Eine kombinatorische Schwierigkeit, die in der vorgelegten Arbeit noch keine Rolle spielt, ergibt sich durch die Auswahl der Koordinaten, die zum Verschwinden gebracht werden sollen. Bei den von uns betrachteten Beispielen ließen sich diese Koordinaten ohne viel Mühe bestimmen. Für größere Probleme muß aber ein effizienter Algorithmus angegeben werden, der diese Koordinaten findet, oder es muß eine solche Herangehensweise gänzlich vermieden werden. Jener Algorithmus könnte z. B. eine eventuelle Faktorisierbarkeit von Polynomen von B ausnutzen, um das Problem zu zerlegen und relevante Koordinaten in den Teilproblemen zu suchen.

Die Betrachtung der gerechneten Beispiele zeigt auch einen weiteren Weg bei der Untersuchung der Anwendbarkeit kontinuierlicher Symmetrien. Es stellt sich nämlich heraus, daß der größte Teil der Zeit, die **gamma** benötigt, zur Berechnung der Menge S_0 verwendet wird. Es erscheint daher sinnvoll, nach einer Möglichkeit zu suchen, die gegebene Transformation auch auf den Teil von $\mathcal{Z}(B)$ anzuwenden, die in unserer Betrachtung als „Ausnahmemenge“ auftritt.

Für diese Untersuchung erscheint eine geometrische Sichtweise von Vorteil. In der Arbeit wird folgende Vorgehensweise benutzt. Es werden gewisse Koordinaten-Hyperebenen gewählt und die durch die Transformation in Bahnen geteilte Nullstellenmenge $\mathcal{Z}(B)$ mit dem Durchschnitt dieser Hyperebenen zum Schnitt gebracht. Die oben erwähnte Ausnahmemenge wird gerade von den Bahnen gebildet, die keinen Punkt mit diesem Durchschnitt gemeinsam haben.

Eine erste Idee, die zu „kleineren“ Ausnahmemengen führen kann, ist, die

Forderung fallenzulassen, daß die Hyperebenen Koordinaten-Hyperebenen sein müssen. Dann entsteht aber die Frage, wie die Gleichungen dieser Hyperebenen zu wählen sind, um alle Bahnen zu erfassen.

Da nicht a priori entschieden werden kann, ob die Bahnen eine solche Lage im Raum haben, daß es überhaupt eine Hyperebene gibt, die alle Bahnen schneidet, können als weitere Verallgemeinerung Hyperflächen zugelassen werden. Dabei muß gefordert werden, daß die Schnittmenge Z dieser Hyperflächen mit der gesuchten Nullstellenmenge $\mathcal{Z}(B)$ wieder eine algebraische Menge ist, d. h. sich durch Polynome beschreiben läßt, da auf diese Menge die Gröbnerbasistechnik angewendet werden soll. Mit anderen Worten, es gibt eine Menge $D = \{d_1, \ldots, d_l\}$ von Polynomen, so daß $Z = \mathcal{Z}(B \cup D)$ ist.[1] Auch hier ergibt sich die Frage, wie die Gleichungen dieser Hyperflächen und die Menge D bestimmt werden könnten.

Ist aber, z. B. aus der Konstruktion der Hyperflächen, gesichert, daß alle Bahnen erfaßt werden, so kann auf eine Gröbnerbasisberechnung der ausgewählten Transformationspolynome $\{\delta_1, \ldots, \delta_l\}$ verzichtet werden. Dann muß nur die entsprechende Menge D, die aber ebenfalls erst zusammen mit den Hyperflächen konstruiert werden muß, zu den Polynomen von B hinzugenommen werden und hiervon eine Gröbnerbasis bestimmt werden.

Schließlich kann auch diese Idee mit der Verwendung von Parametern, die gewissen polynomialen Bedingungen unterworfen sind, kombiniert werden.

[1] Die Bezeichnung lehnt sich an jene im Algorithmus **gamma** an.

Anhang A

Verwendung von REDUCE

REDUCE ist ein weitverbreitetes Computeralgebrasystem und heute ein gängiges Hilfsmittel bei der Lösung vieler Probleme. Eine Beschreibung von REDUCE ist in [10] zu finden.

Wir beschränken uns hier darauf, die wesentlichen Routinen anzugeben, die für die Berechnung der in der Arbeit beschriebenen Beispiele verwendet wurden. Funktionen, die hier nicht erwähnt werden, sind Bestandteil von REDUCE oder CALI und sind in den zugehörigen Dokumentationen spezifiziert oder im verfügbaren Quelltext von CALI zu finden.

Die Programmierung in REDUCE erfolgt im algebraischen oder symbolischen Modus. Für die Bereitstellung neuer Funktionen ist es üblich und effizienter, den symbolischen Modus zu wählen, da dadurch die ständige Konvertierung der Datenstrukturen in ein Standardformat, die algebraische Präfixform, entfällt und außerdem eine größere Freiheit besteht, Daten zu manipulieren, als im algebraischen Modus. Aus diesem Grunde sind auch REDUCE-Pakete, wie etwa CALI, im symbolischen Modus programmiert und stellen Funktionen für den algebraischen Modus zur Verfügung.

Der algebraische Modus ist der „natürliche" REDUCE-Modus. Dieser ist näher an der mathematischen Sprache orientiert. Für eine Beschreibung der Unterschiede beider Modi verweisen wir auf das Benutzerhandbuch [10] von REDUCE.

Wir geben zuerst einige Prozeduren an, die zur Bereitstellung der Daten für die weiter unten präsentierte Implementation des Algorithmus **gamma** nützlich sind.

```
algebraic procedure unknowns(dim,a);
  begin scalar j,k;
  return for j:=1:dim collect for k:=1:dim collect mkid(a,10*j+k);
  end;
```

Diese Funktion erzeugt eine Liste der Unbestimmten A_k^j in einer Form, wie sie später bei der Generierung der Polynome, die Endomorphismen einer Lie-Algebra beschreiben, Verwendung finden.

Die Bereitstellung der Strukturkonstanten für die betrachteten Lie-Algebren kann in verschiedenster Form erfolgen. Wir nehmen an, daß die Strukturkonstanten c_{jk}^l als eine Liste `sc` von Matrizen vorliegen, so daß die j-te Matrix wie folgt aussieht:

$$\begin{pmatrix} c_{j1}^1 & \dots & c_{jN}^1 \\ \vdots & & \vdots \\ c_{j1}^N & \dots & c_{jN}^N, \end{pmatrix}$$

wobei N die Dimension der betrachteten Lie-Algebra ist. Dies hat den Vorteil, daß diese Matrix gerade die zu $\mathrm{ad}(e_j)$ gehörende Matrix ist; vgl. mit der Identität (7.1). Aus der Liste `sc` wird durch folgende Funktion das Element c_{jk}^l ausgewählt.

```
algebraic procedure structureConstant(sc,j,k,l);
  << m := part(sc,j); m(l,k) >>;
```

Die folgende Funktion liefert die im Kapitel 7 beschriebene Menge F, wenn für `llsy`[1] die Unbestimmten eingesetzt werden, wie sie durch die Funktion `unknowns` bereitgestellt werden.

```
algebraic procedure endomorphisms(sc,llsy);
  begin scalar m,m1,m2,j,k,l,poly,dim;
  dim := length sc;
  return (for k:=2:dim join for j:=1:(k-1) join for l:=1:dim join
    <<
    poly := (for m1:=1:dim sum
      part(llsy,l,m1)*structureConstant(sc,j,k,m1)) -
      %--     a_m1^l * c_{jk}^m1
      (for m1:=1:dim sum for m2:=1:dim sum
        part(llsy,m1,j)*part(llsy,m2,k)*structureConstant(sc,m1,m2,l));
        %--   a_j^{m1} * a_k^{m2}      * c_{m1,m2}^l
    if poly then {poly} else {}
    >>)
  end;
```

Mit der Funktion `exponential` werden die ersten Glieder der Exponentialreihe des im Argument übergebenen Wertes berechnet. Da wir Matrizen, die nicht nilpotent sind, sowieso verwerfen, ist in unserem Fall die Betrachtung von 9 Gliedern der Reihe ausreichend.

```
algebraic procedure exponential(m);
  begin scalar j;
  return for j:=0:8 sum m**j/factorial j;
  end;
```

[1] `llsy` (list of list of symbols)

Aus dem Ergebnis der folgenden Routine sind nur die nilpotenten Matrizen zu extrahieren und für weitere Rechnungen zu verwenden.

```
algebraic procedure innerAutomorphisms(dim,sc);
  begin scalar j;
  return for j:=1:dim collect <<exponential(part(sc,j)*mkid('x,j))>>
  end;
```

Da es bei einer großen Menge von Polynomen einen zu hohen Aufwand erfordert, ohne Hilfsmittel zu entscheiden, welche Variablen einer vorgegebenen Menge von Unbestimmten in diesen Polynomen nicht vorkommen, wird auch hier REDUCE eingesetzt. Der folgenden Funktion werden eine Menge `vars` von Variablen und eine Menge `lp`[2] von Polynomen als Argumente übergeben.

```
algebraic procedure freeVariables(vars,lp);
begin scalar p,v;
  return for each v in vars join
    if (for each p in lp sum length coeff(p,v)-1) = 0 then {v} else {};
end;
```

Bei der Behandlung verschiedener Beispiele erweisen sich folgende Funktionen als nützlich.

```
algebraic procedure member!?(u,y);
  if length y=0 then nil else if u=first y then t else member!?(u,rest y);

algebraic procedure setMinus(x,y);
  for each u in x join if member!?(u,y) then {} else {u};
```

Zur Datenkonvertierung benötigen wir weiterhin noch die Prozedur `ll2mat`. Sie wandelt eine Liste von Listen in den REDUCE-Typ Matrix um.

```
symbolic operator ll2mat;
symbolic procedure ll2mat(l);
  begin scalar x;
  return 'mat . for each x in cdr l collect cdr x;
  end;
```

Mit obigen Funktionen und den Funktionen aus dem Modul `cfgb`, das später präsentiert wird, können die Berechnungen der vorn gegebenen Beispiele auf folgende Weise ausgeführt werden. Wir demonstrieren alles am Beispiel der Lie-Algebra $A_{4,7}$.

Die REDUCE-Variable `sc` sei bereits mit den Strukturkonstanten in der oben beschriebenen Weise belegt.

[2] `lp` (list of polynomials)

```
1  dim:=4;
2  ia:=innerAutomorphisms(dim,sc);
3  llsy:=unknowns(dim,'a);
4  aa:=ll2mat llsy;
5  vars:=reverse(append(for j:=1:dim collect part(llsy,j,j),
6    for j:=1:(dim-1) join for k:=1:(dim-j) join
7      {part(llsy,k,j+k),part(llsy,j+k,k)}));
8  polys:=endomorphisms(sc,llsy);
9  setring(vars,{},lex);
10 ipolys:=interReduce polys;
11 setideal(m,ipolys);
12 xvars:={x2,x3};
13 ap:=aa * second ia * third ia;
14 zeros:=for each x in ipolys join if x - mainvar x = 0 then {x} else {};
15 bas  :=for each x in ipolys join if x - mainvar x = 0 then {} else {x};
16 vars :=setMinus(vars,zeros);
17 fvars:=freeVariables(vars,bas);
18 vars :=setMinus(vars,fvars);
19 a21:=a31:=a43:=a41:=a42:=0;
20 trfs :={{ap(1,2),a12},{ap(1,3),a13}};
21 setring(vars,{},lex);
22 setideal(bas,bas);
23 s:=gamma(bas,xvars,trfs);
```

Der Algorithmus **autoReduce** ist in CALI durch die Prozedur `interReduce` implementiert.

Die Variable `polys` repräsentiert die Menge F und `ipolys` die Menge F' aus Kapitel 7.

Die Zeilen 12 und 13 sind abhängig von der konkret betrachteten Lie-Algebra und können nicht allgemein angegeben werden.

In Zeile 19 werden die Variablen mit 0 belegt, von denen bereits geschlossen werden kann, daß sie auch im Endergebnis verschwinden. Dies sind gerade diejenigen, die in Zeile 14 berechnet und in der Variablen `zeros` abgelegt wurden. Die Zeile 19 ist ebenfalls abhängig von der konkreten Lie-Algebra.

In der folgenden Zeile werden die Transformationspolynome ausgewählt.

`setRing` und `setIdeal` sind Prozeduren aus dem Paket CALI und dienen der Vorbereitung einer Gröbnerbasisberechnung.

Anhang B

Implementation von gamma

Für die Implementation von **gamma** kann prinzipiell jedes verfügbare Computeralgebrasystem verwendet werden. Wir geben hier diesen Algorithmus in REDUCE an, da mit CALI ein Paket zur Verfügung stand, das den Buchberger-Algorithmus sowohl mit als auch ohne Faktorisierung bereits in einer „nennerfreien“ Form implementiert. Das Paket CALI muß also in einer REDUCE-Sitzung durch

```
load cali;
```

geladen worden sein, um die unten vorgestellte Funktion `gamma` aufrufen zu können.

Im Programmtext treten außer der Funktion `gamma`, die durch die Anweisung in der zweiten Zeile für den algebraischen Modus verfügbar gemacht wird, noch weitere Funktionen auf. `gamma` dient nur der Konvertierung der Parameter vom algebraischen in den symbolischen Modus, des Aufrufs der Funktion `gamma!*` und der Rückkonvertierung des Ergebnisses. Die dabei aufgerufene Hilfsfunktion `result_2a` gibt ihr Argument in einer algebraischen Präfixform zurück.

Die eigentliche Hauptprozedur, die den Algorithmus **gamma** implementiert, ist `gamma!*`. Es besteht jedoch ein Unterschied. Können nämlich die Transformationspolynome nicht verwendet werden, so wird nicht wie in **gamma** einfach $S_1 = \emptyset$ gesetzt und S_0 durch den Algorithmus **FGB** berechnet, sondern die Berechnung mit der Mitteilung `"Schlechte Wahl"` abgebrochen, da es möglich sein kann, daß der Aufruf von `gamma` mit anderen Transformationspolynomen zu einem günstigeren Ergebnis führt.

Die Funktion `listFGB!*` ist in wesentlichen Teilen eine Kopie der Funktion `listgroebfactor!*` aus dem Quelltext von CALI und wurde für die Berechnungen in `gamma` angepaßt.

```
module cfgb;
symbolic operator gamma;
symbolic procedure gamma(basis,xvariables,transformations);
%++ basis: some polynomials from K[A]
%++ xvariables: list of variables, i.e., {X_1,...X_m}.
%++ transformations: List(R[X],R) where R=K[A].
%++   + the extra information to help to reduce the dimension of the ideal
%++   + an element in this list is a pair (p,v) consisting of the
%++     polynomial p representing the value of the variable v after
%++     the transformation.
%++ Let result := gamma(basis,xvariables,transformations) then
%++ first(result) is a list of pairs (B_i,C_i) where B_i is a GB and
%++ C_i are the corresponding constraints.
%++ The same is true for second(result), but in addition one must apply
%++ the back-transformation to obtain the whole solution set.
%++ The polynomials of the back-transformation are not given explicitly
%++ in the parameter list.
begin scalar vars,bas,trfs,S;
xvars := cdr reval xvariables;
bas := dpmat_from_a reval basis;
trfs := reval transformations; %-- algebraic prefix form is OK
S := gamma!*(bas,xvars,trfs);
return makelist {
  result_2a first S, %-- without group action
  result_2a second S %-- with group action
};
end;
%=======================================================================


symbolic procedure result_2a(R);
%++ R is list of pairs (B,C) (more exactly a list of lists {B,C}) where
%++ B is a dpmat (here: ideal basis), and C is a list of dp_polys.
%++ Returns the same list in algebraic prefix form.
makelist for each BC in R collect makelist
  {dpmat_2a first BC, makelist for each c in second BC collect dp_2a c};
%=======================================================================


symbolic procedure removeSquareFree l;
%++ The input list l represents a product of dpolys. Return a list
%++ which represents the square free part of this product.
%-- (make a set in the mathematical sense)
<<
  if null l then l else
  if dpoly_member!?(first l,cdr l) then removeSquareFree cdr l
  else first(l) . removeSquareFree cdr l
>>;
%=======================================================================
```

```
symbolic procedure dpoly_member!?(x,li);
%++ Test whether the dp_poly x is a member of the list li.
<<
  if null li then nil
  else if null dp_diff(x,first li) then t
  else dpoly_member!?(x,cdr li)
>>;
%=====================================================================

symbolic procedure gamma!*(basis,xvariables,transformations);
%++ basis: dpmat
%++ xvariables: list (lisp mode) of variable names in the trf polys
%++ transformations: algebraic prefix form of transformation of gamma
begin scalar trfbas, tord, ecart, constraints, trfCoords, S0, S1;
tord := degreeorder!* xvariables;
ecart:= first tord;
oldRing := cali!=basering;
(<< %----------------
  setring!* ring_define(xvariables,tord,'revlex,ecart);
  trfbas := dpmat_from_a makelist
    for each x in cdr transformations collect second x;
  trfbas := gbasis!*(trfbas);
  dpmat_print trfbas;

  if dpmat_unitideal!? trfbas then rederr "Schlechte Wahl";

  constraints := makelist for each x in dpmat_list trfbas collect
    bc_2a dp_lc bas_dpoly x
  %----------------
>>) where cali!=basering:= cali!=basering;
setring!* oldRing;
%-- Now the ring is the same ring as before.

constraints := removeSquareFree for each x in cdr constraints join
                 dp_factor dp_from_a x;
%-- We now have the square-free part of the product of all LC's of trfbas.

trfCoords := dpmat_from_a makelist
  for each x in cdr transformations collect third x; %--(first ='list)
%-----------------------
%-- with group action
S1 := groebfactor!*(matsum!*{trfCoords,basis},constraints);
%-----------------------
%-- without group action
S0 := groebf!=newcon({basis,nil},constraints);

%-- inconsistent problems are removed from the list
S0 := for each x in S0 join
  if groebf!=test(second x,dpmat_list first x) then {x};

S0 := listFGB!* S0;
return {S0,S1};
end;
%=====================================================================
```

```
symbolic procedure listFGB!*(R);
%++ R is a list of lists of the form {b,c} where b is a basis (dpmat)
%++ and c is a list of contraints (list of dp_poly).
%++ Returns a list S of results (b,c) such that
%++ \union {Z(B,C) | (B,C) \in R} = \union { Z(b,c) | (b,c) \in S }
%++ The small b's are Gr"obner bases.
%-- This code is based on Gr"abe's listgroebfactor!* in his package CALI.
begin scalar gbs;
  gbs:=for each x in groebf!=preprocess(nil,R) collect
         groebf!=initproblem x;
  gbs:=sort(gbs,function groebf!=problemsort);
  return groebf!=masterprocess(gbs,nil);
end;

endmodule; %-- cfgb
end;
```

Anhang C

Notation

K	(algebraisch abgeschlossener) Körper
R	(euklidischer) Ring (Oft ist $R = K[A]$.)
Q	Quotientenkörper von R
$A_1, \ldots, A_n$	Unbestimmte
$X_1, \ldots, X_m$	weitere Unbestimmte
n	Anzahl der Unbestimmten A_i
m	Anzahl der Unbestimmten X_i
$K[A]$	Abkürzung für $K[A_1, \ldots, A_n]$
$K[X]$	Abkürzung für $K[X_1, \ldots, X_m]$
$\mathcal{A}$	der affine Raum K^n
$\mathcal{X}$	der affine Raum K^m
a	Element von $\mathcal{A}$
x, x'	Elemente von $\mathcal{X}$
$a_1, \ldots, a_n$	Restklassen der Unbestimmten $A_1, \ldots, A_n \in K[A]$ in $K \cong K[A]/\mathfrak{m}$ ($\mathfrak{m}$ maximales Ideal von $K[A]$)
$T(X)$	$T(X_1, \ldots, X_m)$ Terme in $X_1, \ldots, X_m$ (S. 10)
t, s	Elemente von $T(X_1, \ldots, X_m)$
$r, \tilde{r}, r_0, \tilde{r}_0, r_i, \varrho$	Elemente des Rings R
$f, \tilde{f}, f^*$	Polynome aus $R[X]$
$C(f)$	Menge der Koeffizienten des Polynoms f (S. 11)
$T(f)$	Menge der Terme des Polynoms f (S. 11)
$M(f)$	Menge der Monome des Polynoms f (S. 11)
$B = \{b_1, \ldots, b_p\}$	Menge von Polynomen aus $R[X]$, die ein gewisses Ideal $\mathcal{I}(B)$ in $Q[X]$ erzeugen
$G = \{g_1, \ldots, g_k\}$	Menge von Polynomen aus $R[X]$, meist Gröbnerbasis des Ideal $\mathcal{I}(G)$ in $K[X]$
$H = \{h_1, \ldots, h_k\}$	Menge von Polynomen aus $R[X]$, die als Kofaktoren im (erweiterten) Normalformalgorithmus auftreten
$C = \{c_1, \ldots, c_q\}$	Menge von Polynomen aus $K[A]$, die zur Formulierung von „Einschränkungen“ verwendet werden

$\mathcal{I}(B)$	Ideal, das von B erzeugt wird. (S. 11)
$\mathcal{Z}(B)$	Teilmenge von K^n. Menge der gemeinsamen Nullstellen der Polynome von B. (S. 8)
$\mathcal{Z}(B, C)$	Teilmenge von K^n. Andere Schreibweise für die Menge $\mathcal{Z}(B) \setminus \mathcal{Z}(c)$, wobei c das Produkt aller Polynome von C ist. (S. 23)
Γ	Gruppe, die invariant auf $\mathcal{Z}(B)$ operiert und durch $\mathcal{X} = K^m$ parametrisiert ist. (S. 8)
Γ	n-Tupel $(\gamma_1, \ldots, \gamma_n)$ von Polynomen aus $K[X_1, \ldots, X_m, A_1, \ldots, A_n]$ (S. 33)
Γ'	n-Tupel $(\gamma'_1, \ldots, \gamma'_n)$ von Polynomen aus $K[X_1, \ldots, X_m, A_1, \ldots, A_n]$ (S. 33)
$\{\delta_1, \ldots, \delta_t\}$	Teilmenge von $\{\gamma_1, \ldots, \gamma_n\}$

Literaturverzeichnis

[1] T. Becker, V. Weispfenning. *Gröbner Bases*. Springer Verlag, 1993.

[2] B. Buchberger. *Gröbner bases: an algorithmic method in polynomial ideal theory*. In: N. K. Bose (ed.), *Recent Trends in Multidimensional Systems Theory*, D. Reidel Publishing Company, Dortrecht, pp. 184-232.

[3] D. Cox, J. Little, D. O'Shea. *Ideals, Varieties, and Algorithms*. Undergraduate Texts in Mathematics, Springer Verlag, 1992.

[4] J. H. Davenport. *Looking at a Set of Equations*. Bath Computer Science Technical Report 87-06 (1989)

[5] G. Eisenreich. *Lexikon der Algebra*. Akademie-Verlag, Belin, 1989.

[6] K. Gatermann. *Symbolic Solution of Polynomial Equation Systems with Symmetry*. In: S. Watanabe, M. Nagata (ed.), *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC '90)*, ACM Addison-Wesley, New York, 112-119.

[7] A. Giovini, T. Mora, G. Niesi, L. Robbiano, C. Traverso. *"One sugar cube, please," or Selection strategies in the Buchberger algorithm*. In: S. M. Watt (ed.), *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC '91)*, ACM Press, New York, 49-54.

[8] H.-G. Gräbe. *On factorized Gröbner Bases*. In: J. Fleischer et al. (ed.), *Computer Algebra in Science and Engineering: 28-31 August 1994. Bielefeld, Germany*, 77-89, World Scientific, Singapore, 1995.

[9] H.-G. Gräbe. *CALI – A REDUCE Package for Commutative Algebra. Version 2.2.1, June 1995*. Verfügbar über die REDUCE-Bibliothek, z. B. `redlib@rand.org`.

[10] A. C. Hearn. *REDUCE User's Manual Version 3.4*. The RAND Corporation, Santa Monica, 1991.

[11] J. Hietarinta. *Solving the Constant Quantum Yang-Baxter equation in 2 Dimensions with massive use of factorizing Gröbner basis computations* In: P. S. Wang (ed.), *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC '92)*, ACM Press, New York, 350-357.

[12] J. Hilgert, K.-H. Neeb. *Lie-Gruppen und Lie-Algebren*. Vieweg, Braunschweig, 1991.

[13] J. E. Humphreys. *Introduction to Lie Algebras and Representation Theory*. Graduate texts in mathematics 9, 3rd printing, Springer, 1980,

[14] H. Melenk, H.-M. Möller, W. Neun. *Symbolic Solution of Large Stationary Chemical Kinetics Problems*. IMPACT of Computing in Science and Engineering 1, 138-167 (1989).

[15] J. Patera, R. T. Sharp, P. Winternitz, H. Zassenhaus. *Invariants of real low dimension Lie algebras*. In: *Journal of Mathematical Physics*, 986-993, Vol. 17, No. 6, June 1976.

[16] L. S. Pontrjagin. *Topologische Gruppen*. B. G. Teubner, Leipzig, 1958.

Erklärung

Ich versichere, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, Februar 1996 ....

Ralf Hemmecke