# Algebraische Komplexitätstheorie I

# Eine Einführung

Peter Bürgisser, Universität Zürich
Michael Clausen, Universität Bonn

Im Rahmen des 36. Treffens des Séminaire Lotharingien de Combinatoire vom 19.-22.3.96 auf Schloß Thurnau hatten wir die Gelegenheit, in drei Vorträgen Entwicklungen der algebraischen Komplexitätstheorie vorzustellen. Die vorliegende Arbeit stellt eine Ausarbeitung des ersten Vortrags dar, dessen Ziel es war, eine sanfte Einführung in die Thematik zu vermitteln. Im zweiten und dritten Vortrag (siehe die nachfolgenden Ausarbeitungen) wurden dann zwei wichtige, auch kombinatorisch geprägte Teilgebiete der algebraischen Komplexitätstheorie mehr im Detail vorgestellt.

## 1 Vorbemerkungen

Die algorithmische Lösung von Berechnungsproblemen hat seit jeher zu den Hauptaufgaben der Mathematik gehört. Lange Zeit basierten derartige Lösungen auf einem intuitiven Berechenbarkeitsbegriff. Erst in der ersten Hälfte dieses Jahrhunderts hat man, angeregt durch metamatematische Fragestellungen, intensiv nach einer allgemeinen Formalisierung des intuitiven Berechenbarkeitsbegriffs gesucht.

Verschiedene formale Modelle wurden vorgeschlagen, wie z.B. Turingmaschinen, Registermaschinen, While-Programme, rekursive Funktionen, Markov-Algorithmen und Thue-Systeme. Diese Konzepte stellten sich aber alle als äquivalent heraus, was dann zu der Überzeugung führte, damit eine adäquate

Explikation des intuitiven Berechenbarkeits- und Algorithmenbegriffs gefunden zu haben (These von Church). Dies hatte und hat enorme Auswirkungen. Zunächst einmal war es jetzt möglich, die algorithmische Unlösbarkeit von Problemen zu beweisen. Berühmte Beispiele sind das Halteproblem, das Postsche Korrespondenzproblem, das Wortproblem der Gruppentheorie oder Hilberts 10. Problem. Auf der anderen Seite hatten Konzepte wie Turingmaschinen und While-Programme einen großen Einfluß auf die Entwicklung der ersten Universalrechner und deren Programmierung.

Im Zeitalter der Digitalrechner trat dann die Frage nach effizienten Algorithmen immer mehr in den Vordergrund. Manche Probleme lassen sich sehr effizient lösen, während andere, darunter zahlreiche praxisrelevante, sich allen Anstrengungen zur effizienten Lösung widersetzten. Diese Tatsache führte auf die Frage nach einem tieferen Verständnis der inhärenten Schwierigkeiten bei der algorithmischen Lösung von Berechnungsproblemen, woraus die Komplexitätstheorie entstand. Seither hat sich dieses Gebiet stürmisch entwickelt, und es entstanden eine Fülle von Teildisziplinen. Jede dieser Disziplinen benutzt spezielle Berechnungsmodelle, wie z.B. Turingmaschinen, RAMs (Random Access Machines), Boolesche Schaltkreise, straight-line Programme, Berechnungsbäume, oder VLSI-Modelle. Jede Berechnung in einem solchen Modell verursacht Kosten, wie die Anzahl der Rechenschritte einer Turingmaschine oder RAM, der benötigte Speicherplatz, die Anzahl der Gatter in einem Schaltkreis, die Anzahl der Befehle oder die Größe der Chipfläche. Dementsprechend finden Untersuchungen in der Komplexitätstheorie immer auf der Basis eines Berechnungsmodells und eines gewählten Kostenmaßes statt. Unter der *Komplexität* eines Problems versteht man jeweils die minimalen Kosten, die zu seiner Lösung im Rahmen des gewählten Berechnungsmodells erforderlich sind. Für einen Überblick über viele Gebiete der Komplexitätstheorie verweisen wir auf [11].

In unseren drei Vorträgen geht es um algebraische Komplexitätstheorie. Darunter versteht man die Untersuchung der Komplexität algebraischer Berechnungsprobleme im Rahmen algebraischer Berechnungsmodelle, deren wichtigste die straight-line Programme und Berechnungsbäume sind. Die algebraische Komplexitätstheorie entstand aus Fragen der numerischen Mathematik und des symbolischen Rechnens. Das Jahr 1954 wird oft als ihr Geburtsjahr angesehen, als nämlich Alexander Ostrowski [17] die Frage nach der Optimalität der Hornerschen Regel zur Diskussion stellte. Systematische Forschung wurde in der algebraischen Komplexitätstheorie seit Anfang der 70er Jahre betrieben; einen ziemlich aktuellen Überblick geben die Artikel von Strassen [22] und von zur Gathen [9]. Bisher gab es nur die Monographie von Borodin und Munro [5] zu diesem Thema, welche vergriffen und schon lange nicht mehr auf dem neuesten Stand ist. Demnächst erscheint aber unsere umfassende Darstellung der algebraischen Komplexitätstheorie in Buchform [7].

## 2 Straight-Line Programme und Komplexität

In diesem Abschnitt wird das einfachste und zugleich wichtigste Berechnungsmodell der algebraischen Komplexitätstheorie vorgestellt: nämlich straight-line Programme. Auf der Basis dieses Modells und einem Kostenmaß können wir dann die *Komplexität* vieler (algebraischer) Berechnungsprobleme definieren.

Grob gesprochen ist ein straight-line Programm ein Verfahren, das von bestimmten algebraischen Eingaben ausgeht (oft sind das Unbestimmte über einem Körper $k$) und schrittweise mittels algebraischer Operationen wie Addition, Subtraktion, Skalarmultiplikation, Multiplikation oder Division Zwischenergebnisse produziert. Einmal produzierte Zwischenergebnisse können später ohne Kosten wiederverwendet werden. Die Berechnung kann beendet werden, wenn die zu berechnenden Größen unter den Zwischenergebnissen vorkommen.

Die nachfolgende Definition präzisiert diese Vorstellung.

**Definition 1**

- *Eine* Eingabe *der Länge $n$ über einem Körper $k$ ist von der Form $(A; a)$ mit einer $k$-Algebra $A$ und einem $a \in A^n$. (Dabei stellt $A$ den Bereich dar, in dem die Berechnungen stattfinden sollen.)*

- *(Syntax) Ein* straight-line Programm*, das Eingaben der Länge $n$ über $k$ erwartet, ist eine endliche Folge $\Gamma = (\Gamma_1, \ldots, \Gamma_r)$ von Anweisungen der Form $\Gamma_i = (\omega_i; \alpha, \beta)$, falls $\omega_i \in \{+, -, *, /\}$ oder $\Gamma_i = (\omega_i; \alpha)$, falls $\omega_i \in k$. Dabei ist $k$ ein Körper und $-n < \alpha, \beta < i$.*

- *(Semantik) Ein straight-line Programm* $\Gamma = (\Gamma_1, \ldots, \Gamma_r)$ *angesetzt auf die Eingabe* $(A; a)$ *der Länge* $n$ *produziert, sofern ausführbar, eine* Resultatfolge $(A; b)$
$$b = (\underbrace{b_{-n+1}, \ldots, b_0}_{:=a}, b_1, \ldots, b_r),$$
*die folgendermaßen definiert ist: für alle* $1 \leq i \leq r$ *gilt*
$$b_i = \begin{cases} b_\alpha \, \omega_i \, b_\beta & \textit{falls } \Gamma_i = (\omega_i; \alpha, \beta) \\ \omega_i \cdot b_\alpha & \textit{falls } \Gamma_i = (\omega_i; \alpha). \end{cases}$$
*(Die Ausführbarkeit bedeutet, daß nur Divisionen durch Einheiten von* $A$ *vorkommen.) Man sagt, daß* $\Gamma$ *die Elemente* $b_i$ *berechnet.*

- *(Kosten) Bezüglich einer* Kostenfunktion $c : \{+, -, *, /\} \cup k \to \mathbb{N}$ *definiert man die* $c$*-Länge eines straight-line Programms* $\Gamma$ *als* $\sum_i c(\omega_i)$.

- *(Komplexität) Die* Komplexität $L_A^c(F|I)$ *von* $F \subseteq A$ *modulo der Eingabemenge* $I \subseteq A$ *bzgl.* $c$ *ist definiert als die minimale* $c$*-Länge eines straight-line Programms* $\Gamma$*, das aus einer Eingabe der Form* $(A; a)$ *mit* $a \in I^n$, $n \in \mathbb{N}$*, alle Elemente von* $F$ *berechnet.*

Jede in dem gesteckten Rahmen zugelassene Berechnung liefert eine obere Schranke für die Komplexität des Berechnungsproblems. Insofern gibt es hier enge Verbindungen zur Computer Algebra, deren vornehmliche Aufgabe der Entwurf und die Implementation effizienter algebraischer Algorithmen ist. Die Hauptaufgabe der algebraischen Komplexitätstheorie ist die Herleitung von nichttrivialen unteren Schranken. Dies ist meistens ziemlich schwierig, da man ja gegen alle nur denkbaren Algorithmen argumentieren muß, die zugelassen sind und das Problem lösen. Im günstigsten Fall, wenn obere und untere Schranke übereinstimmen, hat man einen *optimalen* Algorithmus gefunden.

Wir illustrieren dies am Beispiel der Polynomauswertung.

**Beispiel.** Zu berechnen sei das allgemeine Polynom $n$-ten Grades
$$p := a_0 X^n + a_1 X^{n-1} + \ldots + a_n X^0$$
aus den Größen $a_0, \ldots, a_n, X$, die wir als Unbestimmte über einem Körper $k$ auffassen. Gerechnet werden soll im rationalen Funktionenkörper $A$ erzeugt durch diese Unbestimmten über $k$. Ostrowski folgend zählen wir nur

die Multiplikationen und Divisionen und betrachten die linearen Operationen als kostenlos. Das heißt, wir verwenden die Kostenfunktion

$$c(*) = c(/) = 1, \ c(+) = c(-) = 0, \ \forall \omega \in k : c(\omega) = 0,$$

und schreiben für die resultierende *nichtskalare* oder *Ostrowski-Komplexität* statt $L_A^c$ einfach $L$. Diese Zählung mag unrealistisch erscheinen. Bedenken Sie aber, daß für die Ostrowski-Komplexität erzielte untere Schranken gültig bleiben, wenn man alle Operationen zählt!

Die Hornersche Regel zur Polynomauswertung basiert auf der "Formel":

$$p = (\ldots(((a_0 \cdot X + a_1) \cdot X + a_2) \cdot X + a_3)\ldots) \cdot X + a_n.$$

Tatsächlich handelt es sich hier um ein straight-line Programm: in einem ersten Schritt berechnen wir $b_1 := a_0 \cdot X$; dann wird $a_1$ zu diesem Zwischenresultat hinzuaddiert, wodurch man das zweite Zwischenresultat erhält: $b_2 := b_1 + a_1$. Dieses wird dann mit $X$ multipliziert: $b_3 := b_2 \cdot X$, usw. Indem wir $b_{-n-1} := X$ und $b_{-i} := a_i$ setzen, können wir die Hornersche Regel wie folgt schreiben:

$$b_{2i} := b_{2i-1} + b_{-i} \quad \text{und} \quad b_{2i-1} := b_{2i-2} \cdot b_{-n-1},$$

für alle $1 \leq i \leq n$. Die Folge der $b_i$ ist die Resultatfolge, also die semantische Seite des zugehörigen straight-line Programms $\Gamma = (\Gamma_1, \ldots, \Gamma_{2n})$,

$$\Gamma_{2i} := (+; 2i-1, -i) \quad \text{und} \quad \Gamma_{2i-1} := (*; 2i-2, -n-1),$$

bezüglich der Eingabe $(A; (X, a_n, \ldots, a_0))$. Die Hornersche Regel kommt neben $n$ (kostenlosen) Additionen mit $n$ Multiplikationen aus. Dies liefert eine obere Schranke für die Ostrowski-Komplexität der Polynomauswertung:

$$L(\sum_{i=0}^{n} a_i X^{n-i} | X, a_0, \ldots, a_n) \leq n.$$

Ostrowski [17] hat 1954 vermutet, daß sogar Gleichheit gilt, war aber nur in der Lage, seine Vermutung für $n \leq 4$ zu beweisen. Der Beweis des allgemeinen Falles gelang erst 12 Jahre später Pan [18] mittels einer Technik, die wir heute Substitutionsmethode nennen. Dabei geht man, grob gesprochen, von einer hypothetischen optimalen Berechnung aus und trivialisiert das erste Zwischenergebnis, das Kosten verursacht hat, durch eine geeignete lineare Substitution der Unbestimmten. Dies führt das Originalproblem in ein einfacheres Problem über. Nun bleibt zu zeigen, daß eine vollständige Trivialisierung des Problems eine Mindestanzahl von Substitutionsschritten erfordert. Diese Anzahl, die typischerweise linear in der Anzahl der Unbestimmten ist, liefert dann eine untere Schranke.

# 3 Gradschranke

Wir diskutieren jetzt eine Methode, die es in vielen Fällen erlaubt, nichtlineare untere Komplexitätsschranken zu beweisen. Wieder beginnen wir mit einem konkreten Beispiel.

**Beispiel.** (Mehrfache Polynomauswertung)
In den Anwendungen tritt nicht selten das Problem auf, ein und dasselbe Polynom an vielen Stellen auszuwerten. Es soll etwa das Polynom $p$ vom Grad $n$ an $n+1$ vorab unbekannten Stellen ausgewertet werden. Das läuft auf die Berechnung von $p(X_0), \ldots, p(X_n)$ hinaus, wo $X_0, \ldots, X_n$ Unbestimmte über $k$ sind: das heißt, wir haben die Vandermonde-Matrix $(X_i^j)_{0 \le i,j \le n}$ mit dem Vektor $(a_n, \ldots, a_0)^\top$ zu multiplizieren.

Wenden wir $n+1$ mal die Hornersche Regel an, so ergibt sich für die nichtskalare Komplexität dieses Problems die obere Schranke $n(n+1)$. Überraschenderweise geht es aber viel schneller! So kommt man z.B. im Fall $k = \mathbb{C}$ mit $O(n \log n)$ Multiplikationen und insgesamt mit $O(n \log^2 n)$ arithmetischen Operationen aus [8, 15, 20, 4].

Diese verbesserte obere Schranke basiert auf folgenden Fakten:

- Polynommultiplikation sowie Division mit Rest sind in der Ostrowski-Zählung mit linearem Aufwand möglich.
- Berechnung der elementarsymmetrischen Polynome $\sigma_1, \ldots, \sigma_n$ in $n$ Unbestimmten geht mit nichtskalarem Aufwand $n \log n$.

Wir bemerken, daß bei Zählung aller Operationen höchstens ein Faktor $\log n$ hinzukommt.

Jetzt erläutern wir die Grundidee der verbesserten oberen Schranke. Sei $q$ ungefähr $n/2$. Wir berechnen die Koeffizienten des Produkts $\prod_0^q (X - X_i)$ (elementarsymmetrische Polynome!) und führen dann Division mit Rest durch:

$$p(X) = g(X)(X - X_0) \cdots (X - X_q) + h(X).$$

Das Restpolynom $h$ hat ungefähr Grad $n/2$ und nimmt an den Stellen $X_i$, $0 \leq i \leq q$, dieselben Werte an wie $p$. Analog berechnen wir bzgl. der anderen Variablen ein Restpolynom $h_1$ und fahren dann mit $h$ und $h_1$ rekursiv weiter.

Im folgenden bereiten wir die Herleitung einer unteren Schranke der Größenordnung $n \log n$ für dieses Problem vor.

Es sei $\Gamma = (\Gamma_1, \ldots, \Gamma_r)$ ein divisionsfreies straight-line Programm, das bei Eingabe $(k[X]; X)$ die Resultatfolge $(k[X]; b)$ produziert; dabei bezeichne $X = (X_1, \ldots, X_n)$ eine Folge von $n$ Unbestimmten über $k$. Für den Grad des $i$-ten Zwischenergebnisses $b_i$ erhält man durch einfache Rechnung:

$$\deg(b_i) \leq 2^{\mu_i},$$

wo $\mu_i := \#\{j \leq i | \omega_j = *\}$ die Anzahl der Multiplikationsanweisungen des Anfangssegments $(\Gamma_1, \ldots, \Gamma_i)$ ist. Wenn wir annehmen, daß $\Gamma$ das Polynom $f = b_r$ optimal im Sinne der Ostrowski-Zählung berechnet, so können wir daraus auf die Gültigkeit der folgenden unteren Schranke schließen:

$$L(f|X) \geq \log \deg(f).$$

Um dies zu verallgemeinern, werden wir einer endlichen Menge von multivariaten Polynomen einen Grad zuordnen. Dazu benötigen wir einige Grundbegriffe aus der algebraischen Geometrie, an die wir zuerst erinnern wollen.

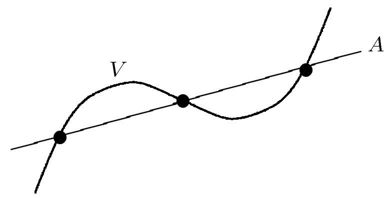
Im folgenden bezeichne $k$ einen algebraisch abgeschlossenen Körper. Zu jeder Menge $F$ von $n$-variaten Polynomen über $k$ gehört ein Nullstellengebilde

$$Z(F) := \{\xi \in k^n \mid \forall f \in F : f(\xi) = 0\}.$$

$Z(F) \subseteq k^n$ ist eine *affine Varietät.* Andererseits kann man jeder Teilmenge $V$ von $k^n$ ein Ideal in $k[X] := k[X_1, \ldots, X_n]$ zuordnen, ihr sogenanntes *Verschwindungsideal*

$$I(V) := \{f \in k[X] \mid \forall \xi \in V : f(\xi) = 0\}.$$

Eine affine Varietät $V$, die nicht als echte Vereinigung affiner Varietäten darstellbar ist, heißt *irreduzibel.* Dies ist genau dann der Fall, wenn $I(V)$ ein Primideal, also $k[X]/I(V)$ ein Integritätsbereich ist. Eine irreduzible affine Varietät $V \subseteq k^n$ besitzt eine *Dimension* $d \leq n$, welche definiert ist als der Transzendenzgrad des Quotientenkörpers von $k[X]/I(V)$ über $k$. Man kann nun einer irreduziblen Varietät $V \subseteq k^n$ der Dimension $d$ einen Grad zuordnen, der folgendermaßen charakterisiert ist: Fast jeder affine Teilraum $A$ von $k^n$ von komplementärer Dimension $n-d$ trifft die Varietät $V$ in der gleichen endlichen Anzahl Punkte. Diese Anzahl nennt man den *(geometrischen) Grad* $\deg V$ von $V$.




$V = Z(Y - X^3 + X) \subset k^2$, $\dim V = 1$, $\deg V = 3$

Den Grad einer nicht notwendig irreduziblen affinen Varietät definiert man in der Komplexitätstheorie etwas anders als es in der algebraischen Geometrie üblich ist, nämlich als die Summe der Grade der irreduziblen Komponenten von $V$. Dies hat den Vorteil, daß die folgende Variante des Bézoutschen Satzes ohne Einschränkung gültig ist, was von zentraler Bedeutung für alle Anwendungen ist. (Für einen Beweis siehe z.B. [7].)

**Satz 2 (Bézoutsche Ungleichung)** *Für affine Varietäten $V$, $W$ in $k^n$ gilt*

$$\deg V \cap W \leq \deg V \cdot \deg W.$$

Nun wollen wir den Zusammenhang zur algebraischen Komplexitätstheorie herstellen. Eine zu berechnende endliche Folge $f_1, \ldots, f_m$ von $n$-variaten Polynomen über $k$ definiert eine polynomiale Abbildung $k^n \to k^m$, deren Graph eine affine Varietät ist, die durch die Gleichungen

$$Y_i - f_i(X_1, \ldots, X_n) = 0 \quad (1 \leq i \leq m)$$

beschrieben ist. Definiert man den Grad von $f_1, \ldots, f_m$ als den Grad des zugehörigen Graphen,

$$\deg(f_1, \ldots, f_m) := \deg \operatorname{graph}(f_1, \ldots, f_m),$$

so besitzt dieser Grad einige wichtige Eigenschaften.

- $\deg(f_1, \ldots, f_m) \leq \deg(b_1, \ldots, b_r)$, falls $(f_1, \ldots, f_m)$ eine Teilfolge von $(b_1, \ldots, b_r)$ ist.
- $\deg(b_1, \ldots, b_r, \alpha b_i + \beta b_j) = \deg(b_1, \ldots, b_r)$, falls $i, j \leq r$ und $\alpha, \beta \in k$.
- $\deg(b_1, \ldots, b_r, b_i * b_j) \leq 2 \deg(b_1, \ldots, b_r)$, falls $i, j \leq r$.

Die erste Eigenschaft paßt sehr gut zu unserer Ausgangssituation bei der Herleitung unterer Komplexitätsschranken: Typischerweise gehen wir von einer hypothetischen optimalen Berechnung mit Resultatfolge $(b_i)_{i \leq r}$ aus. Wir kennen aber nur Bruchstücke dieser Resultatfolge, eben die zu berechnenden $f_1, \ldots, f_m$. Die zweite Eigenschaft paßt sehr gut zur Ostrowski-Zählung: lineare Operationen erhöhen den Grad nicht! Die letzte Eigenschaft besagt, daß eine nichtskalare Multiplikation den Grad höchstens um den Faktor zwei erhöht. Dies ist im wesentlichen eine Folge der Bézoutschen Ungleichung, denn

$$\operatorname{graph}(b_1, \ldots, b_r, b_i * b_j) = (\operatorname{graph}(b_1, \ldots, b_r) \times k) \cap Z(Y_{r+1} - Y_i Y_j),$$

und $Z(Y_{r+1} - Y_i Y_j)$ hat den Grad 2. Damit sind wir bei der sogenannten *geometrischen Gradschranke* angelangt.

**Satz 3 (Strassen [20])** *Sind $f_1, \ldots, f_m \in K := k(X_1, \ldots, X_n)$, so ist*

$$L(f_1, \ldots, f_m | X_1, \ldots, X_n) \geq \log \deg(f_1, \ldots, f_m).$$

*Wenn $m = n$ und die $f_1, \ldots, f_n$ algebraisch unabhängig über $k$ sind, so gilt*

$$L(f_1, \ldots, f_n | X_1, \ldots, X_n) \geq \log[K : k(f_1, \ldots, f_n)].$$

Für die elementarsymmetrischen Polynomen $\sigma_1, \ldots, \sigma_n$ in den Unbestimmten $X_1, \ldots, X_n$ folgt mittels Galoistheorie: $[K : k(\sigma_1, \ldots, \sigma_n)] = |S_n| = n!$. Die geometrische Gradschranke liefert deshalb

$$L(\sigma_1, \ldots, \sigma_n) \geq \log(n!) \geq n(\log n - 2).$$

Im Beispiel der mehrfachen Auswertung eines Polyoms erhält man als Grad der Körpererweiterung nach leichter Rechnung $n^{n+1}$. Dies liefert die untere Schranke $(n+1)\log n$ und zeigt, daß der früher skizzierte Algorithmus bis auf einen konstanten Faktor optimal ist.

# 4 Berechnungsbäume und Grad

Bis jetzt betrachteten wir nur Algorithmen, welche arithmetische Operationen ausführen, aber nicht die Fähigkeit haben, Zwischenresultate zu vergleichen und demgemäß zu verzweigen. Solche Algorithmen nannten wir straight-line Programme. Für manche Berechnungsprobleme ist dieses Modell aber zu eingeschränkt. Zum Beispiel testet Gaußelimination Zwischenresultate auf Null, um die Pivotelemente zu finden.

Ein Modell für algebraische Algorithmen, die auch Tests durchführen, wird durch die *algebraischen Berechnungsbäume* geliefert. Abb. 1 beschreibt schematisch einen algebraischen Berechnungsbaum für die komplexe Division. Zu einer Eingabe $(a,b,c,d) \in \mathbb{R}^4$ mit $cd \neq 0$ wird ein Paar $(x,y) \in \mathbb{R}^2$ berechnet so, daß $x + iy = (a+ib)/(c+id)$, d.h.

$$x = \frac{ac+bd}{c^2+d^2} \quad \text{und} \quad y = \frac{bc-ad}{c^2+d^2}.$$

Falls $c = d = 0$ ist, endet die Rechnung mit der Fehlermeldung $\uparrow$. Demgemäß ist ein algebraischer Berechnungsbaum ein binärer Baum, der neben Eingabe- und Ausgabeknoten noch Berechnungs- und Testknoten enthält. Die Berechnungsknoten haben Ausgrad 1 und enthalten die Instruktionen für die auszuführenden arithmetischen Operationen; die Testknoten haben Ausgrad zwei und enthalten die Anweisungen für die Tests auf Gleichheit. Jede Eingabe – in unserem Beispiel jedes 4-Tupel $(a,b,c,d)$ reeller Zahlen – definiert einen eindeutig bestimmten Pfad von der Wurzel zu einem Blatt in diesem Baum. Die Ausgabe ist eine Teilmenge der berechneten Zwischenresultate, gegebenenfalls zusammen mit der Information in welchem Blatt des Baums man gelandet ist. Manchmal erlaubt man auch Tests “$b_i \leq b_j$” auf kleiner-gleich, was wir aber in diesem Abschnitt nicht betrachten wollen.

An dieser Stelle ist eine grundsätzliche Bemerkung über die Beziehung zwi-

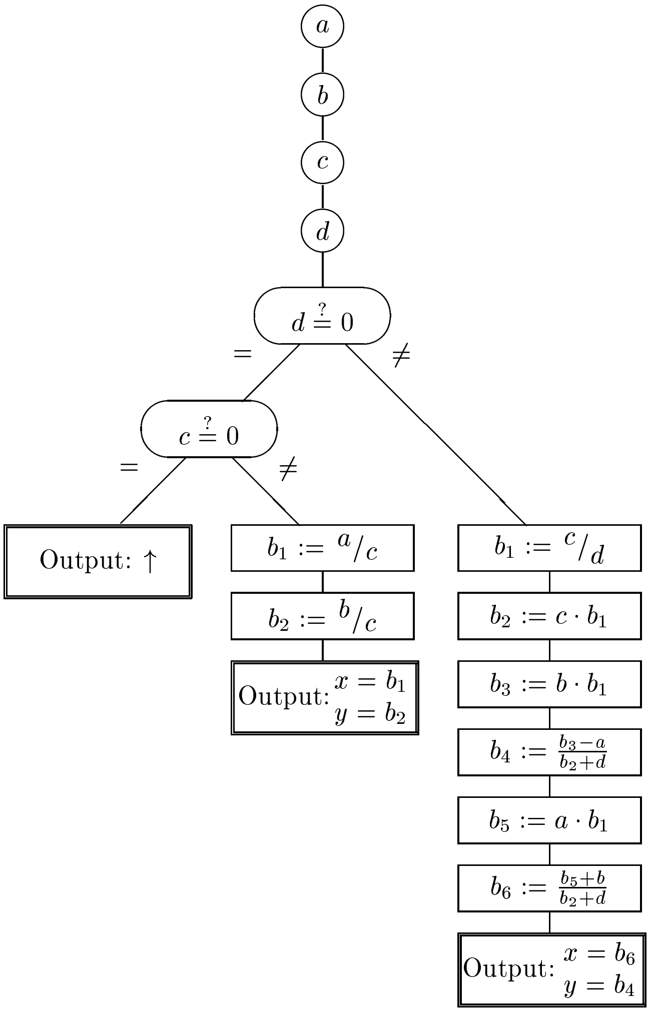



Abbildung 1: Ein Berechnungsbaum für die komplexe Division

schen straight-line Programmen und Berechnungsbäumen am Platz. Für fast alle Eingaben eines Berechnungsbaumes wird derselbe Pfad im Baum durchlaufen. Dieser "generische Pfad" ist im wesentlichen dadurch charakterisiert, daß immer der $\neq$-Verzweigung gefolgt wird. (In unserem Beispiel ist das der rechte Pfad.) Die arithmetischen Operationen entlang dieses Pfades definieren in natürlicher Weise ein straight-line Programm, dessen Kosten offensichtlich eine untere Schranke für den Aufwand des Berechnungsbaumes darstellen. Eine untere Schranke für die Komplexität der Menge der rationalen Funktionen $\left\{\frac{ac+bd}{c^2+d^2}, \frac{bc-ad}{c^2+d^2}\right\}$ in den Unbestimmten $a, b, c, d$ liefert deshalb auch eine untere Schranke, um die komplexe Division mittels Berechnungsbäumen zu berechnen! Dieser allgemeine Zusammenhang erklärt, warum das eingeschränkte Modell der straight-line Programme in der algebraischen Komplexitätstheorie so wichtig ist.

Wir betrachten nun das Problem, die Kettenbruchentwicklung des Quotienten $A_1/A_2$ zweier univariater Polynome über einem Körper $k$ zu berechnen. Diese Kettenbruchentwicklung ist durch die Folge der Quotienten $Q_1, \ldots, Q_{t-1}$ im Euklidschen Algorithmus

$$\begin{aligned}
A_1 &= Q_1 A_2 + A_3 \\
A_2 &= Q_2 A_3 + A_4 \\
&\vdots \\
A_{t-2} &= Q_{t-2} A_{t-1} + A_t, \\
A_{t-1} &= Q_{t-1} A_t,
\end{aligned}$$

gegeben, wobei das Polynom $A_t$ der größte gemeinsame Teiler von $A_1$ und $A_2$ ist. Weil die Anzahl $t-1$ der Quotienten $Q_i$ sowie deren Grade nicht von vornherein bekannt sind, kann diese Berechnung nicht von straight-line Programmen durchgeführt werden. Hingegen kann der Euklidsche Algorithmus mit algebraischen Berechnungsbäumen beschrieben werden. Wir nennen die Folge $(Q_1, \ldots, Q_{t-1}, A_t)$ die *Euklidsche Darstellung* von $A_1$ und $A_2$.

Was ist nun der nichtskalare Aufwand des Euklidschen Algorithmus? Sei im folgenden $n = \deg A_1 \geq \deg A_2$. In Abschnitt 3 erwähnten wir, daß Division mit Rest mit linearem Aufwand möglich ist. Da im schlimmsten Fall $n$ solche Divisionen ausgeführt werden müssen, hat der Euklidsche Algorithmus einen quadratischen Aufwand $O(n^2)$. Ist dies nun optimal?

Die Antwort ist nein! Unter Verwendung von Vorarbeiten von Lehmer [12] entwickelten Knuth [10] und Schönhage [19] Anfang der 70er Jahre ein trickreiches Verfahren, um den größten gemeinsamen Teiler zweier ganzer Zahlen zu berechnen. Deren Algorithmus läßt sich in naheliegender Weise auf Polynome übertragen [16]. Er beruht auf einer rekursiven Prozedur, die jeweils nur Anfangsstücke $Q_1, \ldots, Q_\rho$ der Euklidschen Darstellung berechnet, wobei die Summe der Grade der $Q_1, \ldots, Q_\rho$ etwa der halbe Grad $n/2$ von $A_1$ ist. Diese Prozedur liefert jeweils auch das Matrixprodukt

$$\begin{pmatrix} 0 & 1 \\ 1 & -Q_\rho \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -Q_1 \end{pmatrix}.$$

Darauf basierend kann man einen Algorithmus entwickeln, der die Euklidsche Darstellung, also insbesondere den größten gemeinsamen Teiler zweier gegebener Polynome, ungeheuer rasch berechnet, nämlich mit $O(n \log n)$ Multiplikationen oder Divisionen. Diese kompliziertere Aufgabe braucht also höchstens um einen Faktor $\log n$ mehr Operationen als die viel einfachere, zwei Polynome miteinander zu multiplizieren!

Strassen [21] führte eine Feinanalyse des Knuth-Schönhage Algorithmus durch. Um sein Ergebnis zu erläutern, setzen wir einige Notationen fest. Unter dem *Gradmuster* von $(Q_1, \ldots, Q_{t-1}, A_t)$ wollen wir die Folge $d = (d_1, \ldots, d_t)$ verstehen, wobei

$$d_1 := \deg Q_1, \ldots, d_{t-1} := \deg Q_{t-1}, d_t := \deg A_t.$$

Offenbar gilt $\sum d_i = n = \deg A_1$. Die *Entropie* $H(d)$ der Wahrscheinlichkeitsverteilung $\frac{1}{n} d$ ist definiert als

$$H(d) := -\sum_{d_i > 0} \tfrac{d_i}{n} \log \tfrac{d_i}{n}.$$

Strassens Ergebnis lautet nun so:

**Satz 4** *Der Knuth-Schönhage Algorithmus berechnet die Euklidsche Darstellung zweier gegebener Polynome $A_1, A_2$ mit dem nichtskalaren Aufwand*

$$O(n(1 + H(d))),$$

*wobei $d$ das Gradmuster der Euklidschen Darstellung von $A_1, A_2$ bezeichnet.*

Es ist einer der Höhepunkte der algebraischen Komplexitätstheorie, daß mittels der Gradschranke die Optimalität dieses Verfahrens bewiesen werden kann!

**Satz 5 (Strassen [21])** *Es sei $k$ ein algebraisch abgeschlossener Körper. Jeder algebraische Berechnungsbaum, der die Euklidsche Darstellung zweier Polynome über $k$ berechnet, benötigt für jedes Gradmuster $d$ wenigstens den nichtskalaren Aufwand $n(H(d)-2)$ für (Zariski) fast alle Polynome, deren Euklidsche Darstellung dieses Gradmuster hat.*

Man beachte, daß diese Aussage sogar zeigt, daß der Knuth-Schönhage Algorithmus optimal für *jedes* Gradmuster ist. Auch für nicht algebraisch abgeschlossene Körper $k$ sind untere Schranken bekannt, die allerdings etwas schwächer sind.

Der Beweis von Satz 5 verläuft etwa so: man führt das Problem auf die Untersuchung von straight-line Programmen zurück und verwendet dort die Gradschranke. Weil sich der Grad des Graphen eines Morphismus beim Übergang zur Inversen nicht ändert, wird man auf die Aufgabe geführt, den Grad des Graphen der Abbildung

$$\psi : (Q_1, \ldots, Q_{t-1}, A_t) \mapsto (A_1, A_2)$$

nach unten abzuschätzen. (Dies ist gerade die Inverse der zu berechnenden Abbildung, eingeschränkt auf die Menge der Polynompaare $(A_1, A_2)$, deren Euklidsche Darstellung das Gradmuster $d$ aufweist.)

Die polynomiale Abbildung $\psi$ enthält als homogenen Bestandteil vom höchsten Grad $t$ die Polynommultiplikation

$$\psi_0 : (Q_1, \ldots, Q_{t-1}, A_t) \mapsto Q_1 \cdots Q_{t-1} A_t.$$

Indem man $\psi_0$ als eine Deformation von $\psi$ interpretiert, kann man beweisen, daß der Grad des Graphen von $\psi$ nicht kleiner als der Grad des Graphen von $\psi_0$ ist. Der letztere kann aber leicht durch $n(H(d)-2)$ abgeschätzt werden.

Man vermutet, daß der Knuth-Schönhage Algorithmus auch optimal für die Berechnung des größten gemeinsamen Teilers ist. Diese Aussage liegt aber außerhalb der Reichweite der Gradschranke, welche nur eine lineare untere Schranke für dieses Problem liefert.

# 5 Berechnungsbäume und Betti-Zahlen

Im Unterschied zu vorher untersuchen wir jetzt algebraische Berechnungsbäume über dem Körper der reellen Zahlen $\mathbb{R}$, die auch gemäß kleiner-gleich Tests verzweigen dürfen.

**Beispiel.** Beginnen wir mit dem folgenden einfachen Problem: Gegeben seien $n$ reelle Zahlen, zu entscheiden ist, ob diese paarweise verschieden sind. Ein Informatiker würde diese Aufgabe wohl mit einem der gängigen Sortieralgorithmen lösen, welche nur Vergleiche $x_i < x_j$ und keine arithmetischen Operationen erfordern. Dies ist mit Größenordnung $n \log n$ Vergleichen möglich. Ein anderes Verfahren besteht darin, die Diskriminante $\prod_{i<j}(x_i - x_j)^2$ zu berechnen, was wieder mit $O(n \log n)$ nichtskalaren Operationen möglich ist, und das Resultat auf Null zu testen. Dies sind zwei sehr unterschiedliche Lösungen des Problems: bei der einen wird nur verglichen, bei der anderen wird bis auf einen Test nur gerechnet. Kann man mit weniger als Aufwand $n \log n$ auskommen?

Von einem abstrakten Standpunkt aus studieren wir folgende algorithmische Zugehörigkeitsprobleme. Es sei $W$ eine feste Teilmenge von $\mathbb{R}^n$. Zu einer Eingabe $x \in \mathbb{R}^n$ ist zu entscheiden, ob diese in der Menge $W$ liegt. Als Algorithmen seien beliebige algebraische Berechnungsbäume über $\mathbb{R}$ zugelassen. Es ist leicht zu sehen, daß diese Fragestellungen nur sinnvoll für Mengen $W$ ist, die durch Gleichungen und Ungleichungen reeller Polynome definiert werden können. Solche Mengen nennt man *semialgebraisch.*

Die Leitvorstellung ist, daß das Zugehörigkeitsproblem für Mengen mit komplizierter Topologie schwierig sein sollte. Für das einfachste numerische Maß der topologischen Kompliziertheit, nämlich die Anzahl Zusammenhangskomponenten, wurde dies durch das folgende Resultat von Ben-Or bestätigt.

**Satz 6 (Ben-Or [1])** *Es sei $W$ eine semialgebraische Teilmenge von $\mathbb{R}^n$ mit $b_0(W)$ Zusammenhangskomponenten. Dann benötigt jeder algebraische Berechnungsbaum, der das Zugehörigkeitsproblem für $W$ löst, mindestens*

$$(\log 6)^{-1}\left(\log b_0(W) - n \log 3\right)$$

*Multiplikationen, Divisionen und Vergleiche.*

In unserem Beispiel hat die Menge $W$ genau $n!$ Zusammenhangskomponenten: für jede Permutation $\pi$ beschreibt die Bedingung

$$x_{\pi(1)} < x_{\pi(2)} < \ldots < x_{\pi(n)}$$

eine solche. Deshalb erhalten wir eine untere Schranke der Größenordnung $n \log n$. Natürlich liefert Satz 6 nur dann gute Ergebnisse, wenn die Menge $W$ sehr viele Zusammenhangskomponenten hat.

Man kann das Resultat von Ben-Or als ein reelles Pendant zur Gradschranke interpretieren. Ähnlich wie die Gradschranke auf der Bézoutschen Ungleichung beruht, verwendet der Beweis von Satz 6 ein tieferliegendes Resultat von Milnor [14] und Thom [23], welches die Summe der Bettizahlen von reellen algebraischen Varietäten durch den Grad und die Anzahl Variablen der definierenden Polynome abschätzt. Für den Beweis des obigen Satzes genügt allerdings die Abschätzung der nullten Bettizahl, welche ja gerade die Anzahl Zusammenhangskomponenten beschreibt.

Die Schranke von Ben-Or findet z.B. Anwendungen in der algorithmischen Geometrie. In Kombination mit einer Konstruktion von Mayr und Meyer [13] kann man damit auch eine exponentielle untere Schranke für das Zugehörigkeitsproblem zu Polynomidealen bekommen [6].

In letzter Zeit gelang Björner, Lovász und Yao in einer Reihe von Arbeiten [3, 24, 2, 25] der Nachweis, daß die oben erwähnte Leitvorstellung allgemeiner gültig ist, indem sie Satz 6 auf höhere Bettizahlen verallgemeinerten.

Mit diesem neueren Resultat wollen wir unsere Einführung in die algebraische Komplexitätstheorie abschließen. Wir hoffen, daß wir Ihnen einen kleinen Einblick in die Thematik und Methoden unseres Gebietes geben konnten, sowie Appetit gemacht haben auf mehr! Es werden ja noch zwei Themen ausführlicher vorgestellt: *Schnelle Matrixmultiplikation und Kombinatorik* sowie *Zur Berechnungskomplexität von Permanenten.*

# Literatur

[1] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th*

*ACM STOC, Boston*, pages 80–86, 1983.

[2] A. Björner and L. Lovász. Linear decision trees, subspace arrangements and Möbius functions. *J. Amer. Math. Soc.*, 7(3):677–706, 1994.

[3] A. Björner, L. Lovász, and A.C. Yao. Linear decision trees: volume estimates and topological bounds. In *Proc. 24th ACM STOC*, pages 171–177, 1992.

[4] A. Borodin and R. Moenck. Fast modular transforms. *J. Comp. Syst. Sci.*, 8:366–386, 1974.

[5] A. Borodin and I. Munro. *The Computational Complexity of Algebraic and Numeric Problems.* American Elsevier, 1975.

[6] P. Bürgisser. On the parallel complexity of the polynomial ideal membership problem. Preprint, University of Zürich, 1996.

[7] P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*, 655 pp. Springer Verlag, 1996.

[8] C.M. Fiduccia. Polynomial evaluation via the division algorithm: the fast Fourier transform revisited. In *Proc. 4th ACM STOC*, pages 88–93, 1972.

[9] J. von zur Gathen. Algebraic complexity theory. *Ann. Review of Comp. Sci.*, 3:317–347, 1988.

[10] D.E. Knuth. The analysis of algorithms. In *Actes du congrès international des Mathématiciens, Nice*, volume 3, pages 269–274, 1970.

[11] J. van Leeuwen (ed.). *Handbook of Theoretical Computer Science*, volume A. Elsevier Science Publishers B. V., 1990.

[12] D.H. Lehmer. Euclid's algorithm for large numbers. *Amer. Math. Monthly*, 45:227–233, 1938.

[13] E.W. Mayr and A.R. Meyer. The complexity of the word problem for commutative semigroups and polynomial ideals. *Adv. Math.*, 46:305–329, 1982.

[14] J. Milnor. On the Betti numbers of real varieties. In *Proc. AMS*, volume 15, pages 275–280, 1964.

[15] R. Moenck and A. Borodin. Fast modular transforms via division. In *Proc. 13th Annual Symposium on Switching & Automata Theory*, pages 90–96, 1972.

[16] R.T. Moenck. Fast computation of GCDs. In *Proc. 5th ACM STOC*, pages 142–151, 1973.

[17] A.M. Ostrowski. On two problems in abstract algebra connected with Horner's rule. In *Studies in Mathematics and Mechanics presented to Richard von Mises*, pages 40–48. Academic Press, 1954.

[18] V.Ya. Pan. Methods for computing values of polynomials. *Russ. Math. Surv.*, 21:105–136, 1966.

[19] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Act. Inf.*, 1:139–144, 1971.

[20] V. Strassen. Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten. *Num. Math.*, 20:238–251, 1973.

[21] V. Strassen. The computational complexity of continued fractions. *SIAM J. Comp.*, 12:1–27, 1983.

[22] V. Strassen. Algebraic complexity theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 11, pages 634–672. Elsevier Science Publishers B. V., Amsterdam, 1990.

[23] R. Thom. Sur l'homologie des variétés algébriques réelles. In S.S. Cairns, editor, *Differential and Combinatorial Topology*, pages 255–265. Princeton Univ. Press, 1965.

[24] A.C. Yao. Algebraic decision trees and Euler characteristic. In *Proc. 33rd FOCS*, 1992.

[25] A.C. Yao. Decision tree complexity and Betti numbers. In *Proc. 26th ACM STOC*, 1994.