Der Algorithmus zur Hironaka-Zerlegung*

nach B.Sturmfels

von Ingolf Schäfer

Inhaltsverzeichnis

1 Einleitung

In diesem Vortrag soll es um den Algorithmus zur Bestimmung der Hironaka-Zerlegung gehen, wie er in [Stu93] beschrieben wird. Dabei geht es um das folgende Problem: Eine endliche Untergruppe $\Gamma < \mathrm{GL}_n(\mathbb{C})$ operiert in natürlicher Weise auf den Polynomen in n Veränderlichen mit komplexen Koeffizienten. Von Interesse ist nun der Invariantenring $\mathbb{C}[X_1, ..., X_n]^\Gamma$ der Gruppe, d.h. die Menge aller Polynome, die unter der Γ-Wirkung fix bleiben.

Ein klassisches Resultat ist nun, dass diese Fixpunktmenge einen Ring bildet, der ein Cohen-Macaulay-Ring ist - man vergleiche die vorherigen Vorträge. Dort wurde ebenfalls bewiesen, dass es in diesem Fall eine sogenannte Hironaka-Zerlegung gibt, d.h. es existieren $\theta_1, ..., \theta_n \in \mathbb{C}[X_1, ..., X_n]^\Gamma$, die homogen sind, so dass $\mathbb{C}[X_1, ..., X_n]^\Gamma$ ein endlich erzeugter, freier $\mathbb{C}[\theta_1, ..., \theta_n]$-Modul ist. Diese θ_i heißen dann auch die **primären Invarianten**. Eine entsprechende Basis $(\eta_1, ..., \eta_{t)}$ aus *homogenen* Elementen heißt dann ein System von **sekundären Invarianten** zu den θ_i. Wir erhalten also die **Hironaka-Zerlegung**:

$$\mathbb{C}[X_1, ..., X_n]^\Gamma = \bigoplus_{i=1}^{t} \eta_i \cdot \mathbb{C}[\theta_1, ..., \theta_n].$$

Zur Warnung sei noch gesagt, dass eine solche Hironaka-Zerlegung überhaupt nicht eindeutig ist.

*. Dieses Dokument wurde mit dem Textverarbeitungsprogramm **GNU TEXMACS** erstellt. (siehe www.texmacs.org

In den vorherigen Vorträgen wurde zwar der grundsätzliche Aufbau des Algorithmus bereits beschrieben, aber wir beginnen dennoch mit einer Wiederholung dieses Aufbaus. Im Anschluss daran wird noch eine minimale Verbesserung des Algorithmus beschrieben. Dabei soll der Algorithmus sowohl in normalen Worten als auch in Pseudocode beschrieben werden.

Schließlich wird noch ein weiterer Algorithmus vorgestellt, um die primären Invarianten auszurechnen, der keine probabilistische Komponente hat. Die hier gezeigten Algorithmen sind allerdings nicht diejenigen, die auch im richtigen Leben eingesetzt werden. Dazu sind die optimierten Algorithmen wie z.B. bei [Kem99] besser geeignet. Wir werden zum Beispiel die Möglichkeiten nicht nutzen, die eine vorherige Berechnung der Molien-Darstellung der Hilbertreihe böte.

2 Der Algorithmus

Der Algorithmus besteht aus zwei Stufen. Zuerst werden die primären Invarianten bestimmt, dann die sekundären Invarianten. Tatsächlich ist diese erste Stufe des Algorithmus, wie sie in Sturmfels beschrieben wird, jedoch nur probabilistisch.

Wir gehen von einer Monomialordnung, die die partielle Ordnung nach dem Multigrad verfeinert, aus und zählen die Monome ab: $m_1 < m_2 < m_3 < \ldots$. Von der endlichen Gruppe brauchen wir erstmal nur den sogenannten Reynolds-Operator *, d.h. zu $f \in \mathbb{C}[X_1, ..., X_n]$ bilden wir $f^* = \sum_{\sigma \in \Gamma} f \circ \sigma$.

Der Algorithmus sieht nun so aus:

Wir gehen die Abzählung der Monome m_i solange einzeln durch, bis das von den m_i^* erzeugte Ideal als Radikal das irrelevante Ideal hat. Sinnvoller Weise überprüfen wir in jedem Schritt zuerst, ob ein m_j^* bereits zum Ideal der vorherigen m_i^* gehört, in diesem Fall nehmen wir gleich das nächste m_{j+1}^*. Dieses Verfahren terminiert und gibt ein System von m_i^*, deren erzeugter Unterring n algebraisch unabhängige, homogene Elemente enthält. Dies sind die primären Invarianten.

Nun müssen die sekundären Invarianten bestimmt werden. Wir setzen zunächst eine Gradschranke $g = -n + \sum_{i=1,...,n} \deg \theta_i$ und gehen nun wieder von Anfang an die Monome m_i bis zum ersten Monom m_i durch, dessen Grad größer als die Gradschranke g ist. Nehmen wir nun alle abhängigen Elemente aus $\{1, m_1, ..., m_{i-1}\}$ heraus, dann ist der Rest ein System von Sekundärinvarianten.

Zur Verdeutlichung derselbe Algorithmus nochmal im Pseudocode, wobei die Querverweise auf die Unterroutinen aus Sturmfels in den Kommentaren stehen.

```
t:INTEGER; bound:INTEGER; P:Set; S:Set;

t:=0; P:=∅;
REPEAT
  REPEAT
    t:=t+1;
  UNTIL Reynolds(m[t]) NOT IN Radical(<P>)     (* Check with Subroutine 2.5.1 *)
  P:=P∪{Reynolds(m[t])},
UNTIL Radical(<P>)=M;                          (* Check with Subroutine 2.5.2 *)

IF NOT Cardinality(P)=n THEN
  P:=Subroutine2.5.10(P);
  (* Modify to algebraically independent set of invariants *)[1]
END;

(* P contains primary invariants now *)
```

1. Wie die Bezeichnung *Modify* schon vermuten lässt, ist diese Unterroutine nur probabilistisch, wie im letzten Vortrag erörtet wurde.

```
S:={1};
t:=0;
bound:=-n;

FOREACH p IN P DO
  bound := bound + Degree(p);
END;

REPEAT
  t:=t+1;
  IF Reynolds(m[t]) NOT IN C[P]_Span(S) THEN   (* Check with Subroutine 2.5.6 *)
    S:=S∪{m[t]};
  END;
UNTIL Degree(Reynolds(m[t]))>bound.

(* now P are the primary, S are the secondary invariants)
```

Im vorherigen Vortrag wurde bereits gezeigt, dass der erste Teil des Algorithmus funktioniert. Es soll jetzt noch gezeigt werden, dass der zweite Teil, also die Bestimmung der sekundären Invarianten funktioniert. Das leistet das folgende Lemma.

Lemma 1. (Sturmfels Lemma 2.5.11)
Sei $P = (\theta_1, ..., \theta_n)$ ein System von primären Invarianten, wie in der Hironaka-Zerlegung oben. Dann ex. ein endliches System S von sekundären Invarianten, deren Grad durch $-n + \sum_{i=1}^{n} \operatorname{grad}(\theta_i)$ beschränkt ist, so dass P und S die Hironaka-Zerlegung liefern.

Beweis. Setze $d_i = \operatorname{grad} \theta_i$. Nach dem Satz über die Grade der sekundären Invarianten (Sturmfels Proposition 2.3.6) gilt, dass der Grad der sekundären Invarianten maximal der Grad des Polynoms

$$p(z) := \phi_\Gamma(z) \cdot \prod_{i=1}^{n} (1 - z^{d_i})$$

ist, wobei

$$\phi_\Gamma(z) = \frac{1}{\#\Gamma} \sum_{\sigma \in \Gamma} \frac{1}{\det(\mathrm{id} - z\sigma)}$$

die Molien-Darstellung der Hilbertreihe bezeichnet. Multiplizieren wir mit dem Hauptnenner der Molien-Reihe, also mit $q(z) := \prod_{\sigma \in \Gamma} \det(\mathrm{id} - z\sigma)$, dann erhalten wir:

$$q(z)p(z) = \phi_\Gamma(z)q(z) \prod_{\sigma \in \Gamma} (1 - z^{d_i}).$$

Wir wollen $\deg p$ abschätzen und betrachten dazu den Grad der rechten Seite. Dieser ist per Konstruktion höchstens $n \cdot^{\#} \Gamma - n + \sum_{i=1}^{n} d_i$. Der Grad der rechten Seite ist aber $n \cdot^{\#} \Gamma + \deg p$, woraus unmittelbar

$$\deg p \leqslant -n + \sum_{i=1}^{n} d_i$$

folgt. □

3 Exkurs: Die Unterroutine 2.5.6

Leider findet sich im Buch von Sturmfels keine Begründung, warum die Unterroutine 2.5.6 funktioniert. Deswegen muss dies hier nachgeholt werden.

Die Unterroutine 2.5.6 bekommt als Eingaben einen Satz primäre Invarianten $\theta_1, ..., \theta_n$ und einige aber nicht notwendig alle Sekundärinvarianten $\eta_1, ..., \eta_k$. Die Unterroutine soll nun prüfen, ob ein gegebenes Polynom f in dem Unterring $R := \bigoplus_{i=1}^{k} \eta_i \, \mathbb{C}[\theta_1, ..., \theta_n]$ liegt.

Die Unterroutine arbeitet nun so: Man führt Hilfs-Unbestimmte $\boldsymbol{Y} = (Y_1, ..., Y_n)$ und $\boldsymbol{Z} = (Z_1, ., , Z_k)$ ein und definiert auf $\mathbb{C}[\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}]$ die folgende Monomialordnung.

$\boldsymbol{X}^\alpha \boldsymbol{Y}^\beta \boldsymbol{Z}^\gamma < \boldsymbol{X}^{\alpha'} \boldsymbol{Y}^{\beta'} \boldsymbol{Z}^{\gamma'}$, wenn $\alpha < \alpha$ bezüglich der rein lexikografischen Ordnung, oder, falls $\alpha = \alpha'$, wenn $\beta < \beta'$ bezüglich der graduierten lexikografischen Ordnung, oder, falls $\alpha = \alpha'$ und $\beta = \beta'$, wenn $\gamma < \gamma'$ bezüglich der rein lexikografischen Ordnung ist.

Wir berechnen jetzt die reduzierte Gröbner-Basis des Ideals $\langle \theta_1 - Y_1, ..., \theta_n - Y_n, \eta_1 - Z_1, ..., \eta_k - Z_k \rangle$. Falls der Rest von f nach Gröbner-Divisionsalgorithmus die Form $\sum_{i=1}^{k} z_i \, p_i(y_1, ..., y_n)$ hat, dann ist f Element von R und hat die eindeutige Zerlegung

$$f(\boldsymbol{x}) = \sum_{i=1}^{k} \eta_i(\boldsymbol{x}) \, p_i(\theta_1(\boldsymbol{x}), ..., \theta_n(\boldsymbol{x})). \tag{1}$$

Zur Begründung stellen wir fest, dass f genau dann in R liegt, wenn f modulo Ideal einen solchen Rest hat. Dass nun genau dieser Rest beim Divisionsalgorithmus herauskommt, liegt an der Monomialordnung.

4 Der Algorithmus an zwei Beispielen

In diesem Abschnitt führen wir die Wirkung des Algorithmus an zwei Beispielen durch, die wir ohne Computerunterstützung behandeln können. Es ist klar, dass wir dabei nicht die Unterroutinen alle konkret ausführen, sondern auch ggfs. die Erkenntnisse der Theorie nutzen, um uns vom Ergebnis der Unterroutine zu überzeugen.

4.1 Beispiel 1

Wie betrachten ein sehr einfaches Beispiel, dessen Lösung wir bereits aus den ersten Vorträgen kennen. Sei $\Gamma = S_3$ und operiere auf den Polynomen in X_1,X_2 und X_3 durch Permutation der Indices.

In der Zykelschreibweise haben wir dann $\Gamma = \{\mathrm{id}, (12), (13), (23), (123), (132)\}$.Wir können nun den Reynolds-Operator explizit für ein gegebenes Monom ausrechnen. Als Monomialordnung wählen wir $X_1 < X_2 < X_3$ als Grundlage von *grlex*.

Das bedeutet wir haben die folgende Abzählung für die Grade 1 und 2:

Grad 1			Grad 2					
m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
X_1	X_2	X_3	X_1^2	X_1X_2	X_2^*	X_1X_3	X_2X_3	X_3^2

Für den Grad 3 erhalten wir:

Grad 3									
m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}	m_{17}	m_{18}	m_{19}
X_1^3	$X_1^2X_2$	$X_1X_2^2$	X_2^3	$X_1^2X_3$	$X_1X_2X_3$	$X_2^2X_3$	$X_1X_3^2$	$X_2X_3^2$	X_3^3

Für den Reynolds-Operator ergibt sich:

$$\begin{aligned}
m_1^* = m_2^* = m_3^* &= \frac{1}{3}(X_1 + X_2 + X_3) \\
m_4^* = m_6^* = m_9^* &= \frac{1}{3}(X_1^2 + X_2^2 + X_3^2) \\
m_5^* = m_7^* = m_8^* &= \frac{1}{3}(X_1X_2 + X_1X_3 + X_2X_3) \\
m_{10}^* = m_{13}^* = m_{19}^* &= \frac{1}{3}(X_1^3 + X_2^3 + X_3^3) \\
m_{11}^* = m_{12}^* = m_{14}^* = m_{16}^* = m_{17}^* = m_{18}^* &= \frac{1}{6}(X_1^2X_2 + X_1X_2^2 + X_1^2X_3 + X_2^2X_3 + X_1X_3^2 + X_2X_3^2) \\
m_{15}^* &= X_1X_2X_3 \\
&\vdots
\end{aligned}$$

Damit können wir jetzt den Algorithmus durchlaufen. Der Übersicht halber zerlegen wir den Algorithmus wieder in die beiden Teil-Algorithmen zur Bestimmung der primären und sekundären Invarianten.

4.1.1 Die primären Invarianten

Wir beginnen mit m_1^*. Dies ist nicht im Radikal des Nullideals und muss also zu P hinzugenommen werden. Das von $\theta_1 := m_1^*$ erzeugte Ideal hat aber nicht das irrelevante Ideal als Radikal. Wir müssen also weitersuchen. Die beiden Monome m_2 und m_3 bringen nichts Neues.

Mit $\theta_2 := m_4^*$ erhalten wir eine weitere primäre Invariante. Das Radikalideal von $\langle \theta_1, \theta_2 \rangle$ ist aber ungleich dem irrelevanten Ideal. Das m_5^* bis m_9^* keine weiteren primären Invarianten geben, kann man mit Hilfe expliziter Gröbner-Rechnung nach Unterroutine 2.5.2 testen. Wir lösen dies hier durch die Angabe einer direkten Darstellung: $\theta_2 - \theta_1^2 = -2m_5^*$.

Mit $\theta_3 := m_{10}^*$ finden wir die letzte primäre Invariante. Das $\langle \theta_1, \theta_2, \theta_3 \rangle$ als Radikal das irrelevante Ideal hat, rechnet man mit Unterroutine 2.5.2 nach oder benutzt die Theorie. Damit haben wir die primären Invarianten gefunden. Man kann bespielsweise direkt ausrechnen, dass aus $\theta_1 = \theta_2 = \theta_3 = 0$, sofort $X_1 = X_2 = X_3 = 0$ folgt.

4.1.2 Die sekundären Invarianten

Zuerst rechnen wir die Schranke aus:

$$g = 1 + 2 + 3 - 3 = 3.$$

Als erste sekundäre Invariante wählen wir $\eta_1 = 1$. Dann gehen wir die Monome durch. Da für alle Monome mit Grad höchstens drei der Wert des Reynolds-Operator immer in $\mathbb{C}[\theta_1, \theta_2, \theta_3]$ liegt, bleibt dies auch die einzige sekundäre Invariante.

An dieser Stelle bietet sich natürlich eine erste Optimierung des Algorithmus an. Zumindest sollten wir doch nur solange nach zweiten Invarianten suchen, bis wir die Anzahl

$$t = \frac{d_1 \cdot \ldots \cdot d_n}{\#\Gamma}$$

erreicht haben. In unserem Fall ist das

$$t = \frac{1 \cdot 2 \cdot 3}{6} = 1.$$

Die gesuchte Zerlegung ist damit:

$$\mathbb{C}[X_1, X_2, X_3]^\Gamma \simeq \mathbb{C}[\theta_1, \theta_2, \theta_3].$$

4.2 Beispiel 2

Um die Sache etwas spannender zu gestalten, betrachten wir jetzt die Untergruppe $\Gamma' = \{\mathrm{id}, (123), (132)\}$, also die Operation dieser Untergruppe auf den Polynomen in drei Veränderlichen. Man rechnet leicht nach, dass wir für Grad kleiner 3 den gleichen Reynolds-Operator wie oben erhalten, für Grad 3 gilt allerdings:

$$\begin{aligned} m_{10}^* = m_{13}^* = m_{19}^* &= \frac{1}{3}(X_1^3 + X_2^3 + X_3^3) \\ m_{11}^* = m_{14}^* = m_{16}^* &= \frac{1}{3}(X_1^2 X_2 + X_2^2 X_3 + X_1 X_3^2) \\ m_{12}^* = m_{17}^* = m_{18}^* &= \frac{1}{3}(X_1 X_2^2 + X_1^2 X_3 + X_2 X_3^2) \\ m_{15}^* &= X_1 X_2 X_3 \end{aligned}$$

4.2.1 Die primären Invarianten

Da der Reynolds-Operator bis zum ersten Monom des Grads 3 übereinstimmt, also erst ab m_{11} unterschiedlich ist, erhalten wir die gleichen primären Invarianten.

4.2.2 Die sekundären Invarianten

Für die Anzahl der sekundären Invarianten gilt diesmal allerdings

$$t = \frac{1 \cdot 2 \cdot 3}{3} = 2.$$

Wir brauchen also außer $\eta_1 = 1$ noch eine weitere sekundäre Invariante. Der Algorithmus liefert hier dann $\eta_2 = m_{11}^*$.

Die gesuchte Zerlegung ist damit:

$$\mathbb{C}[X_1, X_2, X_3]^\Gamma \simeq \mathbb{C}[\theta_1, \theta_2, \theta_3] \oplus \eta_2 \mathbb{C}[\theta_1, \theta_2, \theta_3].$$

5 Diskussion und ein deterministischer Algorithmus

Der eingesetzte Algorithmus ist, wie wir im Beispiel bereits sahen, weit von einem optimalen Algorithmus entfernt. Eine bessere und genau beschriebe Implementierung in Maple findet sich bei [Kem93]. Derselbe Autor hat später ([Kem99]) auch noch einen optimalen Algorithmus angegeben, der die Grade der primären Invarianten in einer *spezifizierbaren* Weise optimiert.

Wir können unseren Algorithmus natürlich auch dadurch verbessern, dass wir zuerst die Hilbert-Reihe berechnen und damit ja schon Aussagen über die Anzahl der Invarianten eines bestimmten Grads haben. Dies ersparte bereits in unseren Beispielen etliche Durchläufe der ersten Schleife.

An dieser Stelle soll allerdings der sogenannte Algorithmus von Dade in Form aus [Stu93] wiedergegeben werden. Das Hauptproblem an unserem Algorithmus ist, dass wir im schlechten Fall primäre Invarianten unnötig hohen Grades produzieren. Es kann sogar vorkommen, dass die Grade der primären Invarianten größer als die Gruppenordnung ist. Der Algorithmus von Dade erzeugt nun primäre Invarianten, deren Grade jeweils die Gruppenordnung teilen. Dies ist unseren Beispielen glücklicherweise von selbst der Fall. Außerdem ist dieser Algorithmus deterministisch.

Wir wollen dies Verfahren allerdings nicht im Pseudocode formulieren, sondern begnügen uns mit einer skizzenhaften Beschreibung:

Der Algorithmus von Dade zur Bestimmung der primären Invarianten
Suche rekursiv Linearformen $\ell_i \in \mathbb{C}^{n*}$, d.h. lineare Polynome, die auf $\ell_1 \circ \sigma_1$, ..., $\ell_{i-1} \circ \sigma_{i-1}$ für alle σ_1, ..., $\sigma_{i-1} \in \Gamma_<$ nicht verschwinden. Dann sind die $\theta_i = \prod_{A_i} \ell_i \circ \sigma$ primäre Invarianten, wobei $A_i = \{\ell_i \circ \sigma \,|\, \sigma \in \Gamma\}$.

Die Bestimmung solcher Linearformen ist mit Mitteln der linearen Algebra möglich und der Grad der θ_i ist per Konstruktion sicher Teiler der Gruppenordnung. Da per Konstruktion die gemeinsame Nullmenge der θ_i nur der Ursprung ist und die θ_i auch homogene Invarianten sind, haben wir also wirklich einen Satz von primären Invarianten gefunden.

Es stellt sich nun die Frage, ob sich dieser Algorithmus auch lohnt, denn, obwohl es sich ja nur um Rechnungen aus der Linearen Algebra handelt, kann es bei großer Gruppenordnung natürlich schnell sehr aufwändig werden. Einen Vergleich des Rechnenaufwandes findet man in [Kem99].

Literaturverzeichnis

[Kem93] Kemper, Gregor: *The Invar Package for Calculating Rings of Invariants.* IWR, Heidelberg, 1993.

[Kem99] Kemper, Gregor: *An Algorithm to Calculate Optimal Homogeneous Systems of Parameters.* J. Symbolic Computation, 27:171–184, 1999.

[Stu93] Sturmfels, Bernd: *Algorithms in Invariant Theory.* Texts and Monographs in Symbolic Computation. Springer-Verlag, 1993.