48. Workshop über Komplexitätstheorie, Datenstrukturen und Effiziente Algorithmen

Matthias Galota, Henning Schnoor, Heribert Vollmer

Fachgebiet Theoretische Informatik
Institut für Informationssysteme
Universität Hannover

48. Workshop über Komplexitätstheorie, Datenstrukturen und Effiziente Algorithmen

Hannover, 24. Juni 2003
Senatssaal, Hauptgebäude Universität Hannover

Programm

10:00 Begrüßung und Frühstück
10:30 *Matthias Homeister* (U Göttingen): Untere Schranken für Parity Branching Programme
11:00 *Bodo Manthey* (U Lübeck): Über die Berechnung des Hamming-Abstands
11:30 *Frank Balbach* (U Lübeck): Unterschiedliche Inferenztypen im selben Hypothesenraum

12:00 Mittagspause

13:30 *Till Tantau* (U Berlin): Logspace–Optimierungsprobleme und ihre Approximierbarkeit
14:00 *Birgit Schelm* (U Berlin): Strukturelle Eigenschaften von Average Case Approximationsklassen
14:30 *Sven Kosub* (U München): Boolean NP–Partitions and Projective Closuse

15:00 Kaffeepause

15:30 *Jan Arpe* (U Lübeck): One–Way–Kommunikation symmetrischer Funktionen: Vom Zwei–Spieler– zum Mehr–Spieler–Fall
16:00 *Elmar Böhler und Klaus Wagner* (U Würzburg): Polynomialzeithüllen
16:30 *Arfst Nickelsen, Till Tantau und Lorenz Weizsäcker* (U Berlin): Aggregate mit Komponentengröße 1 charakterisieren PSPACE

17:00 Ende des Workshops

Untere Schranken für Parity Branching Programme

Matthias Homeister

Institut für Numerische und Angewandte Mathematik, Georg–August–Universität Göttingen
Lotzestr. 16–18, 37083 Göttingen, Germany
homeiste@math.uni-goettingen.de

Branching Programme (BPs) sind ein Modell für logarithmisch platzbeschränkte Berechnungen und bestimmte Varianten können als Datenstruktur für Boolesche Funktionen genutzt werden (dann bezeichnet man BPs als BDDs - Binary Decision Diagrams). Ein BP über den Variablen $\{x_1, \ldots, x_n\}$ ist ein gerichteter azyklischer Graph mit einer unbeschrifteten Quelle und zwei Senken, die mit den Booleschen Konstanten $0, 1$ beschriftet sind. Alle inneren Knoten eines BPs sind mit einer Variablen x_i beschriftet und haben bei deterministischen PBs genau zwei ausgehende Kanten, wobei die eine mit 0 und die andere mit 1 beschriftet ist. Bei nichtdeterministischen Varianten erlaubt man jedem Knoten, beliebig viele Nachfolger zu haben. Eine Belegung der Variablen führt nun sehr natürlich zu einem Berechnungspfad, wobei so ein Pfad akzeptierend ist, wenn er in die 1-Senke führt.

Da für uneingeschränkte BPs bisher keine superpolynomialen unteren Schranken bewiesen werden konnten und solche auch nicht als Datenstruktur verwendbar sind (z.B. ist der Erfüllbarkeitstest NP-vollständig), betrachtet man eingeschränkte Varianten. Intensiv untersucht worden ist die folgende. Ein Branching Programm heißt *read–once* Branching Programm (BP1), wenn auf jedem Pfad von der Wurzel zu einer Senke jede Variable höchstens einmal getestet wird. BP1s sind sehr intensiv untersucht worden und untere Schranken wurden für deterministische BP1s, nichtdeterministische BP1s und sogar für nichtdeterministische read–k–times BPs bewiesen. Superpolynomiale untere Schranken für BP1s mit Parity-Akzeptierungsmodus ($\oplus$BP1s: akzeptiere eine Eingabe, wenn die Zahl der akzeptierenden Pfade ungerade ist) zu beweisen, ist noch immer ein offenes Problem.

Dieser Vortrag soll einen Überblick über neuere Ergebnisse auf diesem Gebiet geben. Und zwar werden folgende Varianten von Read–once Parity BPs betrachtet, die alle über den Begriff der Graphsteuerung definiert sind. Sei G ein deterministisches BP1. Ein BP1 $\mathcal{B}$ heißt durch die Graphsteuerung G gesteuert, wenn für jede Belegung a, auf den Pfaden in $\mathcal{B}$ die Variablen in derselben Reihenfolge getestet werden, wie auf dem Pfad in G.

- *Wohlstrukturierte graphgesteuerte $\oplus$BP1s*. Diese sind als Datenstruktur fürBoolesche Funktionen verwendbar. Die Darstellungskraft ist höher als die von $\oplus$OBDDs und deterministischen BP1s.

- *Allgemeine graphgesteuerte $\oplus$BP1s*. Man kann zeigen, dass sich $\oplus$BP1s,für die eine Graphsteuerung existiert, sehr natürlich charakterisieren lassen. Weiterhin kennt man einige untere Schranken.

- *Summen von graphgesteuerten $\oplus$BP1s*. Bestimmte lineare Codes lassen sich nur mit exponentiell großen Instanzen darstellen. Diese unteren Schranken sind die allgemeinsten, die man bisher für $\oplus$BPs kennt.

Über die Berechnung des Hamming-Abstands

Bodo Manthey

Institut für Theoretische Informatik
Universität zu Lübeck
Wallstraße 40, 23560 Lübeck
manthey@tcs.uni-luebeck.de

Der Hamming-Abstand $h(x, y)$ zweier Strings x und y gleicher Länge ist die Anzahl der Positionen, an denen sich x und y unterscheiden. Der Editierabstand $d(x, y)$ von x und y ist die minimale Anzahl an Zeichen, die gelöscht, eingefügt oder ersetzt werden müssen, um von x zu y zu gelangen (x und y müssen nicht unbedingt gleich lang sein). Der Hamming-Abstand $h(L, x)$ eines Strings $x \in \Sigma^\star$ zu einer Sprache $L \subseteq \Sigma^\star$ ist der minimale Hamming-Abstand von x zu einem Wort aus L: $h(L, x) = \min_{y \in L \cap \Sigma^{|x|}} h(y, x)$. Analog ist der Editierabstand $d(L, x)$ von x zu L definiert: $d(L, x) = \min_{y \in L} d(y, x)$.

Wir zeigen, dass die Berechnung des Hamming- und des Editierabstands zu Sprachen im Allgemeinen schwer ist.

Zunächst präsentieren wir eine Sprache $L \in \mathsf{AC}^0$, zu der der Hamming-Abstand schwer zu approximieren ist; der Hamming-Abstand zu L kann, unter der Annahme $\mathsf{NP} \neq \mathsf{P}$, nicht mit Faktor $O\bigl(n^{\frac{1}{3}-\epsilon}\bigr)$ approximiert werden (für beliebiges $\epsilon > 0$). Das Gleiche gilt für die Berechnung des Editierabstands zu L.

Des Weiteren zeigen wir, dass das Berechnen des Hamming-Abstands auch im Sinne der parametrisierten Komplexität schwer ist: Für jedes $t \in \mathbb{N}$ gibt es eine Sprache $L_t \in \mathsf{AC}^0$, so dass das Problem, den Hamming-Abstand zu L_t zu berechnen, $\mathsf{W}[t]$-hart ist. Außerdem gibt es eine Sprache $L_\mathsf{P} \in \mathsf{P}$, so dass das Berechnen des Hamming-Abstands zu L_P $\mathsf{W}[\mathsf{P}]$-hart ist.

Abschließend reduzieren wir das Problem der Hamming-Abstands-Berechnung auf das Problem der Editierabstands-Berechnung und umgekehrt. Beide Probleme sind also in gewisser Weise gleich schwer zu approximieren.

Der Vortrag basiert auf einer Arbeit zusammen mit Rüdiger Reischuk.

Unterschiedliche Inferenztypen im selben Hypothesenraum

Frank Balbach

Universität zu Lübeck

Der Induktiven Inferenz rekursiver Funktionen liegt ein Szenario mit drei wesentlichen Bestandteilen zu Grunde: (1) Eine Klasse U rekursiver Funktionen, (2) ein Hypothesenraum in Form einer Nummerierung ψ rekursiver Funktionen, sowie (3) einer rekursiven Funktion S (die sog. Strategie).

Unter „Lernen“ versteht man nun Folgendes. Die Strategie S erhält schrittweise die Folge der Funktionswerte einer Funktion $f \in U$ als Eingabe. Die Ausgabe von S besteht in jedem Schritt aus einer natürlichen Zahl („Hypothese“). Die Funktion f gilt als gelernt, wenn die Folge der ausgegebenen Hypothesen konvergiert und der Grenzwert eine Nummer von f in der Nummerierung ψ darstellt. Die Klasse U gilt als gelernt, wenn alle $f \in U$ von derselben Strategie erlernt werden.

Zu diesem Grund-Inferenztyp („LIM“) gibt es zahlreiche Variationen. Beispielsweise wird beim Inferenztyp CONS gefordert, dass alle ausgegebenen Hypothesen solche Funktionen beschreiben, die mit den der Strategie eingegebenen Funktionswerten konsistent sind; bei TOTAL müssen die durch die Hypothesen beschriebenen Funktionen total sein; bei CP[1] müssen die Hypothesenfunktionen aus der Klasse U stammen. TOTAL und CP lassen sich jeweils mit CONS kombinieren, woraus CONS-TOTAL und CONS-CP entstehen.

Ist nun U bzgl. ψ gemäß eines Inferenztyps I erlernbar, so stellt sich die Frage, ob U bzgl. ψ auch gemäß eines anderen Inferenztyps J erlernbar ist. Insbesondere wäre es schön, wenn ein solcher Übergang von I nach J unabhängig vom Hypothesenraum ψ möglich wäre.

Betrachtet man für I und J die obigen Inferenztypen, so kommt man u. a. zu folgenden Ergebnissen:

- Der Übergang von CONS nach TOTAL ist *nur für endliche* Klassen möglich,
- der Übergang von CP nach CONS-CP ist *für alle* hier sinnvollen (d. h. CONS-CP-erlernbaren) Klassen möglich,
- der Übergang von LIM nach CONS ist *für bestimmte* Klassen möglich, für andere jedoch nicht.

Es treten also recht unterschiedliche Phänomene auf, je nach dem, welche Inferenztypen man betrachtet.

[1] CP=Class Preserving

Logspace-Optimierungsprobleme und ihre Approximierbarkeit

Till Tantau

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Franklinstraße 28/29, 10587 Berlin, Germany
Fax +49-30-314-73500
tantau@cs.tu-berlin.de

Im Vortrag werden Logspace-Optimierungsprobleme vorgestellt. Sie sind analog zu den wohl untersuchten Polynomialzeit-Optimierungsproblemen definiert. Genau wie im Polynomialzeitfall können Logspace-Optimierungsprobleme unterschiedliche Approximationseigenschaften haben, obwohl die zugrundeliegenden Entscheidungsprobleme die gleiche Komplexität aufweisen. Beispiele sind das Kürzeste-Wege-Problem für gerichtete Graphen, für ungerichtete Graphen und für Tournaments. Wie im Polynomialzeitfall kann man auch Logspace-Approximationsklassen bilden. Unter der Annahme $\mathrm{L} \neq \mathrm{NL}$ gilt, dass sich bestimmte natürliche Logspace-Optimierungsprobleme nicht in logarithmischem Platz approximieren lassen; manche lassen sich mit konstanter Güte approximieren, besitzten aber kein Approximationsschema; und manche besitzen ein Approximationsschema, lassen sich aber nicht exakt lösen. Ein Beispiel für ein Problem des letzten Typs ist das Kürzeste-Wege-Problem für Tournaments.

Strukturelle Eigenschaften von Average Case Approximationsklassen

Birgit Schelm
Technische Universität Berlin
Fakultät IV –Elektrotechnik und Informatik
10587 Berlin
bts@cs.tu-berlin.de

Die Ausweitung der Average Case Komplexitästheorie auf Approximationsprobleme ermöglicht die Definition von Average Case Approximationsklassen, die die Klassifikation von Approximationsproblemen gemäß ihrer *durchschnittlichen* Approximierbarkeit ermöglicht. Dadurch lassen sich bekannte Ergebnisse über das durchschnittliche Verhalten von Approximationsproblemen – sowohl in Bezug auf ihr durchschnittliches Laufzeitverhalten, als auch in Bezug auf die im Durchschnitt erzielte Approximationsgüte – vereinheitlichen, obwohl diese Neuinterpretation bereits bekannter Ergebnisse nicht immer offensichtlich ist.

Die strukturellen Eigenschaften der Average Case Approximationsklassen sind jedoch auch für sich genommen interessant. In meinem Vortrag werde ich, angelehnt an entsprechende Definitionen aus dem Bereich der Worst Case Approximierbarkeit, entsprechende Average Case Approximationsklassen definieren und deren Eigenschaften vorstellen. Mit Hilfe einer die durchschnittliche Approximierbarkeit erhaltenden Reduktion lassen sich für die Klasse aller NP-Optimierungsprobleme mit P-berechenbaren Eingabeverteilungen vollständige Probleme angeben. Der Hauptaugenmerk meines Vortrags wird sich jedoch darauf richten, Nichtapproximierbarkeitsbeweise aus der Worst Case Approximierbarkeit auf die Average Case Approximierbarkeit zu übertragen, um zu zeigen, dass die Average Case Approximationsklassen eine Folge echter Inklusionen bilden, falls DistNP $\not\subseteq$ AvgP, d.h. falls NP-Probleme mit P-berechenbaren Eingabeverteilungen existieren, die sich auch im Durchschnitt nicht effizient lösen lassen.

Boolean NP-Partitions and Projective Closure

Sven Kosub

Institut für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching b. München, Germany
kosub@in.tum.de

When studying complexity classes of partitions we often face the situation that different partition classes have the same component classes. The projective closures are the largest classes among these with respect to set inclusion. In this paper we investigate projective closures of classes of boolean NP-partitions, i.e., partitions with components that have complexity upper-bounds in the boolean hierarchy over NP. We prove that the projective closures of these classes are represented by finite labeled posets. We give algorithms for calculating these posets and related problems. As a consequence we obtain representations of the set classes $\mathrm{NP}(m) \cap \mathrm{coNP}(m)$ by means of finite labeled posets.

One–Way–Kommunikation symmetrischer Funktionen: Vom Zwei-Spieler- zum Mehr-Spieler-Fall

Jan Arpe

Institut für Theoretische Informatik
Universität zu Lübeck
Wallstr. 40, 23560 Lübeck

arpe@tcs.uni-luebeck.de

Die folgenden beiden Problemstellungen der deterministischen One-Way-Kommunikation werden für symmetrische Boolesche Funktionen vom Zwei-Spieler- auf den Mehr-Spieler-Fall erweitert: das Design optimaler Protokolle und das Richtungs- bzw. Reihenfolgen-Problem.

Ferner wird das mit der One-Way-Kommunikation verwandte Modell der simultanen Kommunikation auf den Fall mehrerer Spieler verallgemeinert. In diesem Kommunikationsmodell schreiben die Spieler simultan jeweils eine Nachricht auf ein öffentliches Board. Das Ergebnis muss dann allein aus den Informationen auf dem Board zu berechnen sein.

Abschließend wird ein verallgemeinerter Protokolltyp vorgestellt (die Spieler schreiben nun nacheinander, so dass die Nachrichten von früher veröffentlichten Nachrichten abhängen dürfen), von dem sich jedoch herausstellt, dass er für symmetrische Funktionen keine effizienteren Protokolle zulässt als das eingeschränkte Modell.

Die Ergebnisse basieren auf einer gemeinsamen Arbeit mit Andreas Jakoby und Maciej Liśkiewicz (FCT 2003).

Polynomialzeithüllen

Elmar Böhler und Klaus Wagner

Theoretische Informatik
Universität Würzburg
Am Hubland
97074 Würzburg

In vielen Bereichen der Mathematik und der Informatik interessiert man sich für die Frage, ob ein gegebenes Objekt aus einer endlichen Grundmenge von Objekten, unterZuhilfenahme einiger weniger Operatoren, erzeugt werden kann. Beispielsweisebei der Konstruktion von Schaltkreisen, fragt man sich, ob ein Schaltkreismit bestimmten Eigenschaften, etwa so dass er eine gegebene boolesche Funktion berechnet, mit Hilfe gegebener Gatter "zusammengebaut" werden kann. Damit meint man hier, dass aus gegebenen Gattern und Schaltkreisen, mit Hilfe von wenigen, mathematisch beschreibbaren Operationen, neue Schaltkreise definiert werden. Im Falle der Schaltkreise handelt es sich bei diesen Operationen im wesentlichen um die Komposition, das "ineinander einsetzen", von Schaltkreisen. Wenn wir diese Operationen in einer Menge O festhalten und fragen, ob ein Schaltkreis C aus einer endlichen Menge von Gattern B erzeugt werden kann, heißt das, dass man wissen will ob C in der O-Hülle von B liegt. Ähnliche Fragestellungen im Zusammenhang mit Algebren ergeben sich, wenn man zu einer gegebenen Algebra $(A, \circ)$ (hier ist A eine Menge und $\circ$ eine Verknüpfung auf A) fragt, ob ein Element $z \in A$ sich durch iteratives Anwenden von $\circ$ auf eine endliche Grundmenge $B \subseteq A$ erzeugen läßt. Beispielsweise ist die Frage, ob sich eine Zahl n aus der Menge $\{2, \ldots, \lfloor \frac{n}{2} \rfloor\}$ durch Multiplikation erzeugen läßt, ist nichts anderes als das Primzahlproblem.

Wir betrachten Algebren mit Grundmenge Σ^* und einer zweistelligen Operation f, die allerdings deterministisch in Polynomialzeit berechenbar sein soll (kurz: $f \in \mathrm{FP}$) und fragen nach der Komplexität der entsprechenden Hüllen. Anders ausgedrückt, betrachten wir die Komplexität der folgenden Probleme:

PROBLEM: Das Generierungsproblem von f, $\mathrm{GEN}(f)$:
EINGABE: $x_1, \ldots, x_n, z$
FRAGE: Ist z in $[\{x_1, \ldots, x_n\}]_f$?

Hierbei ist $[B]_f$ die f-Hülle der Menge B. Man sieht leicht ein, dass es f aus FP gibt, so dass ihr Generierungsproblem unentscheidbar wird. Darum untersuchen wir Generierungsprobleme für Funktionen, die längenmonoton sind. Bei einer längenmonotonen Funktion muss der Funktionswert immer mindestens genauso lang sein wie jedes Argument. Für solche Funktionen liegt das Generierungsproblem immer in $\mathrm{EXPTIME}$. Wir bestimmen eine kommutative Verknüpfung f für die $\mathrm{GEN}(f)$ vollständig ist für $\mathrm{EXPTIME}$. Für alle längenmonotonen, assoziative Funktionen liegt $\mathrm{GEN}(f)$ in PSPACE, aber auch hiergibt es ein f so dass $\mathrm{GEN}(f)$ vollständig wird für PSPACE. Für längenomnotone, assoziative f die zusätzlich auch kommutativ sind, ist das Generierungsproblem in NP. Beispiele für solche Funktionen, für die das Generierungsproblem NP-vollständig wird, sind die Addition auf $\mathbb{N}$ oder die Multiplikation auf $\mathbb{N} \setminus \{0\}$.

Aggregate mit Komponentengröße 1 charakterisieren PSPACE

Arfst Nickelsen, Till Tantau, Lorenz Weizsäcker
{nicke, tantau}@cs.tu-berlin.de
weizack@informatik.hu-berlin.de

Bei Aggregaten wird wie bei Schaltkreisen die Rechenarbeit von Gattern geleistet. Während der Verknüpfungsgraph eines Schaltkreises azyklisch sein muss, kann er bei Aggregate Zyklen enthalten. In einem Schaltkreis wird bei gegebener Eingabe jedem Gatter nur einmal ein Wert zugeordnet. Dagegen ist die Berechnung eines Aggregates eine Folge von Konfigurationen: Jedem Gatter wird ein Wert in Abhängigkeit der Werte seiner Vorgänger im vorangegangen Schritt zugeordnet. Zum Einem kann dadurch ein Gatter in einer Berechnung mehrmals verwendet werden. Zum Anderen können so Bauteile des Aggregats, d.h. Gruppen von Gattern, Zwischenergebnisse weiterverarbeiten, die sie selbst berechnet haben.

Logspace-uniforme Aggregat-Familien polynomieller Größe können alle Sprachen aus PSPACE entscheiden. Wir untersuchen, inwieweit sich die Berechnungsstärke von Aggregaten verkleinert, wenn wir den freien Austausch von Zwischenergebnissen verhindern. Dazu beschränken wir die Größe der starken Zusammenhangskomponenten im Verknpfungsgraphen. Es zeigt sich, dass dies bei polynomieller Größe keine Einschränkung darstellt.

Jede Sprache aus PSPACE läßt sich durch eine logspace-uniforme, polynomiell große Familie von Aggregaten mit Komponentengröße 1 entscheiden. Dabei sind die einzigen Zyklen in den ansonsten azyklischen Verknpfunggraphen Schleifen an xor-Gattern.