Über die Komplexität der Multiplikation in eingeschränkten Branchingprogrammmodellen

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik
von

Philipp Wölfel

Dortmund
2003

Tag der mündlichen Prüfung:	15.12.2003
Dekan:	Prof. Dr. Bernhard Steffen
Gutachter:	Prof. Dr. Ingo Wegener, Prof. Dr. Berthold Vöcking

Bewertungsskala: ausgezeichnet, sehr gut, gut, genügend

Danksagung

Viele Personen haben auf die ein oder andere Weise Anteil am Gelingen dieser Arbeit. Besonders bedanken möchte ich mich bei Ingo Wegener, der mich bereits während des Studiums hervorragend betreut hat und mir seitdem ein wertvoller Ansprechpartner – nicht nur für wissenschaftliche Fragen – war. Die Mitarbeiter des Lehrstuhl 2 sind verantwortlich für eine motivierende und kreative Arbeitsatmosphäre; dafür bedanke ich mich bei ihnen ebenso wie für zahlreiche hilfreiche Gespräche. Einige wichtige Forschungsergebnisse über die Multiplikation sind in Zusammenarbeit mit Beate Bollig und Martin Sauerhoff entstanden. Beiden möchte ich für viele Diskussionen und die geduldige Beantwortung meiner Fragen bedanken. Bei meinem Ansprechpartner für alle Fragen zu universellem Hashing, Martin Dietzfelbinger, bedanke ich mich für die durch ihn vermittelten Einsichten.

Ganz besonderer Dank gebührt meinen Eltern, die mir während schwieriger Zeiten die größtmögliche Unterstützung haben zukommen lassen. Die entscheidende Inspiration für das Gelingen jeglicher Forschung sind aber die, die mir den Blick auf das Wesentliche öffnen. Dafür danke ich Miriam, Joshua und Isabell.

Inhaltsverzeichnis

KAPITEL 1

Motivation

Wie schwer ist es zu multiplizieren? Jeder hat in der Schule die Methode zum „schriftlichen“ Multiplizieren – die sog. „Schulmethode der Multiplikation“ – kennengelernt. Unsere Schulkenntnisse reichen also aus, einen effizienten Algorithmus zum Multiplizieren von langen Zahlen anzugeben. Um zwei Zahlen mit n Stellen zu multiplizieren, benötigt die Schulmethode $\Theta(n^2)$ arithmetische Operationen. Der Algorithmus – obwohl sehr einfach – ist bei weitem nicht optimal; Schaltkreise, die auf der Schulmethode beruhen, haben eine Größe von $O(n^2)$ und eine Tiefe von $O(n \log n)$. Das asymptotisch beste bekannte Verfahren (von Schönhage und Strassen, 1971) führt hingegen zu Schaltkreisen der Größe $O(n \log n \log\log n)$ und der Tiefe $O(\log n)$. Aber ist dieses Verfahren schon optimal?

In der Informatik besteht ein großes Interesse, die Komplexität von fundamentalen arithmetischen Funktionen zu bestimmen. Schließlich kommen sie in fast jedem Algorithmus auf die ein oder andere Weise vor und jeder Prozessor enthält Befehle zum Addieren, Subtrahieren, Multiplizieren und Dividieren. Während man bei oberen Schranken (z.B. für die Schaltkreisgröße und -tiefe) relativ gute Ergebnisse erzielen kann, ist man mit den heutigen Techniken noch nicht in der Lage z.B. im allgemeinen Schaltkreismodell vernünftige untere Schranken zu zeigen. So kennt man für keine explizit definierte boolesche Funktion in n Variablen – also auch nicht für Multiplizierer – eine bessere untere Schranke als $5n$ für die Schaltkreisgröße.

U.a. die Schwierigkeit große untere Schranken zu beweisen hat dazu geführt, dass andere Berechnungsmodelle für boolesche Funktionen herangezogen wurden. So haben sich *Branchingprogramme*, auch *Binary Decision Diagrams* (kurz BDDs) genannt, als eines der wichtigsten nichtuniformen Rechenmodelle etabliert. Das wichtigste Komplexitätsmaß für Branchingprogramme steht in starker Beziehung zur Schaltkreiskomplexität und zur Platzkomplexität nichtuniformer Turingmaschinen: Die Größe eines minimalen Branchingprogramms für eine boolesche Funktion liegt zwischen der Schaltkreisgröße und der Formelgröße der Funktion und ihr Logarithmus entspricht ungefähr dem Platzbedarf einer nichtuniformen Turingmaschine für die Funktion (vgl. Wegener, 1987, S. 414ff.).

Zwar konnten bis heute auch keine großen unteren Schranken für die Größe von allgemeinen Branchingprogrammen für explizit definierte Funktionen gezeigt werden, aber es gibt zahlreiche Möglichkeiten, die Mächtigkeit von Branchingprogrammen auf natürliche Weise einzuschränken und für viele eingeschränkte Rechenmodelle wurden bereits starke Techniken zum Beweis unterer Schranken entwickelt. Hinzu kommt, dass eingeschränkte Branchingprogrammmodelle, allen voran die sog. OBDDs (s. Abschnitt 2.1, Definition 2.2), in ihrer Funktion als Datenstruktur für boolesche Funktionen in zahlreichen Anwendungen Verwendung finden – zu den wichtigsten Beispielen gehören das Model-Checking und die Verifikation von Schaltkreisen.

Angesichts der Problematik, untere Schranken für allgemeine Rechenmodelle zu finden, konzentriert sich die Forschung bei eingeschränkten Branchingprogrammmodellen zunächst meist darauf, große (d.h. möglichst exponentielle) untere Schranken für beliebige explizit definierte Funktionen zu beweisen. Oftmals sind diese Funktionen so definiert, dass sie gut zu den zur Verfügung stehenden Beweistechniken passen. Daneben besteht aber auch ein großes Interesse daran, die Komplexität von in der Praxis bedeutsamen booleschen Funktionen kennen zu lernen. Solche Untersuchungen können helfen, die Techniken zum Beweis von unteren oder oberen Schranken zu verfeinern bzw. zu verallgemeinern. Andererseits erfordert der Beweis unterer Schranken für eine spezielle Funktion den Nachweis, dass diese gewisse „schwierige" Eigenschaften besitzt, die sie von „einfachen" Funktionen abgrenzt. So ein Nachweis liefert häufig interessante und manchmal überraschende Erkenntnisse über die betrachtete Funktion, die wiederum helfen können, die Komplexität dieser oder ähnlicher Funktionen in allgemeineren Rechenmodellen zu bestimmen. Für eingeschränkte Modelle wie OBDDs, die tatsächlich als Datenstruktur in der Praxis verwendet werden, liegt eine weitere Bedeutung solcher Untersuchungen auf der Hand: Sie können aufzeigen, für welche Probleme die Datenstruktur geeignet ist und wo ihre Grenzen liegen.

Zu den wichtigsten Funktionen gehören sicherlich die fundamentalen arithmetischen Operationen. Addition und Subtraktion sind zu „einfach", als dass weitere komplexitätstheoretische Untersuchungen in Branchingprogrammmodellen lohnenswert erscheinen – schließlich haben schon sehr eingeschränkte Branchingprogramme (z.B. OBDDs) für Addition oder Subtraktion nur lineare Größe. Anders ist die Situation bei der Multiplikation bzw. bei der booleschen Funktion, die ein (geeignet gewähltes) Ausgabebit des Produkts zweier Binärzahlen darstellt. Sie wird i.A. für „schwierig" gehalten, aber es gibt bisher nur für sehr eingeschränkte Branchingprogrammmodelle untere Schranken und selbst diese Schranken waren bisher noch nicht zufrieden stellend. So beweist zwar Bryant (1991) eine exponentielle untere Schranke für die Größe von OBDDs, die das „mittlere Ausgabebit" der Multiplikation berechnen; sie ist aber dennoch zu schwach, um zu zeigen, dass OBDDs zur Verifikation von Schaltkreisen für die 64-Bit-Multiplikation ungeeignet sind.

In der vorliegenden Arbeit wird daher die Komplexität der Multiplikation in eingeschränkten Branchingprogrammmodellen genauer untersucht. Dabei werden im ersten Teil bekannte Ergebnisse für eingeschränkte Branchingprogrammmodelle verbessert und später werden für etwas allgemeinere Modelle die ersten exponentiellen unteren Schranken bewiesen. Zum Beweis dieser neuen Ergebnisse werden ganz andere Eigenschaften der Multiplikation herangezogen, als dies beim Nachweis unterer Schranken in früheren Arbeiten (z.B. bei Bryant, 1991) der Fall war. Dabei handelt es sich u.a. um die bekannte Tatsache, dass sog. *universelle Hashklassen* (das sind Familien von Hashfunktionen mit starken Zufallseigenschaften) auf der Multiplikation beruhen.

KAPITEL 2

Einleitung

In der Komplexitätstheorie unterscheidet man zwischen *uniformen* Rechenmodellen, die Eingaben beliebiger Länge erhalten können, und *nichtuniformen* Modellen, bei denen die Eingabelänge fest vorgegeben ist. Uniforme Rechenmodelle – die wichtigsten Beispiele sind Turing- und Registermaschinen – modellieren daher die Softwareebene, während nichtuniforme Modelle der Hardwareebene entsprechen. Die meisten nichtuniformen Rechenmodelle berechnen eine *boolesche Funktion* $f : \{0, 1\}^n \to \{0, 1\}^m$. Die Klasse dieser Funktionen bezeichnen wir mit $B_{n,m}$ und wir kürzen $B_{n,1}$ mit B_n ab.

Wegen des starken Bezugs zur Hardware ist der *boolesche Schaltkreis* das bedeutendste nichtuniforme Rechenmodell. Bereits 1949 hat Shannon gezeigt, dass für fast jede boolesche Funktion $f \in B_n$ der kleinste Schaltkreis eine Größe von mindestens $2^n/n$ hat. Trotzdem konnten für die Schaltkreisgröße explizit definierter Funktionen bis heute nur lineare untere Schranken gezeigt werden; der Rekord liegt derzeit bei $5n - o(n)$ (Iwama und Morizumi, 2002). Selbst bei den eingeschränkteren *Formeln* – d.h. Schaltkreisen mit Fan-Out 1 – ist bis heute die beste untere Schranke für die Größe zur Darstellung einer explizit definierten Funktion nur $\Omega(n^2/\log n)$ (Nečiporuk, 1966).

Unter anderem mit dem Ziel, neue Techniken zum Beweis unterer Schranken zu entwickeln, aber auch als Datenstruktur für boolesche Funktionen wurden in den letzten Jahrzehnten Branchingprogramme intensiv untersucht. Deren Definition geht auf Lee (1959) und Akers (1978) zurück.

Definition 2.1 Ein *Branchingprogramm* (kurz BP) über einer Variablenmenge $X = \{x_1, \ldots, x_n\}$ ist ein gerichteter azyklischer Graph mit einer Wurzel w und zwei Senken, die mit den booleschen Konstanten 0 bzw. 1 markiert sind. Die inneren Knoten eines Branchingprogramms sind mit den Variablen aus X markiert und haben jeweils zwei ausgehende Kanten, die mit 0 bzw. 1 markiert sind. Zu einer Eingabe $a = a_1 \ldots a_n \in \{0, 1\}^n$ ist der *Berechnungspfad* derjenige Pfad von w zu einer Senke, der jeden mit einer Variablen $x_i \in X$ markierten Knoten über die mit a_i markierte Kante verlässt. Ein Berechnungspfad heißt *akzeptierend*, wenn er die 1-Senke erreicht, und die Eingabe a wird von dem Branchingprogramm *akzeptiert*, wenn der Berechnungspfad von a akzeptierend ist. Das Branchingprogramm stellt die Funktion $f \in B_n$ dar, für die $f^{-1}(1)$ genau die vom Branchingprogramm akzeptierten Eingaben enthält.

Die *Größe* eines Branchingprogramms G ist die Anzahl seiner Knoten und wird mit $|G|$ bezeichnet. Die *Branchingprogrammgröße* oder auch *Branchingprogrammkomplexität* einer Funktion $f \in B_n$, kurz BP(f), ist die Größe des minimalen Branchingprogramms, das f darstellt.

In dieser Arbeit werden vorwiegend Branchingprogramme für boolesche Funktionen mit einem Ausgabebit, d.h. Funktionen aus B_n betrachtet. Die obige Definition lässt sich aber auch leicht für Funktionen $f \in B_{n,m}$ mit $m \geq 2$ Ausgabebits erweitern, indem man mehrere Wurzeln zulässt. Die von der i-ten Wurzel ausgehenden Berechnungspfade definieren dann die Projektion von f auf das i-te Ausgabebit. Eine weitere Möglichkeit besteht darin, die Ausgabebits nicht an Senken, sondern an Kanten zu erzeugen. D.h., es gibt sog. Ausgabekanten, die jeweils mit einem Paar (i, c), $1 \leq i \leq m$ und $c \in \{0, 1\}$, markiert sind. Ein solches Branchingprogramm stellt dann eine Funktion $f \in B_{n,m}$ dar, wenn für jede Eingabe a mit $f(a) = y_1 \ldots y_m$ der Berechnungspfad von a genau die mit (i, y_i), $1 \leq i \leq m$, markierten Kanten enthält.

Eine wesentliche Bedeutung erlangen Branchingprogramme durch ihre Beziehung zu anderen wichtigen nichtuniformen Rechenmodellen. Sei $s(f)$ der minimale Platzbedarf einer nichtuniformen Turingmaschine, die die Funktion f berechnet. Cobham (1966) hat für jede Folge $f = (f_n)_{n \in \mathbb{N}}$ von Funktionen $f_n \in B_n$ gezeigt, dass

$$\log\big(\mathrm{BP}(f_n) + n\big) = \Theta\big(s(f_n) + \log n\big)$$

gilt. Also ist der Platzbedarf einer nichtuniformen Turingmaschine im Wesentlichen der Logarithmus der Branchingprogrammgröße. Auch der Zusammenhang zwischen Schaltkreisgröße bzw. Formelgröße und Branchingprogrammgröße ist recht gut eingegrenzt (s. z.B. Wegener, 1987; Sauerhoff, Wegener und Werchner, 1999; Giel, 2001): So gilt für $f = (f_n)_{n \in \mathbb{N}}$, $f_n \in B_n$, und alle $\epsilon > 0$

$$\frac{C(f_n)}{3} \leq \mathrm{BP}(f_n) = O\big(L(f_n)^{1+\epsilon}\big),$$

wobei $C(f_n)$ bzw. $L(f_n)$ die Größe des kleinsten Schaltkreis bzw. der kleinsten Formel für f_n angeben.

2.1 Eingeschränkte Branchingprogrammmodelle

Aufgrund der Beziehung zwischen Branchingprogrammkomplexität und Formelgröße folgt schon, dass es keine guten unteren Schranken für die Branchingprogrammgröße explizit definierter Funktionen gibt. Auch hier stammt das bis heute beste Ergebnis, eine Schranke von $\Omega(n^2/\log^2 n)$, von Nečiporuk (1966). Unter anderem um Techniken zum Beweis unterer Schranken zu entwickeln, wurde das Berechnungsmodell auf vielfache Weise eingeschränkt. Ein weiterer Grund, eingeschränkte Modelle zu betrachten, liegt darin, dass sie als Datenstruktur für boolesche Funktionen fungieren können. Allerdings macht eine Datenstruktur nur dann Sinn, wenn auf ihr geeignete Operationen effiziente ausgeführt werden können. Das allgemeine Branchingprogrammmodell ist aber zu mächtig, als dass es die effiziente Ausführung von Operationen erlaubt.

Einschränkungen können die Anzahl der erlaubten Lesezugriffe auf Variablen während der Berechnung eines Funktionswerts oder die Reihenfolge, in der auf Variablen zugegriffen werden darf, betreffen. Beim OBDD, dem für praktische Anwendungen vielleicht bedeutendsten Branchingrogrammmodell, ist sowohl die Reihenfolge als auch die Anzahl der Variablenzugriffe beschränkt.

Definition 2.2 Eine *Variablenordnung* über eine Menge X von n Variablen ist eine Bijektion $\pi : X \to \{1, \ldots, n\}$. Ein Branchingprogramm über X heißt *Ordered Binary Decision Diagram*, kurz OBDD, wenn es eine Variablenordnung π gibt, sodass für jede Kante von einem x_i-Knoten zu einem x_j-Knoten des OBDDs $\pi(x_i) < \pi(x_j)$ gilt. Ein OBDD mit Variablenordnung π heißt π-OBDD. Die π-OBDD-Größe einer Funktion $f \in B_n$ ist die Größe des minimalen π-OBDDs für f und die OBDD-Größe von f ist die Größe des minimalen OBDDs für f (mit einer beliebigen Variablenordnung).

Wenn man sich ein OBDD graphisch vorstellt, so kann man die Knoten in *Ebenen* einteilen, sodass alle Knoten einer Ebene mit der gleichen Variablen markiert sind und Kanten von einer Ebene nur in eine darunter liegende Ebene führen.

OBDDs wurden von Bryant (1985, 1986) als Datenstruktur für boolesche Funktionen entwickelt und haben seitdem in zahlreichen Gebieten wie Model Checking, Schaltkreisverifikation oder Logikminimierung Anwendung gefunden (s. z.B. die Monographien von Hachtel und Somenzi 1996, Minato 1996 oder Wegener 2000). Es ist entscheidend für den Einsatz von π-OBDDs als Datenstruktur, dass sich auf ihnen viele für die oben genannten Anwendungen wichtige Operationen effizient durchführen lassen, wenn die Variablenordnung π fest ist. Zu den wichtigsten Beispielen gehören die Syntheseoperation, die aus den π-OBDDs für zwei boolesche Funktionen $f, g \in B_n$ ein π-OBDD für die boolesche Funktion $f \odot g$ berechnet, wobei $\odot$ ein beliebiger binärer Operator ist, die Minimierung eines π-OBDDs, d.h. die Berechnung eines minimalen π-OBDDs für die Funktion $f \in B_n$ gegeben als π-OBDD, oder der Erfüllbarkeitstest, der zu einem OBDD entscheidet, ob es eine zu akzeptierende Eingabe gibt. Für die Anwendung bei der Verifikation von Schaltkreisen kann man z.B. ausnutzen, dass eine Folge dieser drei Operationen einen Äquivalenztest liefert, der entscheidet, ob zwei Schaltkreise die gleiche Funktion darstellen (s. auch Abschnitt 2.2).

Die Techniken zum Beweis unterer oder oberer Schranken für OBDDs sind ausgereift und für zahlreiche wichtige bzw. praxisrelevante Funktionen sind asymptotisch optimale Schranken bekannt. Hilfreich ist hierbei, dass für eine Funktion $f \in B_n$ und eine vorgegebene Variablenordnung π das π-OBDD minimaler Größe bis auf Isomorphie eindeutig bestimmt ist, und dass sich mithilfe des *Struktursatzes* von Sieling und Wegener (1993) die Anzahl von Knoten auf jeder Ebene des minimalen π-OBDDs anhand der Subfunktionsstruktur von f charakterisieren lässt.

Bei OBDDs ist durch die Variablenordnung einerseits die Reihenfolge, in der die Variablen auf einem Pfad von der Wurzel zu einer Senke vorkommen, festgelegt und andererseits darf jede Variable auf jedem Berechnungspfad nur einmal vorkommen. Eine natürliche Verallgemeinerung von OBDDs sind FBDDs, bei denen zwar die Einschränkung an die Anzahl von Variablentests aufrecht erhalten bleibt, nicht aber die Beschränkung an die Reihenfolge, in der die Variablen auf den Berechnungspfaden vorkommen dürfen.

Definition 2.3 Ein *Free Binary Decision Diagram*, kurz FBDD, ist ein Branchingprogramm, bei dem jeder Pfad von einer Wurzel zu einer Senke jede Variable höchstens einmal enthält. Wenn jeder solche Pfad jede Variable genau einmal enthält, heißt das FBDD *vollständig*.

FBDDs, auch als Read-Once-Branchingprogramme bezeichnet, wurden als erstes von Masek (1976) betrachtet. Zwar erlauben sie auch einige effiziente Operationen – z.B. gibt es einen co-RP Algorithmus, der testet ob zwei FBDDs die gleiche Funktion darstellen – aber für viele wichtige Operationen sind keine effizienten Algorithmen bekannt. Es wird vermutet, dass ein deterministischer Äquivalenztest nicht in polynomieller Zeit möglich ist (vgl.

Wegener, 2000, S. 144), und die Minimierung ist NP-hart – tatsächlich gibt es unter der Annahme NP≠P noch nicht einmal ein polynomielles Approximationsschema zur Minimierung eines FBDDs (vgl. Sieling, 2002).

Sieling und Wegener (1995) (vgl. auch Gergov und Meinel, 1994) haben deshalb sog. *graphgesteuerte* FBDDs definiert, bei denen es ein zusätzliches vollständiges FBDD mit nur einer Senke gibt, das *Graphordnung* genannt wird. Dieses bestimmt, in welcher Reihenfolge auf einem Berechnungspfad die Variablen getestet werden dürfen. D.h., wenn in dem graphgesteuerten FBDD für eine Eingabe a auf dem Berechnungspfad von a eine Variable x_i vor einer Variablen x_j erreicht wird, so muss x_i auch auf dem Berechnungspfad von a in der Graphordnung vor x_j erreicht werden. Z.B. kann man ein OBDD auch als graphgesteuertes FBDD betrachten, bei dem die Graphordnung eine Liste ist. Wenn man nun eine polynomiell große Graphordnung G festlegt, so sind Operationen wie Synthese, Minimierung oder Äquivalenztest auch in polynomieller Zeit für FBDDs mit Graphordnung G möglich. Da die Operationen teilweise jedoch schwieriger zu realisieren sind als bei π-OBDDs, haben graphgesteuerte FBDDs in der Praxis keine so große Verbreitung wie π-OBDDs gefunden.

Exponentielle untere Schranken für explizit definierte Funktionen für allgemeine FBDDs sind bereits aus den 80er Jahren bekannt (Žák, 1984; Wegener, 1988) und die beste bekannte untere Schranke für eine Funktion aus P hat mit $2^{n-O(\log^2 n)}$ eine fast optimale Größenordnung (Andreev, Baskakov, Clementi und Rolim, 1999). Auch für Funktionen von einfacher kombinatorischer Struktur wurden exponentielle untere Schranken gezeigt, z.B. für Funktionen, die in disjunktiver Form in polynomieller Größe dargestellt werden können und kurze Primimplikanten haben (Bollig und Wegener, 1998).

Der nächste Schritt hin zu allgemeineren Branchingprogrammen besteht darin, mehrere Variablentests auf den einzelnen Pfaden zu erlauben. Während bei FBDDs jeder graphtheoretische Pfad auch Berechnungspfad für eine Eingabe ist, ist dies bei Branchingprogrammen, bei denen auf einem Pfad mehrmals die gleiche Variable vorkommen darf, im Allgemeinen nicht der Fall. Deshalb muss man zwischen *syntaktischen* Einschränkungen an Eigenschaften der graphtheoretischen Pfade und *semantischen* Einschränkungen an die Berechnungspfade unterscheiden.

Definition 2.4

(a) Ein Branchingprogramm heißt *(syntaktisches)* k-BP, wenn jeder Pfad jede Variable höchstens k-mal enthält. Wenn diese Einschränkung nur für die Berechnungspfade gilt, so heißt das Branchingprogramm *semantisches* k-BP.

(b) Ein Branchingprogramm heißt *(syntaktisches)* $(1, +k)$-BP, wenn auf jedem Pfad höchstens k Variablen mehr als einmal vorkommen. Wenn diese Einschränkung nur für die Berechnungspfade gilt, so heißt das Branchingprogramm *semantisches* $(1, +k)$-BP.

So allgemeine Branchingprogrammmodelle wie k-BPs und $(1, +k)$-BPs haben als Datenstruktur für boolesche Funktionen kaum Bedeutung. Sie werden vor allem untersucht, um exponentielle untere Schranken für immer allgemeinere Modelle zu beweisen. So wäre eine untere Schranke für die Größe von $(1, +n)$-BPs für eine Funktion aus B_n natürlich auch eine untere Schranke für uneingeschränkte Branchingprogramme. Die ersten exponentiellen unteren Schranken für (syntaktische) k-BPs, $k \geq 2$, stammen von Borodin, Razborov und Smolensky (1993) sowie Okol'nishnikova (1993). Thathachar (1998) hat für alle $k \in \mathbb{N}$ gezeigt, dass die Klasse der Funktionen, die ein k-BP polynomieller Größe haben, eine echte

Teilmenge der Klasse der Funktionen ist, die ein $(k+1)$-BP polynomieller Größe haben. Also ist die Hierarchie der Klassen der durch k-BPs in polynomieller Größe darstellbaren Funktionen echt. Exponentielle untere Schranken für $(1, +k)$-BPs, $k \geq 1$, stammen z.B. von Sieling (1996) oder Savický und Žák (2000) und exponentielle untere Schranken für semantische $(1, +k)$-BPs finden sich z.B. bei Žák (1995), Savický und Žák (1997b) oder Jukna und Razborov (1998). Ähnlich wie bei k-BPs ist auch die Hierarchie der Klassen von Funktionen, die durch polynomiell große $(1, +k)$-BPs dargestellt werden können, zumindest für $k = O(n^{1/2}/\log n)$, echt (vgl. Savický und Žák, 2000). Ein ähnliches Hierarchieresultat gilt sogar für semantische $(1, +k)$-BPs (s. ebenda).

Erst vor ein paar Jahren wurde ein Durchbruch beim Beweis exponentieller unterer Schranken für ein noch allgemeineres Modell als das k-BP-Modell erzielt, bei dem nur die *Länge* – also die Anzahl von Knoten auf dem längsten Berechnungspfad – beschränkt ist (Ajtai, 1999; Beame, Saks, Sun und Vee, 2003). Man bezeichnet dabei die Länge des Branchingprogramms mit *Time* (kurz T) sowie den Logarithmus von dessen Größe als *Space* (kurz S) und versucht Time-Space-Tradeoffs zu beweisen, also untere Schranken für das Produkt $T \cdot S$. So beweisen z.B. Beame et al. einen Tradeoff von $T = \Omega\left(n\sqrt{\log(n/s)/\log\log(n/S)}\right)$ für eine explizit definierte Funktion $f \in B_n$. Dies impliziert, dass jedes Branchingprogramm für diese Funktion f mit $2^{O(n^{1-\epsilon})}$ Knoten, $\epsilon > 0$, eine Länge von $\Omega\left(n\sqrt{\log n/\log\log n}\right)$ hat.

Nichtdeterministische Branchingprogramme

Nichtdeterminismus gehört zu den wichtigsten Konzepten der theoretischen Informatik. In Analogie zu nichtdeterministischen Turingmaschinen, die die Möglichkeit haben, akzeptierende Rechenwege zu „raten“, wurde Nichtdeterminismus bei Branchingprogrammen durch Hinzufügen nichtdeterministischer Knoten so definiert, dass zu jeder Eingabe mehrere Berechnungspfade möglich sind. Nichtdeterministische Knoten sind unmarkierte Knoten mit Ausgangsgrad 2. Berechnungspfade sind wie in Definition 2.1 definiert, aber nun gibt es zu jeder Eingabe $a \in \{0,1\}^n$ mehrere Berechnungspfade, da so ein Berechnungspfad jeden nichtdeterministischen Knoten über eine beliebige ausgehende Kante verlassen kann. Die von dem nichtdeterministischen Branchingprogramm dargestellte Funktion $f \in B_n$ bildet genau die Eingaben auf 1 ab, für die es mindestens einen akzeptierenden Berechnungspfad gibt.

Neben diesem existenziellen Nichtdeterminismus, der aufgrund des Zusammenhangs zu nichtdeterministischen Turingmaschinen der natürlichste ist, wurden noch zahlreiche andere Formen des Nichtdeterminismus definiert (s. z.B. Meinel, 1990). Beispiele sind der AND-Nichtdeterminismus, bei dem genau die Eingaben akzeptiert werden, für die alle Rechenwege akzeptierend sind, oder der XOR-Nichtdeterminismus, bei dem nur Eingaben mit einer ungeraden Anzahl akzeptierender Rechenwege akzeptiert werden. Natürlich gibt es zu allen eingeschränkten Branchingprogrammmodellen aus Definition 2.4 auch entsprechende nichtdeterministische Varianten, die dadurch definiert sind, dass zusätzliche nichtdeterministische Knoten erlaubt sind.

Exponentielle untere Schranken sind häufig für nichtdeterministische Branchingprogrammmodelle wesentlich schwieriger zu beweisen als für deterministische Modelle. Zwar funktioniert die Technik von Borodin, Razborov und Smolensky (1993) zum Beweis unterer Schranken für deterministische (syntaktische) k-BPs auch für nichtdeterministische k-BPs, aber schon für nichtdeterministische semantische 2-BPs konnte bisher noch keine exponentielle untere Schranke für eine explizit definierte Funktion bewiesen werden.

2.2 Die Komplexität arithmetischer Funktionen in Branchingprogrammmodellen

Während man einerseits noch weit davon davon entfernt ist, exponentielle untere Schranken für explizit definierte Funktionen im allgemeinen Branchingprogrammmodell zu beweisen, sind andererseits für eingeschränktere Modelle schon teilweise recht gute Methoden zum Nachweis unterer Schranken entwickelt worden. Dies gilt insbesonders für Branchingprogrammtypen, die als Datenstruktur verwendet werden, allen voran für OBDDs. Gerade für OBDDs wurde auch die Komplexität vieler wichtiger und für die Praxis bedeutsamer boolescher Funktionen untersucht.

Die Bedeutung unterer bzw. oberer Schranken für die OBDD-Größe wichtiger Funktionen ist vor allem durch Anwendungen wie die Verifikation von Schaltkreisen motiviert. Ein typisches Problem, das mit OBDDs gelöst werden kann, ist Folgendes: Gegeben ist die Spezifikation einer Funktion $f \in B_n$, dargestellt z.B. durch ein OBDD oder durch einen Schaltkreis, dessen Korrektheit schon verifiziert wurde, sowie ein Entwurf eines neuen Schaltkreises, der die Funktion f darstellen soll. Man möchte nun herausfinden, ob der Schaltkreisentwurf tatsächlich korrekt ist, d.h. man muss beweisen, dass die von ihm berechnete Funktion g gleich der Funktion f ist. Dazu kann man nun mithilfe mehrerer Syntheseoperationen aus dem Schaltkreisentwurf ein π-OBDD für die Funktion g erzeugen und anschließend mit einer weiteren Syntheseoperation das π-OBDD für $f \oplus g$ berechnen, wobei $\oplus$ die XOR-Operation bezeichnet (falls f durch einen Schaltkreis gegeben ist, berechnet man zuvor auch das π-OBDD für f). Mithilfe des Erfüllbarkeitstests kann nun entschieden werden, ob $f \oplus g$ mindestens eine erfüllende Eingabe hat, was äquivalent zu $f \neq g$ ist.

Jede einzelne für so eine Verifikation notwendige OBDD-Operation benötigt im Worst-Case nur quadratische Zeit bzgl. des größten beteiligten OBDDs. Trotzdem kann die Gesamtlaufzeit dieser Methode sehr schlecht sein, wenn zu große OBDDs entstehen. Mittlerweile wurden zahlreiche Techniken und Heuristiken entwickelt, um das Entstehen von großen OBDDs während der Syntheseoperationen zu verhindern und einfache Schaltkreise wie Addierer können mit OBDDs effizient verifiziert werden (s. z.B. Minato, 1996). Solche Methoden versagen jedoch, wenn schon das π-OBDD für die Spezifikation zu groß ist. So konnte z.B. trotz zahlreicher Versuche erst vor wenigen Jahren für den gesamten 16-Bit Multiplikationsschaltkreis $c6288$, einer der bekanntesten ISCAS (International Symposium on Circuits and Systems) Benchmark Schaltkreise, ein OBDD berechnet werden (Yang, Chen, Bryant und O'Hallaron, 1998). Es sei angemerkt, dass das resultierende OBDD mithilfe mehrerer Wurzeln alle Ausgabebits des Multiplizierers darstellte. Es bestand aus mehr als 40 Millionen Knoten, das größte OBDD, das während der Syntheseoperationen entstand, hatte sogar mehr als 110 Millionen Knoten und der maximale Speicherplatzbedarf betrug 3803 Megabyte. Soweit dem Autor bekannt ist, konnte bis heute aus keinem Multiplikationsschaltkreis mit mehr als 16 Bit ein entsprechendes OBDD berechnet werden.

Natürlich bedeuten diese Erfahrungswerte noch nicht, dass es nicht auch kleine OBDDs für 16-Bit oder sogar 32-Bit Multiplizierer geben kann. Die Größe des minimalen OBDDs für eine Funktion hängt stark von der gewählten Variablenordnung ab – tatsächlich gibt es viele Funktionen mit polynomieller OBDD-Größe, die für die meisten Variablenordnungen π exponentielle π-OBDD-Größe haben (mehrere Beispiele finden sich in der Monographie von Wegener, 2000). So könnte es möglich sein, dass bei den Versuchen, OBDDs für Multiplizierer zu berechnen, immer nur die falschen Variablenordnungen gewählt bzw. heuristisch gefunden wurden.

Wenn wir aber beweisen können, dass es in Wirklichkeit gar keine kleinen OBDDs für Multiplizierer gibt, kann man auf weitere Versuche, OBDDs zur Verifikation einzusetzen, verzichten und sich auf die Suche nach alternativen Lösungen konzentrieren. Wenn man die Gründe versteht, aus denen so eine Datenstruktur für die Darstellung und Verifikation wichtiger Funktionen ungeeignet ist, so kann dies auch dabei helfen bessere Datenstrukturen und Methoden zu entwickeln. Es ist daher von großem Interesse, die OBDD-Größe von wichtigen Funktionen zu untersuchen, um entscheiden zu können, ob der Einsatz von OBDDs z.B. zur Verifikation entsprechender Schaltkreise sinnvoll ist.

Viele der allgemeineren Branchingprogrammmodelle spielen zwar als Datenstruktur für boolesche Funktionen keine wesentliche Rolle, dennoch sind auch hier komplexitätstheoretische Untersuchungen für wichtige Funktionen wie die Multiplikation von Bedeutung. Wenn ein neues Berechnungsmodell untersucht wird, werden häufig zunächst untere Schranken für beliebige, aber explizit definierte Funktionen bewiesen. Oft werden für die ersten Beweise dann Funktionen gerade so definiert, dass sie gut zur Beweistechnik passen. Auch wenn dieses Vorgehen wichtig ist, um erst einmal Beweistechniken zu entwickeln, darf man nicht vergessen, dass solche künstlich definierten Funktionen nicht an sich von komplexitätstheoretischem Interesse sind. Stattdessen muss am Ende immer das Ziel stehen, mit den gelernten Techniken untere Schranken für wichtige und bedeutende Funktionen zu beweisen, deren Komplexität uns wirklich interessiert. Wenn man eine bestimmte ausgewählte Funktion betrachtet, die ihre Bedeutung unabhängig vom betrachteten Berechnungsmodell erlangt, so wird meist der Nachweis unterer oder oberer Schranken schwieriger sein, als wenn man die Funktion im Hinblick auf bekannte Beweistechniken geeignet definieren kann. So muss man im Fall einer vorgegebenen Funktion entweder die Eigenschaften, die man zur Anwendung einer bekannten Beweistechnik benötigt, erst nachweisen oder es müssen neue Beweistechniken gefunden bzw. bekannte Beweistechniken verändert werden, sodass sie auf die entsprechenden Eigenschaften der Funktion anwendbar sind. Während ersteres dabei meist zu einem umfassenderen Verständnis der Funktion führt, kann letzteres allgemeinere Beweistechniken oder sogar ein besseres Verständnis des Berechnungsmodells bewirken. Abgesehen davon haben einige Funktionen, wie z.B. die fundamentalen arithmetischen Operationen, eine solch große Bedeutung (nicht nur für die Informatik), dass es eines der vorrangigen Ziele der Wissenschaft sein muss, möglichst viel über sie zu erfahren. Für Branchingprogramme heißt dies, dass die Komplexität solcher Funktionen für alle bedeutenden eingeschränkten Modelle möglichst gut bestimmt werden sollte.

Die Bedeutung der Multiplikation

Sicherlich gehören die vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division unter allen arithmetischen Funktionen zu den bedeutendsten. Um auf sie näher einzugehen, benutzen wir folgende Notation für die Interpretation von Bitstrings als nichtnegative ganze Zahlen.

Definition 2.5 Sei $\mathbb{Z}_k = \{0, \ldots, k-1\}$ und $\mathbb{B}_n$ die Menge der Binärdarstellungen von Zahlen aus $\mathbb{Z}_{2^n}$, d.h. die Menge der Strings $x = x_{n-1} \ldots x_0$ mit $x_i \in \{0, 1\}$. Für $x \in \mathbb{B}_n$ sei $|x| = \sum_{i=0}^{n-1} x_i 2^i$ die Interpretation von x als ganze Zahl in $\mathbb{Z}_{2^n}$. Umgekehrt benutzen wir die Notation $\langle z \rangle_\ell^k$, um zu einer Zahl $z \in \mathbb{Z}_{2^n}$ den zugehörigen binären Teilstring an den Bitpositionen $\ell, \ldots, k$ zu beschreiben, also für $x \in \mathbb{B}_n$ und $0 \leq \ell \leq k < n$ ist $\langle |x| \rangle_\ell^k = x_k \ldots x_\ell$.

Wir benutzen die Schreibweise $\mathbb{B}_n$ anstatt $\{0, 1\}^n$, um zu implizieren, dass die entsprechenden Bitstrings Binärzahlen darstellen. So gilt im Folgenden die Konvention, dass $x = x_{n-1} \ldots x_0$ ist, wenn wir x als Element aus $\mathbb{B}_n$ gewählt haben.

Addition und Subtraktion sind für die meisten Branchingprogrammmodelle einfach.

Definition 2.6 Die Funktion $\mathrm{ADD}_n : \mathbb{B}_n \times \mathbb{B}_n \to \mathbb{B}_{n+1}$ berechnet die Summe zweier ganzer Zahlen, d.h. die Abbildung $(x, y) \mapsto z$ für $z \in \mathbb{B}_{n+1}$ mit $|z| = |x| + |y|$. Die Funktion $\mathrm{ADD}_{i,n} \in \mathbb{B}_n \times \mathbb{B}_n \to \{0, 1\}$, $0 \leq i \leq n$, berechnet das i-te Ausgabebit der Addition, d.h. die Funktion $(x, y) \mapsto z_i$ für $z = \mathrm{ADD}_n(x, y)$.

Folgende obere und untere Schranke für die OBDD-Größe der Addition sind allgemein bekannt.

Theorem 2.7

(a) Sei τ die Variablenordnung über $Z = \{x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}\}$ mit

$$\big(\tau^{-1}(1), \ldots, \tau^{-1}(2n)\big) = (x_{n-1}, y_{n-1}, \ldots, x_0, y_0).$$

Dann gibt es für $n \geq 2$ ein τ-OBDD der Größe $9n - 5$, das die Funktion ADD_n an $n + 1$ Wurzeln darstellt.

(b) Sei π eine Variablenordnung über Z, $1 \leq k \leq 2n$ und I die Menge der Indizes $i \in \{0, \ldots, n-1\}$, für die $\pi(x_i) \leq k < \pi(y_i)$ oder $\pi(y_i) \leq k < \pi(x_i)$ gilt. Dann hat jedes π-OBDD für $\mathrm{ADD}_{n-1,n}$ mindestens $\Omega\big(2^{|I|}\big)$ Knoten.

Die Beweise findet man z.B. in der Monographie von Wegener (2000) (die untere Schranke wird dort nicht explizit so dargestellt, sie geht aber implizit aus dem Beweis von Theorem 5.3.3 hervor). Da auch die Subtraktion lineare OBDD-Größe hat, sind komplexitätstheoretische Untersuchungen von Addition und Subtraktion in allgemeineren Modellen uninteressant.

Das weiter oben genannte Beispiel aus der Schaltkreisverifikation ist ein Indiz dafür, dass die Multiplikation vermutlich keine so einfache Funktion ist.

Definition 2.8 Die Funktion $\mathrm{MUL}_n : \mathbb{B}_n \times \mathbb{B}_n \to \mathbb{B}_{2n}$ berechnet das Produkt zweier ganzer Zahlen, d.h. die Abbildung $(x, y) \mapsto z$ für $z \in \mathbb{B}_{2n}$ mit $|z| = |x| \cdot |y|$. Die Funktion $\mathrm{MUL}_{i,n} : \mathbb{B}_n \times \mathbb{B}_n \to \{0, 1\}$, $0 \leq i \leq 2n - 1$, berechnet das i-te Ausgabebit der Multiplikation, d.h. die Funktion $(x, y) \mapsto z_i$ für $z = \mathrm{MUL}_n(x, y)$.

Das *mittlere Ausgabebit* der Multiplikation, also die Funktion $\mathrm{MUL}_{n-1,n}$, ist für den Beweis unterer Schranken das interessanteste, denn mithilfe einfacher Reduktionen erhält man bei fast allen Branchingprogrammmodellen aus einer unteren Schranke für $\mathrm{MUL}_{n-1,n}$ auch eine untere Schranke für $\mathrm{MUL}_{i,n}$.

Um dies zu erläutern, beschreiben wir das Konzept der Read-Once-Projektionen, dass von Bollig und Wegener (1996) als Spezialfall polynomieller Projektionen definiert wurde.

Definition 2.9 Seien $f = (f_n)_{n\in\mathbb{N}}$ und $g = (g_n)_{n\in\mathbb{N}}$ Folgen boolescher Funktionen. Dann ist f eine *Read-Once Projektion* von g, i.Z. $f \leq_{\text{rop}} g$, wenn es eine polynomiell beschränkte Funktion p gibt, sodass für alle $n \in \mathbb{N}$

$$f_n(x_1, \ldots, x_n) = g_{p(n)}\big(y_1, \ldots, y_{p(n)}\big)$$

für $y_j \in \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n, 0, 1\}$ gilt, wobei $\left\{y_i, y_j\right\} \not\subseteq \left\{x_k, \overline{x_k}\right\}$ für alle $i \neq j$ gilt.

Um die Qualität einer Read-Once Projektion genauer anzugeben, schreiben wir auch $f_n \leq_{\text{rop}} g_{p(n)}$, wenn p die polynomiell beschränkte Funktion aus der Reduktion ist.

Man sieht nun leicht, dass bei fast allen eingeschränkten Branchingprogrammmodellen für die Darstellung einer Funktion f_n nicht mehr Knoten als für die Darstellung der Funktion $g_{p(n)}$ benötigt werden, wenn $f_n \leq_{\text{rop}} g_{p(n)}$ gilt. Dazu nutzt man aus, dass man bei einem Branchingprogramm G, das die Funktion $f_n \in B_n$ darstellt, eine beliebige Variable x_k durch eine Konstante $c \in \{0, 1\}$ ersetzen kann, indem man alle Kanten, die auf einen mit x_k markierten Knoten v zeigen, stattdessen auf den c-Nachfolger von v zeigen lässt. Anschließend kann man alle mit x_k markierten Knoten entfernen und erhält ein nicht größeres Branchingprogramm G' für die Subfunktion $f_{n|x_k=c}$. Außerdem kann man einen x_k-Knoten in einen $\overline{x_k}$-Knoten transformieren, indem man einfach die Markierungen der ausgehenden Kanten des Knotens vertauscht. Führt man diese Operationen auf eingeschränkten Branchingprogrammen aus, so erhält man keine größeren Branchingprogramme, die in den meisten Fällen den gleichen Einschränkungen unterliegen. Auf diese Weise ergibt sich folgende allgemein bekannte Aussage (vgl. Bollig und Wegener, 1996).

Bemerkung 2.10 *Gilt $f_n \leq_{\text{rop}} g_{p(n)}$, so ist die Größe eines minimalen (nichtdeterministischen)* OBDD*s,* FBDD*s, k-*BP*s oder $(1, +k)$-*BP*s für f_n nicht größer als die Größe eines genauso eingeschränkten Branchingprogramms für $g_{p(n)}$.*

Die folgende Idee zur Reduktion des mittleren Bits der Multiplikation auf ein beliebiges anderes Ausgabebit hat Bryant (1991) schon benutzt, um untere Schranken für die OBDD-Größe aller Ausgabebits der Multiplikation anzugeben.

Theorem 2.11 (Bryant, 1991) *Sei $0 \leq i \leq 2n-1$ und j das Minimum von $i+1$ und $2n-i-1$. Dann gilt $\mathrm{MUL}_{j-1,j} \leq_{\text{rop}} \mathrm{MUL}_{i,n}$.*

Beweis: Für $0 \leq i < n-1$ und $x, y \in \mathbb{B}_n$ gilt offenbar

$$\begin{aligned}\mathrm{MUL}_{i,n}(x_{n-1}\ldots x_0,\ y_{n-1}\ldots y_0) &= \mathrm{MUL}_{i,n}(0^{n-i-1}\,x_i\ldots x_0, 0^{n-i-1}\,y_i\ldots y_0)\\ &= \mathrm{MUL}_{i,i+1}(x_i\ldots x_0, y_i\ldots y_0).\end{aligned}$$

Dies beweist $\mathrm{MUL}_{i,i+1} \leq_{\text{rop}} \mathrm{MUL}_{i,n}$.

Sei nun $n-1 \leq i \leq 2n-1$ und $j = 2n-i-1$. Es gilt für alle $0 \leq k \leq n-1$

$$\left|x_{n-1}\ldots x_k 0^k\right| \cdot \left|y_{n-1}\ldots y_k 0^k\right| = 2^{2k} \cdot |x_{n-1}\ldots x_k| \cdot |y_{n-1}\ldots x_k|,$$

wobei 0^k eine Folge von k 0en bedeutet. Also ist

$$\begin{aligned}&\mathrm{MUL}_{i,n}\big(x_{n-1}\ldots x_{n-j}0^{n-j}, y_{n-1}\ldots y_{n-j}0^{n-j}\big)\\ &\qquad = \mathrm{MUL}_{i-2(n-j)),j}\big(x_{n-1}\ldots x_{n-j}, y_{n-1}\ldots y_{n-j}\big).\end{aligned}$$

Mit $i-2(n-j) = j-1$ folgt dann $\mathrm{MUL}_{j-1,j} \leq_{\text{rop}} \mathrm{MUL}_{i,n}$. ■

Aus der Arbeit von Wegener (1993) lässt sich folgern, dass die Division für die meisten Branchingprogrammmodelle nicht leichter als die Multiplikation ist. Das folgende Resultat wird auch in der Monographie von Wegener (2000, Theorem 4.6.2) beschrieben und bewiesen.

Theorem 2.12 (Wegener, 1993) *Sei* $\mathrm{DIV}_n \in B_{2n,n}$ *die Funktion, die die n signifikantesten Bits des Quotienten zweier als Binärdarstellung gegebener Zahlen berechnet. Es gilt* $\mathrm{MUL} \leq_{\mathrm{rop}} \mathrm{DIV}$.

Da ein Branchingprogramm für MUL_n als Subgraph natürlich auch ein Branchingprogramm für $\mathrm{MUL}_{n-1,n}$ enthält, liefern in den meisten eingeschränkten Branchingprogrammmodellen untere Schranken für das mittlere Bit der Multiplikation auch untere Schranken für die Division. Umgekehrt lassen sich aber untere Schranken für die Division nicht durch Read-Once Projektionen auf die Multiplikation übertragen (vgl. Bollig und Wegener, 1996). Für den Beweis unterer Schranken sind deshalb die Multiplikation und insbesondere die Funktion $\mathrm{MUL}_{n-1,n}$ am interessantesten.

Bekannte Schranken für die Komplexität der Multiplikation

Für die allgemeinsten Berechnungsmodelle wie Schaltkreise, sind recht gute obere Schranken für die Multiplikation bekannt. So ist es nicht überraschend, dass man auch im allgemeinen Branchingprogrammmodell leicht polynomielle obere Schranken erhält. Die Schulmethode der Multiplikation liefert ein Branchingprogramm der Größe $O(n^2 i)$ für die Funktion $\mathrm{MUL}_{i,n}$ (vgl. Wegener, 2000). Für sehr eingeschränkte Branchingprogrammmodelle hingegen, z.B. solchen wie wir sie weiter oben definiert haben, wurden bisher noch keine nichttrivialen oberen Schranken bewiesen. Da allerdings jede Funktion aus B_n eine OBDD-Größe von $O(2^n/n)$ hat, folgt eine obere Schranke von $O\big(2^{2n}/n\big)$ für die OBDD-Größe des mittleren Bits der Multiplikation.

Für Branchingprogramme, die alle Ausgabebits von MUL_n an den Kanten berechnen, gibt Dietzfelbinger (1996) als Korollar ein Time-Space-Tradeoff von $\Omega(n^2)$ an, das auf Erkenntnissen von Mansour, Nisan und Tiwari (1993) über die Komplexität von Hashfunktionen beruht. Diese untere Schranke ist offensichtlich optimal, da man leicht ein Time-Space-Produkt von $O(n^2)$ mit einem Branchingprogramm erreicht, das ein vollständiger binärer Baum der Tiefe $2n$ ist. Überträgt man den Beweis der unteren Schranke auf nur ein Ausgabebit, so führt dies nur zu einer trivialen unteren Schranke von $\Omega(n)$ (s. auch Kapitel 3, Theorem 3.2). Selbst für OBDDs impliziert diese Technik somit nur mindestens lineare Größe.

Exponentielle untere Schranken für die Komplexität des mittleren Bits der Multiplikation konnten jedoch in einigen eingeschränkten Branchingprogrammmodellen bewiesen werden. So hat Bryant bereits 1986 gezeigt, dass es für jede Variablenordnung π ein Ausgabebit $i \in \{0, \dots, 2n-1\}$ gibt, sodass das minimale π-OBDD für $\mathrm{MUL}_{i,n}$ mindestens $2^{n/8}$ Knoten hat. Dieses Resultat bedeutet natürlich noch nicht, dass OBDDs für die Verifikation von Multiplizierern ungeeignet sind, da es nicht ausschließt, dass es für jede der Funktionen $\mathrm{MUL}_{i,n}$, $0 \leq i \leq 2n-1$, eine Variablenordnung π_i gibt, sodass das π_i-OBDD für $\mathrm{MUL}_{i,n}$ nur polynomielle Größe hat. Bryant bewies aber einige Jahre später, dass das mittlere Ausgabebit der Multiplikation, also die Funktion $\mathrm{MUL}_{n-1,n}$, sogar für jede Variablenordnung exponentielle Größe hat.

Theorem 2.13 (Bryant, 1991) *Jedes* OBDD *für* $\mathrm{MUL}_{n-1,n}$ *hat mindestens* $2^{n/8}$ *Knoten.*

Einige Jahre nach Bryants unterer Schranke hat Gergov (1994) den Beweis für längenbeschränkte *stereotype* (engl. oblivious) Branchingprogramme verallgemeinert. Das sind Branchingprogramme, bei denen sich die Knoten – wie bei OBDDs – in Ebenen einteilen lassen, sodass alle Knoten einer Ebene mit der gleichen Variablen markiert sind und Kanten nur von einer Ebene zu einer weiter unten liegenden Ebene verlaufen. Tatsächlich sind OBDDs stereotype FBDDs. Gergov hat bewiesen, dass jedes stereotype Branchingprogramm der Länge $2kn$ für $\mathrm{MUL}_{n-1,n}$ mindestens $2^{\Omega(n/k^3 2^{4k})}$ Knoten hat, was für $k = o(\log n / \log\log n)$ nicht polynomiell ist. Der Beweis überträgt im Wesentlichen mithilfe eines Lemmas von Alon und Maass (1988) eine allgemeine untere Schrankentechnik für OBDDs auf stereotype Branchingprogramme und wendet diese dann auf die Eigenschaften von $\mathrm{MUL}_{n-1,n}$ an, die auch Bryant für die OBDD-Schranke ausgenutzt hat.

Auf FBDDs lässt sich Bryants Beweisidee jedoch nicht ohne weiteres übertragen. Mit einem wesentlich komplexeren Beweis konnte Ponzio (1995, 1998) aber eine schwach exponentielle untere Schranke für das mittlere Bit der Multiplikation zeigen.

Theorem 2.14 (Ponzio, 1998) *Jedes* FBDD *für* $\mathrm{MUL}_{n-1,n}$ *hat* $2^{\Omega(\sqrt{n})}$ *Knoten.*

Viele untere Schranken für OBDDs gelten auch für nichtdeterministische OBDDs, da sie auf den gleichen Argumenten beruhen, mit denen untere Schranken für nichtdeterministische Kommunikationskomplexität bewiesen werden. So ist es nicht überraschend, dass Bryants Beweis für OBDDs auch die gleiche untere Schranke für nichtdeterministische OBDDs liefert. Bollig (2001) hat eine exponentielle untere Schranke von $2^{\Omega(n/\log n)}$ für $\mathrm{MUL}_{n-1,n}$ in einem etwas allgemeineren nichtdeterministischen Modell gezeigt, nämlich für nichtdeterministische baumgesteuerte FBDDs. Das sind nichtdeterministische graphgesteuerte FBDDs, bei denen die Graphordnung ein Baum polynomieller Größe ist (also eine Verallgemeinerung von OBDDs, bei denen die Graphordnung eine Liste ist). Für den Beweis werden ähnliche Erkenntnisse über die Multiplikation ausgenutzt wie in Bryants Beweis für OBDDs.

Schließlich haben noch Ablayev und Karpinski (1998) die Darstellung der Multiplikation für *randomisierte* OBDDs untersucht. Bei solchen OBDDs gibt es zusätzliche Knoten mit Ausgangsgrad 2, an denen während einer Berechnung ein zufälliger Nachfolger gewählt wird. Auf diese Weise wird jede Eingabe mit einer bestimmten Wahrscheinlichkeit akzeptiert bzw. verworfen. Ablayev und Karpinski bewiesen, dass ein randomisiertes OBDD eine Größe von mindestens $2^{\Omega(n/\log n)}$ hat, wenn es zu jeder Eingabe (x, y) den Funktionswert $\mathrm{MUL}_{n-1,n}(x, y)$ mit einer konstanten Wahrscheinlichkeit $\epsilon > 1/2$ korrekt berechnet.

Obwohl man für einige beschränkte Branchingprogrammmodelle exponentielle untere Schranken kennt, bleiben noch wichtige Fragen ungeklärt. So bleibt z.B. offen, ob 64- oder gar 128-Bit Multiplikationsschaltkreise mit Branchingprogrammen verifiziert werden können, da keine der o.g. unteren Schranken ausschließt, dass es OBDDs mit weniger als 100 000 Knoten für das mittlere Bit der 128-Bit-Multiplikation gibt. Wie oben angedeutet, legen alle empirischen Ergebnisse aber nahe, dass jedes OBDD für einen 16-Bit Multiplizierer aus mehreren Millionen Knoten besteht. Bollig und Wegener (1999, S. 3) äußern daher die Vermutung, dass die OBDD-Größe von $\mathrm{MUL}_{n-1,n}$ mindestens in der Größenordnung von 2^n liegt.

Auch wenn Ponzios unterer Schranke von $2^{\Omega(\sqrt{n})}$ für FBDDs ein wichtiger Durchbruch bei der Erforschung der Komplexität der Multiplikation in Branchingprogrammmodellen war, ist eine schwach exponentielle untere Schranke noch nicht zufrieden stellend, da auch hier eher eine Größenordnung von $2^{\Omega(n)}$ vermutet wird. Alle anderen Beweise unterer Schranken für die Multiplikation basieren sehr stark auf den kommunikationskomplexitätstheoretischen Argumenten aus Bryants Beweis und demonstrieren neue Beweistechniken für die jeweils betrachteten Branchingprogrammmodelle, liefern aber kaum neue Einsichten in die Struktur der Multiplikation. Letztendlich konnte bis vor Beginn der hier dokumentierten Forschung keine superpolynomielle untere Schranke für die Komplexität von $\mathrm{MUL}_{n-1,n}$ in einem Branchingprogrammmodell bewiesen werden, das echt allgemeiner als das FBDD-Modell ist.

2.3 Ergebnisse dieser Arbeit

Die Untersuchungen im Rahmen dieser Dissertation haben zu neuen Erkenntnissen über die Komplexität der Multiplikation geführt. Einer der entscheidenden Gründe für das Entstehen der Resultate ist eine neue Sichtweise auf diese Funktion. So basieren alle hier vorgestellten unteren Schranken darauf, dass mithilfe der Multiplikation sog. universelle Hashklassen konstruiert werden können (Dietzfelbinger, Hagerup, Katajainen und Penttonen, 1997; Dietzfelbinger, 1996; Woelfel, 1999, 2000) und nutzen Eigenschaften aus, die sich für die Multiplikation aus dieser Tatsache ableiten lassen. Wir werden daher im nächsten Kapitel zunächst eine kurze Einführung in das Konzept des universellen Hashings geben und den Zusammenhang zur Multiplikation darstellen.

In Kapitel 4 untersuchen wir dann die OBDD-Größe von $\mathrm{MUL}_{n-1,n}$ und beweisen eine untere Schranke von $2^{n/2}/61$ sowie eine obere Schranke von $O\big(2^{4n/3}\big)$. Die untere Schranke belegt, dass das kleinste OBDD für das mittlere Bit der 64-Bit-Multiplikation aus mehr als 70 Millionen Knoten besteht – bei der 128-Bit-Multiplikation sind es sogar schon mehr als $3 \cdot 10^{17}$ Knoten. Damit ist erstmals bewiesen, dass mit heutigen Mitteln 128-Bit-Multiplikationsschaltkreise nicht mit OBDDs verifiziert werden können.

In Kapitel 5 zeigen wir dann eine untere Schranke von $\Omega\big(2^{n/4}\big)$ für die FBDD-Größe von $\mathrm{MUL}_{n-1,n}$. Dies ist die erste echt exponentielle untere Schranke für diese Funktion. Einige Aussagen, die wir in den Kapiteln 4 und 5 herleiten, beschreiben grundsätzliche Erkenntnisse über die Multiplikation. In den Kapiteln 6 und 7 entwickeln wir Techniken zum Beweis unterer Schranken für noch allgemeinere Branchingprogrammmodelle, die zu diesen Erkenntnissen „passen“. Wir zeigen zunächst in Kapitel 6 eine untere Schranke von $\Omega\big(2^{\frac{n}{48(k+1)}}\big)$ für die Größe von semantischen $(1,+k)$-BPs, die die Funktion $\mathrm{MUL}_{n-1,n}$ darstellen. Somit ist die $(1,+k)$-BP-Größe dieser Funktion für $k = o(n/\log n)$ noch superpolynomiell. In Kapitel 7 betrachten wir schließlich ein FBDD-Modell mit eingeschränktem Nichtdeterminismus. Ein sog. $(\vee,k)$-FBDD entspricht einem nichtdeterministischen FBDD mit höchstens $k-1$ nichtdeterministischen Knoten, die auf jedem Pfad nur vor den deterministischen Knoten vorkommen dürfen. Wir beweisen in diesem Modell eine untere Schranke von $\Omega\big(2^{n/(7k)}\big)$ für das mittlere Bit der Multiplikation. Sowohl $(1,+k)$-BPs als auch $(\vee,k)$-FBDDs sind die ersten Branchingprogrammmodelle, die echt allgemeiner als FBDDs sind und für die superpolynomielle untere Schranken für die Komplexität der Multiplikation gezeigt werden konnten.

Die hier vorgestellten Resultate beruhen auf folgenden Publikationen: Woelfel (2001) beschreibt die Ergebnisse für OBDDs, Bollig und Woelfel (2001) behandeln FBDDs und bei Woelfel (2002b) werden die unteren Schranken für $(1,+k)$-BPs und $(\vee,k)$-FBDDs hergeleitet.

KAPITEL 3

Universelles Hashing und der Zusammenhang zur Multiplikation

Hashing gehört zu den grundlegendsten Konzepten der Algorithmik. Dabei ist eine endliche Menge U von sog. *Schlüsseln* sowie eine Teilmenge $S \subseteq U$ gegeben. Die Schlüssel aus S sollen mit Hilfe einer *Hashfunktion* $h : U \to \{0, \ldots, k-1\}$ auf die k Tabellenplätze einer Tabelle abgebildet werden. Eine typische Anwendung ist z.B. das Wörterbuchproblem, bei dem S die Menge der Schlüsselwörter darstellt und die zu einem Schlüsselwort $x \in S$ zu speichernde Information in der Tabelle an der Position $h(x)$ abgelegt werden soll. Da die Hashfunktion i.A. nicht für alle $S \subseteq U$ injektiv sein kann, muss mit *Kollisionen* von Schlüsseln gerechnet werden, also damit, dass es verschiedene Schlüssel $x, y \in S$ mit $h(x) = h(y)$ gibt. Um mit solchen Schlüsselkollisionen umzugehen, wurden zahlreiche Verfahren entwickelt, die z.B. Schlüssel bei auftretenden Kollisionen auf andere Tabellenplätze verteilen oder alle Schlüssel, die auf eine Tabellenposition abgebildet werden, in einer an dieser Tabellenposition gespeicherten Datenstruktur – wie z.B. einer verketteten Liste – speichern.

Typischerweise ist das Universum U viel größer als die Menge S der zu speichernden Schlüssel und die Tabellengröße k sollte in der Größenordnung von $|S|$ liegen. Dadurch erhält man bei der Wahl einer festen Hashfunktion (also unabhängig von S) im Worst-Case immer ein schlechtes Laufzeitverhalten. Denn mit Hilfe des Schubfachprinzips sieht man, dass es zu jeder Hashfunktion $h : U \to \{0, \ldots, k-1\}$ eine Menge $S \subseteq U$ der Kardinalität n gibt, aus der mindestens $\min\{n, |U|/k\}$ Schlüssel auf die gleiche Tabellenposition abgebildet werden. Aus diesem Grund beschränkt man sich bei der Analyse von Wörterbuch-Algorithmen, die auf einer fest gewählten Hashfunktion basieren, meist auf den Average-Case, d.h. auf das Verhalten unter der Annahme, dass S zufällig gewählt wird.

Carter und Wegman (1979) haben deshalb randomisierte Hashverfahren vorgeschlagen, bei denen die Hashfunktion nicht für jede Eingabe fest gewählt wird, sondern zufällig aus einer Familie $\mathcal{H}$ von Hashfunktionen $U \to R$ für eine endliche Menge R. Die Familie $\mathcal{H}$ bezeichnet man als *Hashklasse*, und R ist der *Wertebereich* (engl. *range*). Bei geeigneter Wahl der Hashklasse $\mathcal{H}$ kann dann die Analyse für jede Menge $S \subseteq U$ in Abhängigkeit von der zufälligen Wahl der Hashfunktion erfolgen. Natürlich muss die Hashklasse gewisse Eigenschaften aufweisen, damit die Algorithmen sich für jede Menge $S \subseteq U$ gut verhalten.

In den Arbeiten Carter und Wegman (1979) sowie Wegman und Carter (1979) wurde mit folgender Definition der Begriff des *universellen Hashings* geprägt.

Definition 3.1 Eine Familie $\mathcal{H}$ von Hashfunktionen $U \to R$ heißt *universell*, wenn für alle verschiedenen Schlüssel $x, x' \in U$ und eine zufällig gemäß der Gleichverteilung gewählte

Hashfunktion $h \in \mathcal{H}$ die Wahrscheinlichkeit für das Ereignis $h(x) = h(x')$ durch $1/|R|$ nach oben beschränkt ist. Wenn $\big(h(x), h(x')\big)$ sogar gleichverteilt über $R \times R$ ist, so heißt die Familie $\mathcal{H}$ *streng universell*.

Offensichtlich ist jede streng universelle Hashklasse auch universell. Inzwischen sind zahlreiche universelle und streng universelle Hashklassen bekannt, deren Funktionen effizient berechnet werden können. Die wichtigsten Beispiele sind Klassen von linearen Funktionen über endlichen Körpern, Polynomringen oder $\mathbb{Z}/n\mathbb{Z}$.

Abgesehen von der Anwendung beim Wörterbuchproblem wurde universelles Hashing auch zur Lösung zahlreicher anderer algorithmischer Probleme eingesetzt, wie z.B. bei der Authentifizierung von Nachrichten (Wegman und Carter, 1979; Atici und Stinson, 1996), beim Sortieren ganzer Zahlen (Matias und Vishkin, 1991; Andersson, Hagerup, Nilsson und Raman, 1995) oder bei geometrischen Problemen (Dietzfelbinger, Hagerup, Katajainen und Penttonen, 1997). Es gibt aber auch einige Beispiele aus der Komplexitätstheorie, bei denen universelles Hashing verwendet wurde – eine Reihe von Beispielen beschreiben Arvind und Mahajan (1996). Während aber bei den meisten Anwendungen aus der Komplexitätstheorie Hashfunktionen eher als Hilfsmittel für die Beweisführung eingesetzt wurden, haben Mansour, Nisan und Tiwari (1993) die Komplexität von Hashfunktionen streng universeller[1] Hashklassen selbst untersucht.

Theorem 3.2 (Mansour, Nisan und Tiwari, 1993) *Sei $\mathcal{H} = \{g_0, \dots, g_{2^n-1}\}$ eine streng universelle Hashklasse mit Universum $U = \{0,1\}^k$ und Wertebereich $R = \{0,1\}^m$, und sei $f \in B_{n+k,m}$ die boolesche Funktion, die zur bitweisen Beschreibung einer Hashfunktion h und eines Schlüssels x den Hashwert $g(x)$ berechnet, also die Abbildung $z_{n-1} \dots z_0 x_1 \dots x_k \mapsto g_{|z|}(x)$. Dann gilt $T \cdot S = \Omega(m \cdot n)$ für jedes Branchingprogramm mit Ausgabe an den Kanten, das in Zeit T und Platz S die Funktion f berechnet.*

Dieses Resultat führt direkt zu unteren Schranken für den Time-Space-Tradeoff (s. S. 7) von so wichtigen Funktionen wie der Faltung über $\{0,1\}^n$, den linearen Funktionen über endlichen Körpern oder der Matrizenmultiplikation. Den Zusammenhang zur Multiplikation stellt Dietzfelbinger (1996) dar, indem er beweist, dass man mit linearen Funktionen über dem Ring $\mathbb{Z}/2^n\mathbb{Z}$ eine streng universelle Hashklasse bilden kann.

Im Folgenden bezeichnen wir mit „div" die ganzzahlige Division ohne Rest, d.h. $x \operatorname{div} y = \lfloor x/y \rfloor$ für $x, y \in \mathbb{Z}$.

Theorem 3.3 (Dietzfelbinger, 1996) *Sei für $m \le n$ und $a, b \in \mathbb{Z}$ die Funktion $g_{a,b} : \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^m}$ definiert durch $x \mapsto \big((a \cdot x + b) \bmod 2^{n+m}\big) \operatorname{div} 2^n$. Dann bildet die Familie der Funktionen $g_{a,b}$ mit $a, b \in \mathbb{Z}_{2^{n+m}}$ eine streng universelle Hashklasse.*

Es ist entscheidend für die Praxistauglichkeit der Hashklasse und auch für unsere späteren Untersuchungen, dass das Hintereinanderschalten der Modulooperation und der ganzzahligen Division einfach durch das „Ausschneiden" von Bits realisiert werden kann.

Bemerkung 3.4 *Für alle $x \in \mathbb{B}_n$ und $0 \le \ell < k \le n$ gilt*

$$\big(|x_{n-1} \dots x_0| \bmod 2^k\big) \operatorname{div} 2^\ell \;=\; |x_{k-1} \dots x_\ell| \;=\; \big|\langle |x| \rangle_\ell^{k-1}\big|.$$

[1]In der Arbeit von Mansour et al. werden Hashklassen, die nach unserer Definition 3.1 streng universell sind, als universell bezeichnet.

Mit einer einfachen Reduktion folgt aus den Theoremen 3.2 und 3.3 sofort ein Time-Space-Tradeoff von $\Omega(n^2)$ für die Funktion MUL_n.

Korollar 3.5 (Dietzfelbinger, 1996) *Für jedes Branchingprogramm (mit Ausgabe an den Kanten), das in Zeit T und Platz S die Funktion* MUL_n *berechnet, gilt $T \cdot S = \Omega(n^2)$.*

Beweis: Offensichtlich gilt für $a, b \in \mathbb{B}_{2n}$, $x \in \mathbb{B}_n$ und $y \in \mathbb{B}_{3n+1}$ mit $|y| = |a| \cdot |x| + |b|$

$$\begin{aligned}
&\left|b_{2n-1} \dots b_0 a_{2n-1} \dots a_0\right| \cdot \left|0^n\, x_{n-1} \dots x_0\, 0^{2n-1} 1\right| \\
&\quad = \left(2^{2n} \cdot |b| + |a|\right) \cdot \left(2^{2n} \cdot |x| + 1\right) \\
&\quad = 2^{4n} \cdot |b| \cdot |x| + 2^{2n}(|x| \cdot |a| + |b| \cdot 1) + |a| \cdot 1 \\
&\quad = 2^{4n} \cdot |b| \cdot |x| + 2^{2n}|y| + |a| \\
&\quad = 2^{4n} \cdot \left(|y| \operatorname{div} 2^{2n} + |b| \cdot |x|\right) + 2^{2n} \cdot \left(|y| \bmod 2^{2n}\right) + |a|.
\end{aligned}$$

Ist $z \in \mathbb{B}_{8n}$ die Ausgabe eines Branchingprogramms G, das zwei $4n$-Bit Zahlen multipliziert, und bettet man a, b und x so in die Eingabe ein, wie es aus der obigen Formel ersichtlich ist, so sind die Ausgabebits $z_{4n-1} \dots z_{2n}$ die Binärdarstellung von $(|a| \cdot |x| + |b|) \bmod 2^{2n}$. Demnach stellen aber die Ausgabebits $z_{4n-1} \dots z_{3n}$ den Funktionswert $g_{|a|,|b|}(x)$ der in Theorem 3.3 definierten Funktion $g_{|a|,|b|}$ (mit $m = n$) dar. Mit Theorem 3.2 folgt somit $T \cdot S = \Omega(n^2)$ für die Zeit T und den Platz S von G. ∎

Offensichtlich sind Funktionen, die streng universelle Hashklassen bilden, für Branchingprogramme „schwierig“ zu berechnen, zumindest solange auch viele Ausgabebits betrachtet werden. Für einzelne Ausgabebits, d.h. für $m = 1$, erhalten wir mit Theorem 3.2 nur die triviale Aussage, dass der Time-Space-Tradeoff $\Omega(n)$ beträgt, also dass z.B. jedes FBDD mindestens lineare Größe hat. Tatsächlich gibt es sogar streng universelle Hashklassen, deren Funktionen mit OBDDs linearer Größe berechnet werden können.

Für $x \in \{0,1\}^n$ und $y \in \{0,1\}^{n+m-1}$ sei die Faltung von x und y definiert als $x \circ y = z \in \{0,1\}^m$ mit $z_k = \left(\sum_{i=1}^n x_i y_{i+k-1}\right) \bmod 2$. Weiterhin bezeichne $a \oplus b$ die komponentenweise Addition modulo 2 von $a \in \{0,1\}^n$ und $b \in \{0,1\}^n$.

Theorem 3.6 (Mansour, Nisan und Tiwari, 1993) *Die Menge der Hashfunktionen aus $B_{n,m}$ definiert durch $x \mapsto (a \circ x) \oplus b$ mit $a \in \{0,1\}^{n+m-1}$ und $b \in \{0,1\}^m$ ist streng universell.*

Mit einer ähnlichen Reduktion wie der aus Korollar 3.5 erhält man sofort einen Time-Space-Tradeoff von $\Omega(n^2)$ für ein Branchingprogramm, das alle n Ausgabebits der Faltung eines $2n$-Bit Strings mit einem $(n-1)$-Bit String berechnet. Trotzdem konstruiert man für eine geeignete Variablenordnung π ganz einfach ein OBDD linearer Größe für ein Ausgabebit der Faltung.

Bemerkung 3.7 *Sei $1 \le k \le m$. Dann lässt sich die Abbildung*

$$x_n \dots x_1 y_{n+m-1} \dots y_1 \;\mapsto\; \left(\sum_{i=1}^{n} x_i y_{i+k-1}\right) \bmod 2$$

durch ein OBDD *der Größe $O(n)$ darstellen.*

Beweis: Das OBDD liest die Variablen $x_1, y_k, x_2, y_{k+1}, \ldots, x_n, y_{n+k-1}$ in dieser Reihenfolge und auf jeder Ebene enthält das OBDD 2 Knoten – nur die erste Ebene besteht aus einem Knoten (der Wurzel). Wir können die Knoten des OBDDs einer Ebene als Speicher auffassen, in dem einer von 2 Werten gespeichert werden kann, und zwar wird jeweils auf der x_ℓ-Ebene der Wert von $s_\ell = \left(\sum_{i=1}^{\ell-1} x_i y_{k-1+i}\right) \bmod 2$ gespeichert (auf der ersten Ebene gilt $s_1 = 0$; darum genügt dort ein Knoten). Die ausgehende 1-Kante des x_ℓ-Knotens führt dann zu dem $y_{k-1+\ell}$-Knoten, der den Speicherwert s_ℓ darstellt, und die 0-Kante führt direkt zum $x_{\ell+1}$-Knoten, der dem Wert s_ℓ entspricht (für $\ell = n$ führt die Kante zur s_ℓ-Senke). Von der $y_{k-1+\ell}$-Ebene führt die c-Kante, $c \in \{0, 1\}$ zum $x_{\ell+1}$ Knoten, der den Speicherwert $(s_\ell + c) \bmod 2$ repräsentiert (bzw. für $\ell = n$ zur entsprechend markierten Senke). Korrektheit und behauptete Größe des OBDDs sind offensichtlich. ■

Dieses Beispiel zeigt, dass die Berechnung eines einzelnen Ausgabebits von streng universellen Hashklassen leicht sein kann. Also kann auch nicht alleine aus der Tatsache, dass man mithilfe der linearen Funktionen über $\mathbb{Z}/2^n\mathbb{Z}$ eine streng universelle Hashklasse erhält, auf gute untere Schranken für die Funktion $\mathrm{MUL}_{n-1,n}$ geschlossen werden. Trotzdem spielt die Universalität ähnlicher Hashklassen bei den folgenden Beweisen unterer Schranken für das mittlere Bit der Multiplikation eine ganz entscheidende Rolle. Für diese Beweise könnte man auch die streng universell Hashklasse aus Theorem 3.3 benutzen, jedoch genügt uns die einfache Universalität und die weiter unten vorgestellte multiplikative Hashklasse führt zu etwas besseren unteren Schranken.

Dietzfelbinger, Hagerup, Katajainen und Penttonen (1997) haben für $m \le n$ und $a \in \mathbb{Z}_{2^n}$ die Abbildungen $h_a : \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^m}$, $x \mapsto \big((ax) \bmod 2^n\big) \operatorname{div} 2^{n-m}$ betrachtet und bewiesen, dass die Menge der Funktionen h_a mit ungeradem $a \in \mathbb{Z}_{2^n}$ eine fast universelle Hashklasse bildet. Sei im Folgenden $\mathbb{Z}_k^*$ die Menge der invertierbaren Elemente des Rings $\mathbb{Z}/k\mathbb{Z}$, d.h. für $k = 2^n$ die Menge der ungeraden Zahlen aus $\mathbb{Z}_{2^n}$.

Theorem 3.8 (Dietzfelbinger, Hagerup, Katajainen und Penttonen 1997)
Für beliebige verschiedene $x, x' \in \mathbb{Z}_{2^n}$ und ein zufällig gewähltes $a \in \mathbb{Z}_{2^n}^*$ ist die Wahrscheinlichkeit für das Ereignis

$$\big((ax) \bmod 2^n\big) \operatorname{div} 2^{n-m} = \big((ax') \bmod 2^n\big) \operatorname{div} 2^{n-m}$$

durch $2/2^m$ beschränkt.

Der Unterschied zur streng universellen Hashklasse aus Theorem 3.3 liegt also darin, dass man einerseits auf den zufälligen additiven Term verzichten kann und andererseits nur eine n-Bit-Multiplikation statt einer $(n+m)$-Bit-Multiplikation benötigt. Woelfel (1999, 2000) hat im Rahmen seiner Diplomarbeit gezeigt, dass man eine universelle Hashklasse erhält, wenn man zwar eine zusätzliche Addition in Kauf nimmt, aber trotzdem nur mit n-Bit Zahlen rechnet.

Definition 3.9 Die *multiplikative Hashklasse* $\mathcal{M}_{n,m}$, $m \le n$, besteht aus allen Funktionen $h_{a,b}^m : \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^m}$, $x \mapsto \big((ax + b) \bmod 2^n\big) \operatorname{div} 2^{n-m}$ mit $a \in \mathbb{Z}_{2^n}^*$ und $b \in \mathbb{Z}_{2^{n-m}}$.

Theorem 3.10 (Woelfel 1999) *Die multiplikative Hashklasse ist universell.*

Tatsächlich genügt es für die Universalität der Hashklasse, bei der Addition noch einen kürzeren Bit-String zu wählen. Der Beweis ist dann aber komplizierter und für unsere Zwecke genügt das hier angegebene Ergebnis. Da obiges Theorem essentiell für alle unteren Schranken der nachfolgenden Kapitel ist, geben wir im Folgenden den Beweis an. Wir benutzen dafür ein Lemma, das den Einfluss der zufälligen Addition modulo 2^n in Zusammenhang mit der ganzzahligen Division ohne Rest beschreibt.

Lemma 3.11 *Seien $x, x' \in \mathbb{Z}_{2^n}$ und $d = (x' - x) \bmod 2^n$. Dann gilt für ein zufällig gewähltes $b \in \mathbb{Z}_{2^{n-m}}$*

$$\mathbf{Prob}\big(((x+b) \bmod 2^n) \operatorname{div} 2^{n-m} = ((x'+b) \bmod 2^n) \operatorname{div} 2^{n-m}\big)$$
$$= \begin{cases} 1 - d/2^{n-m} & \textit{falls} \quad 0 \le d < 2^{n-m}, \\ 0 & \textit{falls} \quad 2^{n-m} \le d \le 2^n - 2^{n-m}, \\ 1 - (2^n - d)/2^{n-m} & \textit{falls} \quad 2^n - 2^{n-m} < d < 2^n. \end{cases}$$

Beweis: Sei $z_b = (x + b) \bmod 2^n$ und $z'_b = (x' + b) \bmod 2^n$. Wir betrachten zunächst den Fall $2^{n-m} \le d < 2^n - 2^{n-m}$ für beliebiges $b \in \mathbb{Z}_{2^{n-m}}$. Sei $k = z_b \operatorname{div} 2^{n-m}$, also $k \cdot 2^{n-m} \le z_b < (k+1) \cdot 2^{n-m}$. Nach Voraussetzung ist

$$2^{n-m} \le (z'_b - z_b) \bmod 2^n \le 2^n - 2^{n-m}$$

und daher

$$z'_b \notin \big\{k \cdot 2^{n-m}, \ldots, (k+1) \cdot 2^{n-m} - 1\big\}.$$

Somit folgt $z_b \operatorname{div} 2^{n-m} = k \neq z'_b \operatorname{div} 2^{n-m}$.

Wir betrachten nun den Fall $0 \le d < 2^{n-m}$. Sei $\tau = x \bmod 2^{n-m}$ und $k = x \operatorname{div} 2^{n-m}$. Da $z_b = (k \cdot 2^{n-m} + \tau + b) \bmod 2^n$ ist, gilt

$$z_b \operatorname{div} 2^{n-m} = \begin{cases} k & \text{für} \quad 0 \le b < 2^{n-m} - \tau, \\ (k+1) \bmod 2^m & \text{für} \quad 2^{n-m} - \tau \le b < 2^{n-m}. \end{cases}$$

Es ist $z'_b = (k \cdot 2^{n-m} + \tau + d + b) \bmod 2^n$, und so folgt

$$z'_b \operatorname{div} 2^{n-m} = \begin{cases} k & \text{für} \quad 0 \le b < 2^{n-m} - \tau - d, \\ (k+1) \bmod 2^m & \text{für} \quad 2^{n-m} - \tau - d \le b < 2 \cdot 2^{n-m} - \tau - d, \\ (k+2) \bmod 2^m & \text{für} \quad 2 \cdot 2^{n-m} - \tau - d \le b < 2^{n-m}. \end{cases}$$

Insgesamt ist daher $z_b = z'_b$ mit $b \in \mathbb{Z}_{2^{n-m}}$ äquivalent zu

$$0 \le b < 2^{n-m} - \tau - d \quad \text{oder} \tag{3.1}$$
$$2^{n-m} - \tau \le b < \min\big\{2^{n-m},\ 2 \cdot 2^{n-m} - \tau - d\big\}. \tag{3.2}$$

Für $\tau + d \ge 2^{n-m}$ ist (3.1) nie erfüllt, und (3.2) hat genau $2^{n-m} - d$ Lösungen. Für $\tau + d < 2^{n-m}$ hat (3.1) genau $2^{n-m} - \tau - d$ und (3.2) genau τ Lösungen. In jedem Fall gibt es insgesamt genau $2^{n-m} - d$ Werte $b \in \mathbb{Z}_{2^{n-m}}$ mit $z_b = z'_b$.

Für den letzten Fall, $2^n - 2^{n-m} < d \le 2^n$, betrachten wir $d' = (x - x') \bmod 2^n = 2^n - d$. Da $0 \le d' < 2^{n-m}$ ist, folgt die Behauptung aus dem zweiten Fall durch Vertauschen von x und x'. ∎

Beweis zu Theorem 3.10: Seien $x < x'$ beliebige verschiedene Elemente aus $\mathbb{Z}_{2^n}$ und sei $d_a = (ax' - ax) \bmod 2^n$. Wir können $x' - x$ als $r \cdot 2^s$ schreiben, für ein ungerades $r < 2^{n-s}$. Somit gilt für alle $a \in \mathbb{Z}_{2^n}^*$

$$d_a \;=\; (ax' - ax) \bmod 2^n \;=\; (ar2^s) \bmod 2^n \;=\; (ar) \bmod 2^{n-s} \cdot 2^s. \tag{3.3}$$

Die letzte Gleichung folgt mit Bemerkung 3.4, da für den Bitstring z mit $|z| = ar$ gilt $ar2^s = |z\,0^s| = |z| \cdot 2^s$.

Wenn nun a zufällig aus $\mathbb{Z}_{2^n}^*$ gewählt wird, so ist $a \bmod 2^{n-s}$ gleichverteilt über die multiplikative Gruppe $\mathbb{Z}_{2^{n-s}}^*$. Da auch r Element der Gruppe ist, ist $(ar) \bmod 2^{n-s}$ auch gleichverteilt über $\mathbb{Z}_{2^{n-s}}^*$. Mit Gleichung (3.3) folgt dann, dass d_a gleichverteilt über

$$M \;=\; \left\{1 \cdot 2^s, 3 \cdot 2^s, \ldots, (2^{n-s} - 1) \cdot 2^s\right\}$$

ist.

Für zufällig gewähltes $b \in \mathbb{Z}_{2^{n-m}}$ tritt gemäß Lemma 3.11 das Ereignis $h^m_{a,b}(x) = h^m_{a,b}(x')$ unter der Bedingung $d_a < 2^{n-m}$ mit Wahrscheinlichkeit $1 - d_a/2^{n-m}$ ein, unter der Bedingung $d_a > 2^n - 2^{n-m}$ mit Wahrscheinlichkeit $1 - (2^n - d_a)/2^{n-m}$, und sonst gar nicht. Da wie oben gezeigt d_a für zufällig gewähltes $a \in \mathbb{Z}_{2^n}$ gleichverteilt über M ist, folgt für $M_1 = M \cap \left\{0, \ldots, 2^{n-m} - 1\right\}$ und $M_2 = M \cap \left\{2^n - 2^{n-m} + 1, \ldots, 2^n - 1\right\}$

$$\begin{aligned}
&\mathbf{Prob}\left(h^m_{a,b}(x) = h^m_{a,b}(x')\right) \\
&= \frac{1}{|M|}\left(\sum_{d \in M_1}\left(1 - d/2^{n-m}\right) + \sum_{d \in M_2}\left(1 - (2^n - d)/2^{n-m}\right)\right) \\
&= \frac{2}{|M|}\sum_{d \in M_1}\left(1 - d/2^{n-m}\right).
\end{aligned}$$

Die letzte Gleichung gilt, weil M_2 genau die Zahlen $2^n - d$ mit $d \in M_1$ enthält. Es ist $M_1 = \left\{1 \cdot 2^s, 3 \cdot 2^s, \ldots, (2^{n-m-s} - 1) \cdot 2^s\right\}$. Für $s \geq n - m$ ist dies die leere Menge und x und x' kollidieren unter keiner Funktion $h^m_{a,b}$. Falls $s < n - m$ ist, erhalten wir unter Ausnutzung von $1 + 3 + \ldots + (2k - 1) = k^2$

$$\begin{aligned}
&\mathbf{Prob}\left(h^m_{a,b}(x) = h^m_{a,b}(x')\right) \\
&= \frac{2}{|M|} \cdot \left(|M_1| - \frac{2^s}{2^{n-m}}\left(1 + 3 + \ldots + (2^{n-m-s} - 1)\right)\right) \\
&= \frac{2}{2^{n-s-1}} \cdot \left(2^{n-m-s-1} - 2^{s-n+m} \cdot \left(2^{n-m-s-1}\right)^2\right) \\
&= 2^{s-n+2} \cdot 2^{n-m-s-2} \;=\; 2^{-m}.
\end{aligned}$$

Da 2^m die Größe des Wertebereichs ist, folgt die Behauptung. ■

KAPITEL 4

OBDDs

Wie in Abschnitt 2.2 erläutert, hat Bryant (1991) eine untere Schranke von $2^{n/8}$ für die OBDD-Größe der Funktion $\mathrm{MUL}_{n-1,n}$ bewiesen. Dass die untere Schranke vermutlich noch weit von der Wahrheit entfernt ist, haben wir damit begründet, dass empirische Versuche, 16-Bit Multiplikationsschaltkreise in OBDDs zu transformieren, lange Zeit gescheitert sind und der erfolgreiche Versuch von Yang, Chen, Bryant und O'Hallaron (1998) zu einem OBDD mit mehr als 40 Millionen Knoten geführt hat. Selbst für $n = 64$ bzw. $n = 128$ liefert Bryants Beweis aber nur untere Schranken von 256 bzw. 65536. Im ersten Teil dieses Kapitels beweisen wir daher eine bessere untere Schranke von $2^{n/2}/61 - 4$, die aufzeigt, dass OBDDs für das mittlere Bit eines 64-Bit Multiplizierers mehr als 70 Millionen Knoten benötigen – bei einem 128-Bit Multiplizierer sind es schon mehr als $3 \cdot 10^{17}$ Knoten. Der Versuch, 64-Bit Multiplikationsschaltkreise mit OBDDs zu verifizieren, benötigt daher schon eine extrem große Menge an Ressourcen und für 128-Bit Multiplizierer ist dies mit heutigen Mitteln unmöglich.

4.1 Eine neue untere Schranke

Um die Idee, die zu einer besseren unteren Schranke führt, zu motivieren, ist ein Blick auf Bryants Beweis hilfreich. Er betrachtet Subfunktionen der Multiplikation, die durch das Konstantsetzen eines Faktors entstehen.

Definition 4.1 Für festes $a \in \mathbb{Z}_{2^n}$ und $0 \le i \le n-1$ sei $\mathrm{MUL}^a_{i,n} : \mathbb{B}_n \to \{0,1\}$ die Abbildung $x \mapsto z_i$ für $z \in \mathbb{B}_n$ mit $|z| = a \cdot |x|$.

Betrachten wir eine Konstante $a \in \mathbb{Z}_{2^n}$, bei der genau zwei Bits a_{n-m} und a_{n-m-d} mit $m \in \{1, \dots, n\}$ und $d \in \{1, \dots, n-m\}$ auf 1 gesetzt sind. Dann gilt für das Produkt

$$|x| \cdot |a| \;=\; |x| \cdot \left(2^{n-m} + 2^{n-m-d}\right) \;=\; 2^{n-m}|x| + 2^{n-m-d}|x|.$$

Mit Bemerkung 3.4 erhält man nun

$$\mathrm{MUL}^a_{n-1,n}(x) \;=\; \mathrm{ADD}_{n-1,n}(x_{m-1} \dots x_0\, 0^{n-m},\, x_{m+d-1} \dots x_0\, 0^{n-m-d}).$$

Da die letzten 0en eines Summanden keinen Übertrag erzeugen können, ändert sich das Ausgabebit dieser Addition nicht, wenn man die letzten $n-m$ Stellen bei jedem Summanden streicht. Wir erhalten somit

$$\mathrm{MUL}^a_{n-1,n}(x) \;=\; \mathrm{ADD}_{m-1,m}(x_{m-1} \dots x_0,\; x_{m+d-1} \dots x_d).$$

Wählt man nun $d \geq m$, so sind die Variablenmengen $\{x_{m-1}, \ldots, x_0\}$ und $\{x_{m+d-1}, \ldots, x_d\}$ disjunkt. Das bedeutet, dass man aus einem π-OBDD für $\mathrm{MUL}_{n-1,n}$ nur durch Konstantsetzen von Variablen ein entsprechendes OBDD für $\mathrm{ADD}_{m-1,m}$ über der Variablenmenge $\{x_{m-1}, \ldots, x_0, x_{m+d-1}, \ldots, x_d\}$ erhält. Nun hat zwar die Funktion $\mathrm{ADD}_{m-1,m}$ für eine gute Variablenordnung π' ein kleines π'-OBDD, aber aus Theorem 2.7 folgt, dass dies nicht für jede Variablenordnung gilt. Betrachten wir nun eine beliebige Variablenordnung π über die Eingabevariablen für $\mathrm{MUL}^a_{n-1,n}$. Bryants Trick besteht darin, für diese fest vorgegebene Variablenordnung die Parameter m und d so zu wählen, dass es eine möglichst große Menge I von Indizes aus $\{0, \ldots, m-1\}$ gibt, sodass für alle $i \in I$ entweder $\pi(x_i) \leq n/2 < \pi(x_{i+d})$ oder $\pi(x_{i+d}) \leq n/2 < \pi(x_i)$ gilt. Denn dann hat nach Theorem 2.7 das OBDD mit der entsprechenden Variablenordnung für die Addition der disjunkten Bitstrings $x_{m-1} \ldots x_0$ und $x_{m+d-1} \ldots x_0$, also auch das π-OBDD für $\mathrm{MUL}^a_{n-1,n}$, eine Größe von $\Omega\big(2^{|I|}\big)$. Mit einer probabilistischen Wahl von m und d konnte er beweisen, dass diese Parameter für jede Variablenordnung π so festgelegt werden können, dass die Menge I mindestens $n/8$ Elemente enthält. Zusammen mit obiger Reduktion auf die Addition folgt auf diese Weise die untere Schranke von $2^{n/8}$ für die OBDD-Größe von $\mathrm{MUL}_{n-1,n}$.

Die neue Beweisidee

Wie eben dargestellt hat Bryant eine untere Schranke für die π-OBDD-Größe der Funktion $\mathrm{MUL}^a_{n-1,n}$ bewiesen, wobei a in Abhängigkeit von der Variablenordnung π gewählt wurde. Allerdings ist die Auswahl der möglichen Werte für a sehr eingeschränkt, da die Reduktion auf die Addition zweier Zahlen nur so einfach funktioniert, wenn genau zwei Bits auf 1 gesetzt sind. Wenn man mehrere Bits auf 1 setzt, z.B. beliebig viele, so erhält man Additionen von vielen Zahlen und das Ganze wird recht unübersichtlich. Das Problem ist also die bitorientierte Sichtweise auf die Multiplikation, bei der man das Produkt zweier Zahlen x und y als Summe aller $x \cdot 2^i$ auffasst, für die das i-te Bit von y gesetzt ist. Wir wollen nun auch eine Reduktion auf die Addition vornehmen, dabei aber viel mehr mögliche Werte von a in Betracht ziehen, indem wir von der bitorientierten Sichtweise zu einer mengenorientierten Sichtweise übergehen.

Sei π eine Variablenordnung über $\{x_0, \ldots, x_{n-1}\}$ und $X^- \subseteq \{x_0, \ldots, x_{n-1}\}$ die erste Hälfte der Variablen bzgl. π, also die Menge der Variablen x_i mit $\pi(x_i) \leq n/2$, und sei $X^+ = \{x_0, \ldots, x_{n-1}\} - X^-$. Dann können wir eine Eingabe x als Summe $x^- + x^+$ schreiben, wobei x^- die Zahl ist, bei der nur die Bits aus X^- entsprechend x gesetzt und alle anderen Bits 0 sind, und x^+ analog für die Variablen aus X^+ definiert ist. Jetzt können wir das Produkt $a \cdot |x|$ als Summe $a \cdot x^- + a \cdot x^+$ schreiben. Da wir uns nur für das mittlere Ausgabebit interessieren, genügt es modulo 2^n zu rechnen (vgl. Bemerkung 3.4). Sind also $z, z' \in \mathbb{B}_n$ mit $|z| = \big(a \cdot x^-\big) \bmod 2^n$ und $|z'| = \big(a \cdot x^+\big) \bmod 2^n$, so gilt

$$\mathrm{MUL}^a_{n-1,n}(x) \;=\; \mathrm{ADD}_{n-1,n}(z, z').$$

Ein π-OBDD für $\mathrm{MUL}_{n-1,n}$ muss also in der Lage sein, das Ausgabebit an der Postion $n-1$ der Summe zweier Bitstrings z und z' zu berechnen, wobei z durch die in der ersten Hälfte gelesenen Variablen eindeutig bestimmt ist, und z' durch die in der zweiten Hälfte gelesenen Variablen.

Natürlich können z und z' nicht alle möglichen $(n-1)$-Bit Werte annehmen, sie sind ja bei vorgegebenem a nur durch die Belegungen der jeweils $n/2$ Variablen in X^- und X^+ bestimmt. Wenn wir aber a so wählen können, dass die Strings z und z' in den m Bits an den

Positionen $n-1, \dots, n-m$ alle möglichen m-Bit Werte annehmen, so ist die Addition von z und z' vermutlich ähnlich schwierig wie die Addition zweier m-Bit Zahlen, bei denen eine Zahl in der ersten Hälfte des OBDDs und die andere Zahl in der zweiten Hälfte des OBDDs gelesen wird. Wir vernachlässigen dabei nur den Übertrag, der bei der Addition von z und z' an der Position $n-m-1$ entsteht; man kann aber vermuten, dass er für genügend großes m nur in den wenigsten Fällen Einfluss auf das Ausgabebit an der Position $n-1$ hat. Wenn diese Überlegungen stimmen, können wir mit Theorem 2.7 auf eine untere Schranke in der Größenordnung von 2^m hoffen, da unsere Konstruktion einem OBDD mit einer Variablenordnung π' für die Funktion $\mathrm{ADD}_{m-1,m}$ entspricht, bei der die m Bits des einen Summanden in π' vor allen m Bits des anderen Summanden kommen. Im späteren Beweis wird sich herausstellen, dass die Addition von z und z' sogar schon dann schwierig ist, wenn die Anzahl der möglichen m-Bit Werte, die von den Bits $z_{n-1} \dots z_{n-m}$ und $z'_{n-1} \dots z_{n-m}$ angenommen werden, einen Anteil von $(1/2+\epsilon)$, $\epsilon > 0$, aller möglichen m-Bit Werte übersteigt.

Das Überdeckungslemma für universelle Hashklassen

Wir müssen uns also mit der Frage beschäftigen, wie wir a so wählen können, dass man für alle $2^{n/2}$ Möglichkeiten von x^- und alle $2^{n/2}$ Möglichkeiten von x^+ die Produkte $a \cdot x^-$ und $a \cdot x^+$ in den Bits $n-1, \dots, n-m$ möglichst viele Werte annehmen. Hier kommt der Zusammenhang zu den Hashklassen aus Definition 3.9 ins Spiel, denn diese m Bits sind genau die Funktionswerte $h^m_{a,0}(x^+)$ und $h^m_{a,0}(x^-)$.

Wir untersuchen daher zunächst die Frage, wieviele Werte des Wertebereichs die Hashfunktionswerte $h(x)$ mit $x \in M_1$ bzw. $x \in M_2$ für zwei Mengen M_1, M_2, abdecken können, wenn h eine geeignet gewählte Hashfunktion einer universellen Hashklasse ist. Da universelle Hashklassen die Eigenschaft haben, dass zwei beliebige Schlüssel unter einer zufällig gewählten Hashfunktion nur mit kleiner Wahrscheinlichkeit kollidieren, darf es für die meisten Hashfunktionen unter allen Schlüsseln nur wenige Kollisionen geben. D.h. aber, dass die Schlüssel durch die Hashfunktionen auf viele verschiedene Hashwerte verstreut werden müssen.

Für eine Funktion f und eine Teilmenge M des Definitionsbereichs von f bezeichne $f(M)$ das *Bild* von M unter f, also die Menge $\{f(x) \mid x \in M\}$.

Lemma 4.2 *Sei $\mathcal{H}$ eine universelle Hashklasse mit Universum U und Wertebereich R. Für alle $0 \le \epsilon < 1$ und $M, M' \subseteq U$ mit*

$$|M|, |M'| > \frac{2\epsilon}{1-\epsilon} \cdot (|R|-1)$$

gibt es eine Hashfunktion $h \in \mathcal{H}$, sodass $h(M)$ und $h(M')$ jeweils mehr als $\epsilon \cdot |R|$ Elemente enthalten.

Beweis: Sei $r = |R|$ und o.B.d.A. haben M und M' die gleiche Kardinalität k mit $k > 2\epsilon(r-1)/(1-\epsilon)$. Für $h \in \mathcal{H}$ und eine Menge $A \subseteq U$ bezeichne $\delta_h(A)$ die Anzahl der Schlüsselpaare aus A, die unter der Hashfunktion h kollidieren, also

$$\delta_h(A) = \left|\{(x, x') \in A \times A \mid x \neq x' \wedge h(x) = h(x')\}\right|.$$

Da $\mathcal{H}$ universell ist, erhalten wir folgende Abschätzung für den Erwartungswert von $\delta_h(A)$

für eine zufällig aus $\mathcal{H}$ gewählte Hashfunktion h:

$$\mathrm{E}\left[\delta_h(A)\right] \;=\; \sum_{\substack{x,x'\in A\\ x\neq x'}} \mathbf{Prob}\left(h(x)=h(x')\right) \;\leq\; \frac{|A|\cdot(|A|-1)}{r}. \tag{4.1}$$

Mit der Linearität des Erwartungswerts folgt somit für die Zufallsvariable $\delta_h(M)+\delta_h(M')$ ein Erwartungswert von $(2/r)\cdot k(k-1)$. Gemäß der probabilistischen Methode gibt es dann ein $h_0\in\mathcal{H}$, für das gilt

$$\delta_{h_0}(M)+\delta_{h_0}(M') \;\leq\; \frac{2}{r}\cdot k\cdot(k-1). \tag{4.2}$$

Wir zeigen nun, dass h_0 eine Hashfunktion ist, die die Behauptung erfüllt, d.h. dass $|h_0(M)|, |h_0(M')| > \epsilon r$ gilt. Angenommen, dies ist nicht der Fall, und zwar sei o.B.d.A. $h_0(M)=\ell\leq\epsilon r$. Offensichtlich ist $\delta_{h_0}(M)$ gleich der Summe aller Paare verschiedener Elemente aus $h_0^{-1}(y)\cap M$ über alle $y\in h_0(M)$, also

$$\begin{aligned}
\delta_{h_0}(M) &= \sum_{y\in h_0(M)} |h_0^{-1}(y)\cap M|\cdot\left(|h_0^{-1}(y)\cap M|-1\right)\\
&= \sum_{y\in h_0(M)} |h_0^{-1}(y)\cap M|^2 - \sum_{y\in h_0(M)} |h_0^{-1}(y)\cap M|\\
&= -k+\sum_{y\in h_0(M)} |h_0^{-1}(y)\cap M|^2.
\end{aligned}$$

Die letzte Summe besteht aus genau ℓ Summanden, und es gilt $\sum_{y\in h_0(M)}|h_0^{-1}(y)\cap M| = |M| = k$. Da die reelle Funktion $(z_1,\ldots,z_\ell)\mapsto {z_1}^2+\ldots+{z_\ell}^2$ unter der Nebenbedingung $z_1+\ldots+z_\ell=k$ ihr Minimum für $z_1=\ldots=z_\ell=k/\ell$ annimmt, folgt

$$\delta_{h_0}(M) \;\geq\; -k+\ell\cdot(k/\ell)^2 \;\geq\; k\cdot\left(\frac{k}{\epsilon r}-1\right).$$

Trivialerweise gilt $|h_0(M')|\leq r$ und man erhält, in dem man in obiger Rechnung ϵr durch r substituiert, die Ungleichung $\delta_{h_0}(M')\geq k(k/r-1)$. Zusammen mit Ungleichung (4.2) folgt somit

$$\begin{aligned}
& k\cdot\left(\frac{k}{\epsilon r}+\frac{k}{r}-2\right) \;\leq\; \delta_{h_0}(M)+\delta_{h_0}(M') \;\leq\; \frac{2}{r}\cdot k\cdot(k-1)\\
\Rightarrow\;& k\left(\frac{1}{\epsilon r}-\frac{1}{r}\right) \;\leq\; 2-\frac{2}{r}\\
\Rightarrow\;& k \;\leq\; \frac{2(1-1/r)}{(1/\epsilon-1)/r} \;=\; \frac{2(r-1)}{1/\epsilon-1} \;=\; \frac{2\epsilon(r-1)}{1-\epsilon}.
\end{aligned}$$

Dies ist ein Widerspruch zur Voraussetzung. ■

Das Überdeckungslemma für die Multiplikation

Wenn wir das soeben bewiesene Überdeckungslemma auf die multiplikative Hashklasse anwenden, sehen wir, dass es zu je zwei genügend großen Mengen $M, M'\subseteq\mathbb{Z}_{2^n}$ zwei Parameter $a\in\mathbb{Z}_{2^n}$ und $b\in\mathbb{Z}_{2^{n-m}}$ gibt, sodass die Hashwerte $ax+b$ mit x aus M bzw. aus M' in den

Bits $n-1,\dots,n-m$ viele verschiedene Werte abdecken. Wir können dieses Ergebnis aber noch nicht direkt auf die weiter oben beschriebene Idee für den Beweis der unteren Schranke bei der Multiplikation anwenden, da wir dort nur die Multiplikation ohne zusätzliche Addition mit b betrachten. Da b aber nur aus $n-m$ Bits besteht, hat dieser additive Term keinen großen Einfluss auf die m Bits, für die wir uns interessieren.

Insbesondere wenn sich z.B. $ax+b$ und $ax'+b$ in den Bitpositionen $n-m,\dots,n-1$ unterscheiden und zusätzlich das $(n-m-1)$-te Bit beider Zahlen 1 ist, so unterscheiden sich auch ax und ax' in den Bitpositionen $n-m,\dots,n-1$. Diese Beobachtung führt zu folgendem Überdeckungslemma für die Multiplikation.

Lemma 4.3 *Seien $1/2 \le \epsilon < 1$ und $M, M' \subseteq \mathbb{Z}_{2^n}$ mit*

$$|M|, |M'| \; > \; \frac{2\epsilon}{1-\epsilon} \cdot \bigl(2^{m+1}-1\bigr).$$

Dann gibt es ein $a \in \mathbb{Z}^*_{2^n}$ sodass $h^m_{a,0}(M)$ und $h^m_{a,0}(M')$ jeweils mindestens $(2\epsilon-1)\cdot 2^m$ Elemente enthalten.

Beweis: Da die multiplikative Hashklasse $\mathcal{M}_{n,m+1}$ universell ist (s. Theorem 3.10) gibt es gemäß Lemma 4.2 ein $a \in \mathbb{Z}^*_{2^n}$ und ein $b \in \bigl\{0,\dots,2^{n-m-1}\bigr\}$, sodass $h^{m+1}_{a,b}(M)$ und $h^{m+1}_{a,b}(M')$ jeweils mehr als $\epsilon \cdot |\mathbb{Z}_{2^{m+1}}| = \epsilon \cdot 2^{m+1}$ Elemente enthalten. Wir zeigen für dieses a und $h = h^m_{a,0}$, dass $h(M)$ mindestens $(2\epsilon-1)2^m$ Elemente enthält; für M' folgt die Behauptung analog.

Da es in $\mathbb{Z}_{2^{m+1}}$ genau 2^m gerade Zahlen gibt, befinden sich in $h^{m+1}_{a,b}(M)$ mindestens

$$|h^{m+1}_{a,b}(M)| - 2^m \;\ge\; \epsilon 2^{m+1} - 2^m \;=\; (2\epsilon-1)\cdot 2^m$$

ungerade Elemente. Es genügt daher zu zeigen, dass für x, x' mit ungeraden aber verschiedenen Funktionswerten $h^{m+1}_{a,b}(x)$ und $h^{m+1}_{a,b}(x')$ auch die Funktionswerte $h(x)$ und $h(x')$ verschieden sind. Seien also $y = h^{m+1}_{a,b}(x) \ne y = h^{m+1}_{a,b}(x')$ mit y, y' ungerade. Es gilt

$$y \cdot 2^{n-m-1} \;\le\; (ax+b) \bmod 2^n \;<\; (y+1)\cdot 2^{n-m-1}.$$

Da $b \in \mathbb{Z}_{2^{n-m-1}}$ ist, folgt

$$(y-1)\cdot 2^{n-m-1} \;<\; (ax+b) \bmod 2^n - b \;<\; (y+1)\cdot 2^{n-m-1}.$$

Da y ungerade (also mindestens 1) ist, können wir den mittleren Ausdruck modulo 2^n nehmen, ohne dass sich an den Ungleichungen etwas ändert. Wir erhalten somit

$$\frac{(y-1)}{2}\cdot 2^{n-m} \;<\; (ax) \bmod 2^n \;<\; \frac{(y+1)}{2}\cdot 2^{n-m}.$$

Da y ungerade ist, gilt $(y-1)/2 = \lfloor y/2 \rfloor$ und $(y+1)/2 = \lfloor y/2 \rfloor + 1$. Es folgt

$$\lfloor y/2 \rfloor \cdot 2^{n-m} \;<\; (ax) \bmod 2^n \;<\; \bigl(\lfloor y/2 \rfloor + 1\bigr)\cdot 2^{n-m}$$

und somit

$$h^m_{a,0}(x) \;=\; \bigl((ax) \bmod 2^n\bigr) \operatorname{div} 2^{n-m} \;=\; \lfloor y/2 \rfloor.$$

Indem wir y durch y' substituieren, folgt völlig analog $h^m_{a,0}(x') = \lfloor y'/2 \rfloor$. Da aber y und y' verschiedene, ungerade Zahlen sind, sind auch $\lfloor y/2 \rfloor$ und $\lfloor y'/2 \rfloor$ verschieden, und die Behauptung $h^m_{a,0}(x) \ne h^m_{a,0}(x')$ ist bewiesen. ∎

Die untere Schranke

Mithilfe des Überdeckungslemmas für die Multiplikation können wir nun die eingangs beschriebene Idee konkretisieren und folgende untere Schranke für die π-OBDD-Größe von $\mathrm{MUL}^a_{n-1,n}$ beweisen.

Theorem 4.4 *Für jede Variablenordnung π über $X = \{x_0, \ldots, x_{n-1}\}$ gibt es ein $a \in \mathbb{Z}^*_{2^n}$, sodass jedes π-OBDD für $\mathrm{MUL}^a_{n-1,n}$ aus mehr als $2^{\lfloor n/2 \rfloor}/121 - 1$ Knoten besteht.*

Um das Theorem zu beweisen, benutzen wir eine allgemein bekannte Technik, die eine Beziehung zwischen der Anzahl der Subfunktionen der zu betrachtenden Funktion und der OBDD-Größe herstellt. Wir benötigen dazu folgende Notation.

Definition 4.5

(a) Eine *partielle Belegung* einer Variablenmenge $X = \{x_1, \ldots, x_n\}$ ist ein String $\alpha = \alpha_1 \ldots \alpha_n \in \{0, 1, *\}^n$, wobei ein Wert von $\alpha_i \in \{0, 1\}$ bedeutet, dass die Variable x_i mit 0 bzw. 1 belegt ist, und $\alpha_i = *$ bedeutet, dass die Variable x_i *frei* ist. Die *Trägermenge* einer partiellen Belegung α ist die Menge der nicht freien Variablen aus X und wird mit $\mathcal{T}(\alpha)$ bezeichnet. Die Menge partieller Belegungen von X mit Trägermenge $X' \subseteq X$ bezeichnen wir mit $\mathcal{P}(X, X')$.

(b) Sind $\alpha = \alpha_1 \ldots \alpha_n$ und $\beta = \beta_1 \ldots \beta_n$ zwei partielle Belegungen von X mit disjunkten Trägermengen, so ist $\alpha\beta$ gleich der partiellen Belegung $\gamma = \gamma_1 \ldots \gamma_n$ mit

$$\gamma_i = \begin{cases} \alpha_i & \text{falls } \alpha_i \neq *, \\ \beta_i & \text{falls } \beta_i \neq * \quad \text{und} \\ * & \text{sonst.} \end{cases}$$

(c) Für eine Funktion $f \in B_n$ über X und eine partielle Belegung α ist $f_{|\alpha}$ die *Subfunktion von f für α*, definiert durch

$$x_1 \ldots x_n \mapsto f(z_1 \ldots z_n) \quad \text{mit} \quad z_i = \begin{cases} x_i & \text{falls } \alpha_i = * \text{ und} \\ \alpha_i & \text{sonst.} \end{cases}$$

(d) Zu einem Branchingprogramm über X und einer partiellen Belegung α von X ist der *durch α induzierte partielle Berechnungspfad* derjenige eindeutig bestimmte Pfad in G, der an der Wurzel beginnt, jeden mit $x_i \in \mathcal{T}(\alpha)$ markierten Knoten über die mit α_i markierte Kante verlässt, und an dem ersten Knoten endet, dessen Markierung $x_j \in X$ nicht in $\mathcal{T}(\alpha)$ ist.

(e) Ist $\alpha = \alpha_{n-1} \ldots \alpha_0$ eine partielle Belegung von $\{x_0, \ldots, x_{n-1}\}$, so bezeichnen wir mit $|\alpha|$ den ganzzahligen Wert $|z| \in \mathbb{Z}_{2^n}$ für $z \in \mathbb{B}_n$ mit $z_i = \alpha_i$ falls $\alpha_i \in \mathcal{T}(\alpha)$ und $z_i = 0$ falls $\alpha_i = *$.

Bemerkung 4.6 *Offensichtlich gilt für zwei partielle Belegungen α, β von $\{x_0, \ldots, x_{n-1}\}$ mit disjunkten Trägermengen $|\alpha\beta| = |\alpha| + |\beta|$.*

Die folgende Aussage ist allgemein bekannt – sie folgt z.B. aus dem Struktursatz von Sieling und Wegener (1993).

Lemma 4.7 *Sei $f \in B_n$ und π eine Variablenordnung über $X = \{x_1, \dots, x_n\}$. Weiterhin sei X^- für ein $k \in \{1, \dots, n\}$ die Menge der ersten k Variablen bzgl. der Variablenordnung π, d.h. $X^- = \{x_i \in X \mid \pi(x_i) \leq k\}$, und s_k sei die Anzahl verschiedener Subfunktionen $f_{|\alpha}$ mit $\alpha \in \mathcal{P}(X, X^-)$. Dann hat jedes π-OBDD für f mindestens $2s_k - 1$ Knoten.*

Beweis: Sei G ein beliebiges π-OBDD für f. Weiterhin sei v_α zu einer partiellen Belegung α von X der Knoten, auf dem der durch α induzierte partielle Berechnungspfad endet, und V sei die Menge aller Knoten v_α mit $\alpha \in P(X, X^-)$. Da in der Variablenordnung π kein Knoten aus X^- nach einem Knoten aus $X - X^-$ vorkommt, enthält jeder Pfad von $v_\alpha \in V$ zu einer Senke nur Variablen aus $X - X^-$. Also wird durch jede partielle Belegung $\beta \in \mathcal{P}(X, X - X^-)$ ein eindeutiger Pfad von v_α zu einer Senke bestimmt.

Seien nun α, α' zwei verschiedene partielle Belegungen aus $\mathcal{P}(X, X^-)$. Falls $v_\alpha = v_{\alpha'}$ ist, so gilt $f(\alpha\beta) = f(\alpha'\beta)$ für jede partielle Belegung β mit $T(\beta) = X - X^-$, da offenbar die Berechnungspfade für $\alpha\beta$ und $\alpha'\beta$ die gleiche Senke erreichen, und es folgt $f_{|\alpha} = f_{|\alpha'}$. Also gilt $|V| \geq s_k$. Außerdem ist kein Knoten $v_\alpha \in V$ ein innerer Knoten eines anderen partiellen Berechnungpfads, der durch eine partielle Belegung $\alpha' \in \mathcal{P}(X, X^-)$ definiert wird, weil v_α sonst – im Widerspruch zur Definition des durch α induzierten Berechnungspfads – mit einer Variablen aus X^- markiert sein müsste. Somit lässt sich in den Subgraphen von G, der aus den Pfaden von der Wurzel zu den Knoten aus V besteht, ein ungerichteter Baum mit $|V|$ Blättern einbetten, bei dem jeder innere Knoten höchstens 2 Kinder hat. So ein Baum hat insgesamt mindestens $2|V| - 1$ Knoten und aus $|V| \geq s_k$ folgt die Behauptung $|G| \geq 2s_k - 1$. ∎

Um zu beweisen, dass zwei Subfunktionen $f_{|\alpha}, f_{|\alpha'}$ verschieden sind, benötigt man einen Zeugen, d.h. eine Belegung β der noch freien Variablen, für die $f(\alpha\beta) \neq f(\alpha'\beta)$ ist. Das folgende Lemma hilft bei der Konstruktion solcher Zeugen für die Multiplikation. Es wird hier bewusst sehr allgemein formuliert, da wir es so auch in späteren Beweisen verwenden können.

Lemma 4.8 *Sei $2 \leq m \leq n$ und $h = h^m_{a,0}$ eine Funktion der multiplikativen Hashklasse $\mathcal{M}_{n,m}$ mit $a \in \mathbb{Z}^*_{2^n}$. Dann gilt für alle $x, x' \in \mathbb{Z}_{2^n}$ mit*

$$2 \leq \big(h(x) - h(x')\big) \bmod 2^m \leq 2^{m-1}$$

und alle $z \in \mathbb{Z}_{2^n}$

(a) $\big(h(x) + h(z)\big) \bmod 2^m = 0 \quad \Rightarrow \quad \langle a(x+z)\rangle^{n-1}_{n-1} < \langle a(x'+z)\rangle^{n-1}_{n-1}.$

(b) $\big(h(x) + h(z)\big) \bmod 2^m = 2^{m-1} \quad \Rightarrow \quad \langle a(x+z)\rangle^{n-1}_{n-1} > \langle a(x'+z)\rangle^{n-1}_{n-1}.$

(c) $\big(h(x') + h(z)\big) \bmod 2^m = 2^{m-1} - 2 \quad \Rightarrow \quad \langle a(x+z)\rangle^{n-1}_{n-1} > \langle a(x'+z)\rangle^{n-1}_{n-1}.$

(d) $\big(h(x') + h(z)\big) \bmod 2^m = 2^m - 2 \quad \Rightarrow \quad \langle a(x+z)\rangle^{n-1}_{n-1} < \langle a(x'+z)\rangle^{n-1}_{n-1}.$

Beweis: Wir beweisen Aussage (a), also den Fall $\big(h(x) + h(z)\big) \bmod 2^m = 0$. Die Aussagen (b)-(d) folgen mit völlig analogen Rechnungen, die wir im Anhang (S. 69f.) wiedergeben.

Sei $y = h(x) = \big((ax) \bmod 2^n\big) \operatorname{div} 2^{n-m}$ und $\delta = (ax) \bmod 2^{n-m}$. Analog seien y' und δ' für x' definiert. Schließlich sei $k = h(z)$ und $\tau = (az) \bmod 2^{n-m}$. Offenbar sind

$\delta, \delta', \tau \in \mathbb{Z}_{2^{n-m}}$ und es gilt

$$(ax) \bmod 2^n = y \cdot 2^{n-m} + \delta, \tag{4.3}$$

$$(ax') \bmod 2^n = y' \cdot 2^{n-m} + \delta' \quad \text{und} \tag{4.4}$$

$$(az) \bmod 2^n = k \cdot 2^{n-m} + \tau. \tag{4.5}$$

Wir berechnen zunächst das Bit $\langle a(x+z)\rangle_{n-1}^{n-1}$. Aus den Gleichungen (4.3) und (4.5) sowie der Voraussetzung $(y+k) \bmod 2^m = 0$ folgt

$$ax + az \equiv \big(y+k\big) \cdot 2^{n-m} + \delta + \tau \equiv \delta + \tau \pmod{2^n}.$$

Da $\delta + \tau \in \mathbb{Z}_{2^{n-m}}$ ist, erhalten wir mit $m \geq 2$

$$(ax + az) \bmod 2^n < 2 \cdot 2^{n-m} \leq 2^{n-1}.$$

Somit ist das Bit an der Position $n-1$ von $a(x+z)$ nicht gesetzt, also $\langle a(x+z)\rangle_{n-1}^{n-1} = 0$.

Es genügt schließlich, $\langle a(x'+z)\rangle_{n-1}^{n-1} = 1$ zu zeigen. Wir erhalten, indem wir die Gleichungen (4.4) und (4.5) sowie $(y+k) \bmod 2^m = 0$ benutzen,

$$\begin{aligned} ax' + az &\equiv (y'+k) \cdot 2^{n-m} + \delta' + \tau \\ &\equiv \big(y + k - (y - y')\big) \cdot 2^{n-m} + \delta' + \tau \\ &\equiv (y' - y) \cdot 2^{n-m} + \delta' + \tau \pmod{2^n}. \end{aligned} \tag{4.6}$$

Nach Voraussetzung ist $2 \leq (y - y') \bmod 2^m \leq 2^{m-1}$, woraus

$$2^{n-1} \leq \big((y'-y) \cdot 2^{n-m}\big) \bmod 2^n \leq 2^n - 2 \cdot 2^{n-m}$$

folgt. Wir setzen dies in (4.6) ein und erhalten mit $\delta' + \tau \in \big\{0, \ldots, 2 \cdot (2^{n-m} - 1)\big\}$

$$2^{n-1} \leq (ax' + az) \bmod 2^n \leq 2^n - 2.$$

Also ist wie behauptet $\langle a(x'+z)\rangle_{n-1}^{n-1} = 1$. ■

Nach diesen Vorbereitungen können wir nun die untere Schranke für die π-OBDD-Größe von $\mathrm{MUL}_{n-1,n}^a$ (für ein geeignetes a) beweisen.

Beweis von Theorem 4.4: Es genügt, für gerades n eine untere Schranke von $2^{n/2}/121 - 1$ zu zeigen. Dazu teilen wir die Variablen aus X bzgl. π in die erste Hälfte $X^- = \{x \in X \mid \pi(x) \leq n/2\}$ und die zweite Hälfte $X^+ = X - X^-$ ein. Sei $A = \mathcal{P}(X, X^-)$ und $B = \mathcal{P}(X, X^+)$. Da X^- und X^+ eine Partition von X darstellen, gibt es zu jeder Eingabe $x \in \mathbb{B}_n$ für $\mathrm{MUL}_{n-1,n}^a$ zwei eindeutig bestimmte $\alpha \in A$ und $\beta \in B$ mit $x = \alpha\beta$.

Unser Ziel ist nun, eine geeignete Konstante $a \in \mathbb{Z}_{2^n}$ sowie eine möglichst große Teilmenge $A' \subseteq A$ zu finden, sodass alle Subfunktionen $\big(\mathrm{MUL}_{n-1,n}^a\big)_{|\alpha}$ mit $\alpha \in A'$ verschieden sind. D.h., es soll gelten

$$\forall \alpha, \alpha' \in A', \alpha \neq \alpha' \quad \exists \beta \in B: \quad \mathrm{MUL}_{n-1,n}^a(\alpha\beta) \neq \mathrm{MUL}_{n-1,n}^a(\alpha'\beta). \tag{4.7}$$

Aus Lemma 4.7 folgt dann eine untere Schranke von $2|A'| - 1$ für die π-OBDD-Größe von $\mathrm{MUL}_{n-1,n}$.

Sei $M_A = \{|\alpha| \,|\, \alpha \in A\}$ und $M_B = \{|\beta| \,|\, \beta \in B\}$. Dann ist $|M_A| = |M_B| = |A| = |B| = 2^{n/2}$. Weiterhin seien $\epsilon = 16/17$ und $m = n/2 - 6$, sodass gilt

$$\frac{2\epsilon}{1-\epsilon} \cdot \left(2^{m+1} - 1\right) = 32 \cdot \left(2^{n/2-5} - 1\right) = 2^{n/2} - 32 < 2^{n/2} = |M_A| = |M_B|.$$

Wir können somit Lemma 4.3 anwenden, wonach ein $a \in \mathbb{Z}_{2^n}^*$ existiert, sodass $h_{a,0}^m(M_A)$ und $h_{a,0}(M_B)$ jeweils mindestens $(2\epsilon - 1)2^m = (15/17) \cdot 2^m$ Elemente enthalten. Sei im Folgenden $h = h_{a,0}^m$ für dieses a.

Da $h(M_A)$ und $h(M_B)$ Teilmengen von $\mathbb{Z}_{2^m}$ sind und es in $\mathbb{Z}_{2^m}$ nur 2^{m-1} ungerade Elemente gibt, können wir zwei minimale Teilmengen $M_A' \subseteq M_A$ und $M_B' \subseteq M_B$ wählen, sodass $h(M_A')$ und $h(M_B')$ jeweils genau

$$(15/17) \cdot 2^m - 2^{m-1} = (30/17) \cdot 2^{m-1} - 2^{m-1} = (13/17) \cdot 2^{m-1}$$

gerade Elemente enthalten. Da wir M_A' als minimale Teilmenge mit der gewünschten Eigenschaft gewählt haben, ist zudem h injektiv auf M_A'.

Sei nun $B' \subseteq B$ die Menge der partiellen Belegungen β mit $|\beta| \in M_B'$ und

$$A' = \left\{\alpha \in A \;\middle|\; |\alpha| \in M_A' \wedge \exists \beta \in B' : h(|\alpha|) + h(|\beta|) = 2^m\right\}. \tag{4.8}$$

Wir können nun für A' mit Lemma 4.8 ganz einfach Aussage (4.7) beweisen: Seien α und α' verschiedene Elemente aus A'. O.B.d.A. ist

$$\left(h(|\alpha|) - h(|\alpha'|)\right) \bmod 2^m \leq 2^{m-1},$$

denn wenn dies nicht gilt, vertauschen wir α und α'. Zudem ist $h(|\alpha|) \neq h(|\alpha'|)$ (weil h injektiv auf M_A' ist) und da beide Funktionswerte nach Konstruktion von M_A' gerade sind, beträgt der Betrag der Differenz dieser Funktionswerte mindestens 2. Also folgt

$$2 \leq \left(h(|\alpha|) - h(|\alpha'|)\right) \bmod 2^m \leq 2^{m-1}.$$

Schließlich gibt es nach Konstruktion von A' ein $\beta \in B'$ mit $h(|\alpha|) + h(|\beta|) = 2^m$ und die Voraussetzungen für Lemma 4.8 sind (für $x = |\alpha|$, $x' = |\alpha'|$ und $z = |\beta|$) erfüllt. Die Anwendung dieses Lemmas zeigt somit, dass $\langle a(|\alpha| + |\beta|)\rangle_{n-1}^{n-1}$ und $\langle a(|\alpha'| + |\beta|)\rangle_{n-1}^{n-1}$ verschieden sind, also ist

$$\mathrm{MUL}_{n-1,n}^a(\alpha\beta) \neq \mathrm{MUL}_{n-1,n}^a(\alpha'\beta).$$

Insgesamt haben wir (4.7) für die Menge A' bewiesen, und damit auch eine untere Schranke von $2|A'| - 1$ für die π-OBDD-Größe von $\mathrm{MUL}_{n-1,n}^a$.

Wir müssen also nur noch die Kardinalität von A' abschätzen. Es sei daran erinnert, dass $h(M_A')$ und $h(M_B')$ jeweils genau $(13/17) \cdot 2^{m-1}$ gerade Elemente enthalten und dass h injektiv auf M_A' ist. Da für jedes gerade $z \in \mathbb{Z}_{2^m}$ auch $2^m - z$ gerade ist, enthält die Menge $Y = \left\{2^m - h(b) \;\middle|\; b \in M_B'\right\}$ auch genau $(13/17) \cdot 2^{m-1}$ gerade Elemente. Nun ist $|A'|$ wegen der Injektivität von h auf M_A' aber aber nach der Definition aus (4.8) genau die Kardinalität des Schnitts $Y \cap h(M_A')$. Sei Z die Menge der 2^{m-1} geraden Elemente aus $\mathbb{Z}_{2^m}$. Es folgt

$$\begin{aligned}
|A'| &= \left|Y \cap h(M_A')\right| = 2^{m-1} - \left|Z - \left(Y \cap h(M_A')\right)\right| \\
&= 2^{m-1} - \left|\left(Z - Y\right) \cup \left(Z - h(M_A')\right)\right| \\
&\geq 2^{m-1} - \left(2^{m-1} - |Y| + 2^{m-1} - \left|M_A'\right|\right) \\
&= 2 \cdot (13/17) \cdot 2^{m-1} - 2^{m-1} = (9/17) \cdot 2^{m-1}.
\end{aligned}$$

Wir haben zu Beginn $m = n/2 - 6$ gewählt und erhalten somit als untere Schranke für die π-OBDD-Größe von $\mathrm{MUL}^a_{n-1,n}$

$$2 \cdot \left|A'\right| - 1 \;\geq\; 2 \cdot (9/17) \cdot 2^{n/2}/128 - 1 \;\geq\; \frac{9}{1088} \cdot 2^{n/2} - 1 \;>\; \frac{1}{121} \cdot 2^{n/2} - 1. \qquad \blacksquare$$

Aus der eben bewiesenen unteren Schranke für die π-OBDD-Größe von $\mathrm{MUL}^a_{n-1,n}$ erhält man natürlich die gleiche untere Schranke für die OBDD-Größe von $\mathrm{MUL}_{n-1,n}$. Aber tatsächlich kann man hier noch fast um einen Faktor 2 besser werden. Sei G ein π-OBDD für $\mathrm{MUL}_{n-1,n}$ über der Variablenmenge $X \cup Y = \{x_0, \ldots, x_{n-1}\} \cup \{y_0, \ldots, y_{n-1}\}$. Wir haben soeben bewiesen, dass wir die y-Variablen so konstant setzen können, dass das resultierende OBDD aus mindestens $C = 2^{n/2} \cdot 1/121 - 1$ Knoten besteht. Dieses OBDD enthält nur die Knoten von G, die mit x-Variablen markiert sind. Analog können wir aber genauso auch die x-Variablen konstant setzen und erhalten so einen Subgraphen, der nur noch aus mit y-Variablen markierten Knoten und den zwei Senken besteht. Also besteht G aus mindestens $C - 2$ Knoten, die mit y-Variablen markiert sind, und aus ebenso vielen Knoten, die mit x-Variablen markiert sind, sowie aus den zwei Senken.

Korollar 4.9 *Jedes* OBDD*, das die Funktion* $\mathrm{MUL}_{n-1,n}$ *darstellt, besteht aus mindestens* $2^{\lfloor n/2 \rfloor}/61 - 4$ *Knoten.*

4.2 Obere Schranken für die OBDD-Größe der Multiplikation

Wir wollen uns nun der Frage widmen, wie „gut“ die soeben gezeigte untere Schranke ist, und ob man sie ggf. mit einfachen Mitteln noch verbessern kann.

Der Beweis basiert im Wesentlichen auf dem Überdeckungslemma für universelle Hashklassen (Lemma 4.2), aus dem das Überdeckungslemma für die Multiplikation (Lemma 4.3) folgt. Letzteres beweist, dass man bei Wahl zweier Teilmengen M, M' von $\mathbb{Z}_{2^n}$ mit Kardinalität von jeweils $2^{n/2}$ einen Faktor a finden kann, sodass die Funktionswerte von M und M' unter der Abbildung $x \mapsto (ax) \bmod 2^n \operatorname{div} 2^{n/2+c}$ (für eine kleine Konstante c) jeweils fast den ganzen Wertebereich abdecken. Hier könnte man offensichtlich höchstens den konstanten Faktor noch verbessern. Tatsächlich haben wir ja sogar bewiesen, dass die Funktion $\mathrm{MUL}^a_{n-1,n}$ $\Omega\left(2^{n/2}\right)$ verschiedene Subfunktionen für partielle Belegungen hat, die die erste Hälfte der Variablen bzgl. der Variablenordnung als Trägermenge haben. Das bedeutet, dass jedes π-OBDD für $\mathrm{MUL}^a_{n-1,n}$ (für ein geeignet gewähltes a) in der oberen Hälfte fast ein binärer Baum ist. Auch hier kann man bis auf den konstanten Faktor nichts verbessern. Es ist aber nicht klar, ob die Aufteilung der Variablen in eine obere und untere Hälfte optimal ist. Vielleicht bildet das minimale π-OBDD für $\mathrm{MUL}^a_{n-1,n}$ ja sogar z.B. in den ersten $(2/3)n$ Ebenen fast einen binären Baum?

Dass dies nicht so ist, dass unsere Schranke für $\mathrm{MUL}^a_{n-1,n}$ tatsächlich asymptotisch optimal ist, beweist folgender Satz.

Theorem 4.10 *Es existiert eine Variablenordnung π für die es für jedes $a \in \mathbb{Z}_{2^n}$ ein* π-OBDD *für* $\mathrm{MUL}^a_{n-1,n}$ *mit weniger als* $3 \cdot 2^{n/2}$ *Knoten gibt.*

Beweis: Sei π die Variablenordnung über $X = \{x_0, \ldots, x_{n-1}\}$ mit

$$\left(\pi^{-1}(1), \ldots, \pi^{-1}(n)\right) \;=\; (x_0, \ldots, x_{n-1}).$$

Wir beweisen die Aussage, indem wir einen Algorithmus angeben, der die Variablen aus X in der durch π definierten Reihenfolge liest und als Arbeitsspeicher Zustände benutzt, wobei jeder Zustand einem OBDD-Knoten entspricht. D.h. nach jeder gelesenen Variable „kennt" der Algorithmus einerseits die als Nächstes zu lesende Variable x_i sowie einen Wert aus einer Menge $\{1, \ldots, k\}$, wobei k die Anzahl der mit x_i markierten OBDD-Knoten ist. Der nächste Zustand muss dann eindeutig durch die Belegung der Variablen x_i und den vorherigen Zustand bestimmt sein. Es ist offensichtlich, wie sich aus so einem Algorithmus das zugehörige π-OBDD konstruieren lässt.

Sei $m = \lfloor n/2 \rfloor$. In der ersten Hälfte liest der Algorithmus die Variablen $x_0, \ldots, x_{m-1}$ und merkt sich jeweils den Wert aller bisher gelesenen Variablen. Dafür genügen auf der x_i-Ebene 2^i Zustände und die ersten $m + 1$ Ebenen des π-OBDDs bilden einen binären Baum, dessen Blätter die 2^m mit x_m markierten Knoten sind.

Sei für $0 \leq k \leq n$

$$s_k = \left(\left(a \cdot \sum_{i=0}^{k-1} 2^i x_i \right) \bmod 2^n \right) \operatorname{div} 2^k.$$

Offensichtlich ist s_m eindeutig durch die Variablen $x_0, \ldots, x_{m-1}$ bestimmt und jedem der 2^m mit x_m markierten Knoten des bisher konstruierten Teil-OBDDs kann der zugehörige s_m-Wert zugeordnet werden. Mit anderen Worten, der Algorithmus speichert auf der Ebene der mit x_m markierten Knoten den Wert s_m. Wir zeigen nun für $k = m, \ldots, n-2$, dass s_{k+1} eindeutig durch s_k und der Belegung von x_k bestimmt ist. Also kann unser Algorithmus $s_{m+1}, \ldots, s_{n-1}$ sukzessive durch Lesen der Variablen $x_m, \ldots, x_{n-2}$ berechnen. Wir werden auch sehen, dass $\mathrm{MUL}^a_{n-1,n}(x)$ eindeutig durch s_{n-1} und der Belegung von x_{n-1} bestimmt ist. Somit kann der Algorithmus, wenn er s_{n-1} berechnet hat, durch Lesen der Variablen x_{n-1} eine korrekte Ausgabe bestimmen.

Dass s_{k+1} eindeutig durch s_k und die Belegung von x_k bestimmt ist, sieht man am einfachsten, indem man sich die Definition von s_{k+1} mithilfe „ausgeschnittener" Bits vorstellt (vgl. Bemerkung 3.4). Sei $z = z_{n-1} \ldots z_0$ mit $|z| = \left(a \cdot \sum_{i=0}^{k-1} 2^i x_i\right) \bmod 2^n$. Dann ist $s_k = |z_{n-1} \ldots z_k|$. Sei weiterhin $z' = z'_{n-1} \ldots z'_0$ mit $|z'| = \left(a \cdot x_k \cdot 2^k\right) \bmod 2^n$. Da dies ein Vielfaches von 2^k ist, sind die letzten k Bits von z' nicht gesetzt, also $z' = z'_{n-1} \ldots z'_k 0^k$. Dann kann aber bei der Addition von $|z'|$ und $|z|$ in den letzten k Bits kein Übertrag entstehen, d.h.

$$\begin{aligned} & |z_{n-1} \ \ldots z_k \ z_{k-1} \ \ldots \ z_0| \\ + \ & |z'_{n-1} \ \ldots z'_k \ \ 0 \ \ \ldots 0 \ | \\ \equiv \ & |q_{n-1} \ \ldots q_k \ z_{k-1} \ \ldots \ z_0| \quad (\bmod 2^n), \end{aligned} \tag{4.9}$$

wobei

$$s_{k+1} = |q_{n-1} \ldots q_{k+1}| = \left((|z_{n-1} \ldots z_k| + |z'_{n-1} \ldots z'_k|) \bmod 2^n \right) \operatorname{div} 2 \tag{4.10}$$

ist. Also ist s_{k+1} eindeutig durch s_k und z' bestimmt, wobei z' nach Konstruktion eindeutig durch x_k bestimmt ist. Für $k = n-1$ erkennt man auch sofort aus Gleichung (4.9), dass $\left(a \sum_{i=0}^{n-1} 2^i x_i)\right) \bmod 2^n \operatorname{div} 2^{n-1}$ eindeutig durch $z_{n-1} + z'_{n-1}$ bestimmt ist. Somit ist $\mathrm{MUL}^a_{n-1,n}$ auch eindeutig durch s_{n-1} und x_{n-1} bestimmt, und die Existenz des Algorithmus ist bewiesen.

Wir müssen nur noch die Größe des OBDDs abschätzen. Für die ersten $m+1$ Ebenen folgt eine Größe von insgesamt $2^{m+1}-1$, da es sich dabei um einen vollständigen binären Baum mit 2^m Blättern handelt. Die Anzahl x_k-Knoten mit $k > m$ ist offensichtlich genau die Anzahl möglicher Werte für s_k, da der Algorithmus in den mit x_k markierten Knoten den Wert von s_k speichert. Da $s_k \in \mathbb{Z}_{2^{n-k}}$ ist, erhalten wir – indem wir alle s_k mit $m+1 \le k \le n-1$ zu den $2^{m+1}-1$ oberen Knoten und den 2 Senken hinzuzählen – als obere Schranke für die π-OBDD-Größe von $\mathrm{MUL}^a_{n-1,n}$

$$2^{m+1}+1+\sum_{k=m+1}^{n-1} 2^{n-k} \;=\; 2^{m+1}+1+\sum_{i=1}^{n-m-1} 2^i \;=\; 2^{m+1}+2^{n-m}-1.$$

Da wir $m = \lfloor n/2 \rfloor$ gewählt haben, folgt für gerade n eine obere Schranke von $2^{n/2+1}+2^{n/2}-1 = 3\cdot 2^{n/2}-1$ und für ungerade n eine obere Schranke von

$$2^{\lfloor n/2\rfloor+1}+2^{\lceil n/2\rceil}-1 \;=\; 2\cdot 2^{(n+1)/2}-1 \;=\; 2\cdot\sqrt{2}\cdot 2^{n/2}-1 \;<\; 3\cdot 2^{n/2}-1. \qquad \blacksquare$$

Die obere Schranke aus diesem Theorem und die untere Schranke aus Theorem 4.4 unterscheiden sich für gerades n also nur um einen konstanten Faktor von 363. Zwar hat dieses Ergebnis keine besonderen Auswirkungen auf die Konstruktion von OBDDs für Multiplizierer, aber es zeigt, dass man an dem Beweis von unteren Schranken für die OBDD-Größe von $\mathrm{MUL}^a_{n-1,n}$ praktisch nichts verbessern kann. Natürlich kann es noch bessere untere Schranken für die OBDD-Größe von $\mathrm{MUL}_{n-1,n}$ geben, aber die Beweistechnik, die darauf basiert, einen Faktor in Abhängigkeit von der Variablenordnung konstant zu setzen, ist ausgereizt.

Eine obere Schranke für $\mathrm{MUL}_{n-1,n}$

Wir schließen die Analyse der OBDD-Größe der Multiplikation mit einer oberen Schranke von $O(2^{4n/3})$ für die Darstellung von $\mathrm{MUL}_{n-1,n}$ ab. Dies ist die erste nicht triviale obere Schranke für diese Funktion. Bisher war nur die obere Schranke von $O(2^{2n}/n)$ bekannt, die für jede boolesche Funktion aus $\mathbb{B}_{2n}$ gilt.

Ein Algorithmus, der $\mathrm{MUL}_{n-1,n}(x,y)$ berechnen soll, kann erst den einen Faktor y lesen und dann so wie der Algorithmus aus dem Beweis zu Theorem 4.10 das Ergebnis von $\mathrm{MUL}^y_{n-1,n}(x)$ berechnen. Da dieser Algorithmus einen Faktor vollständig speichern muss, erhält man so ein OBDD der Größe $O(2^{3n/2})$. Allerdings erkennt man leicht, dass man – nachdem man k der x-Variablen gelesen hat – auch die k signifikantesten y-Variablen wieder „vergessen“ kann. So erhalten wir ein etwas kleineres OBDD für $\mathrm{MUL}_{n-1,n}$.

Theorem 4.11 *Es gibt ein OBDD mit* $(7/3)\cdot 2^{4n/3}$ *Knoten, das die Funktion* $\mathrm{MUL}_{n-1,n}$ *darstellt.*

Beweis: Sei $X = \{x_0,\dots,x_{n-1}\}$, $Y = \{y_0,\dots,y_{n-1}\}$ und π die Variablenordnung von $X \cup Y$ mit

$$\big(\pi^{-1}(1),\dots,\pi^{-1}(2n)\big) \;=\; (y_0,\dots,y_{n-1},x_0,\dots,x_{n-1}).$$

Wir beschreiben wie im Beweis zu Theorem 4.10 einen Algorithmus, der die Variablen in der durch π beschriebenen Reihenfolge liest, und nach jeder gelesenen Variablen sich in einem Zustand befindet, sodass jeder Zustand einem OBDD-Knoten entspricht.

Sei $m = \lceil n/3 \rceil - 1$. Zunächst liest der Algorithmus die Variablen $y_0,\dots,y_{n-1}$ sowie $x_0,\dots,x_{m-1}$ und merkt sich die Belegung aller gelesenen Variablen. Also entspricht der obere Teil des zugehörigen OBDDs einem vollständigen binären Baum, dessen 2^{n+m} Blätter die mit y_m markierten Knoten des OBDDs sind. Sei nun s_k wie im Beweis von Theorem 4.10 definiert, d.h.

$$s_k = \left(\left(|y| \cdot \sum_{i=0}^{k-1} 2^i x_i \right) \bmod 2^n \right) \operatorname{div} 2^k.$$

Im Beweis zu Theorem 4.10 haben wir bereits gezeigt, dass s_{k+1} eindeutig durch y, s_k und x_k bestimmt ist. Genauer gesagt geht aus den Gleichungen (4.9) und (4.10) hervor, dass s_{k+1} nur von den Bitstrings $z_{n-1} \dots z_k$ und $z'_{n-1} \dots z'_k$ abhängt, wobei $|z_{n-1} \dots z_k| = s_k$ und $|z'_{n-1} \dots z'_0| = (|y| \cdot x_k \cdot 2^k) \bmod 2^n$ sind. Dieser letzte Wert ist aber offensichtlich wegen der Multiplikation mit 2^k unabhängig von den y-Variablen $y_{n-1} \dots y_{n-k}$. Also hängt s_{k+1} nur von s_k und der Belegung der Variablen $y_0, \dots, y_{n-k-1}$ sowie der Belegung von x_k ab. Aus den Gleichungen (4.9) und (4.10) ist auch offensichtlich, dass der Funktionswert $\mathrm{MUL}_{n-1,n}(x, y)$ nur von s_{n-1}, x_{n-1} und y_0 abhängt.

Unser Algorithmus speichert also auf der x_k-Ebene, $m \le k \le n-1$, den Wert s_k sowie die Belegung von $y_0 \dots y_{n-k-1}$. Anschließend wird die Variable x_k gelesen, mit deren Belegung s_{k+1} (bzw. für $k = n-1$ der Funktionswert $\mathrm{MUL}_{n-1,n}(x, y)$) eindeutig bestimmt ist, und die Belegung von y_{n-k-1} kann „vergessen“ werden. Für s_k gibt es wie im Beweis zu Theorem 4.10 ausgeführt 2^{n-k} mögliche Werte und für $y_0 \dots y_{n-k-1}$ gibt es trivialerweise 2^{n-k} Möglichkeiten. Somit genügen insgesamt 2^{2n-2k} x_k-Knoten für $k \ge m+1$. Rechnen wir die zwei Senken sowie den vollständigen binären Baum der Größe $2^{n+m+1} - 1$ mit den x_m-Knoten als Blättern hinzu, so kommen wir insgesamt auf eine OBDD-Größe von

$$\begin{aligned} 2^{n+m+1} + 1 + \sum_{k=m+1}^{n-1} 2^{2n-2k} &= 2^{n+m+1} + 1 + 4 \cdot \sum_{i=0}^{n-m-2} 2^{2i} \\ &= 2^{n+m+1} + 1 + 4 \cdot \frac{2^{2n-2m-2} - 1}{3} < 2^{n+m+1} + \frac{2^{2n-2m}}{3}. \end{aligned}$$

Da wir $m = \lceil n/3 \rceil - 1$ gewählt haben, ist $m = (n+\tau)/3 - 1$ für ein $\tau \in \{0, 1, 2\}$. Wir erhalten daher als obere Schranke

$$2^{(4/3)n+\tau/3} + \frac{2^{(4/3)n-2\tau/3+2}}{3} = 2^{(4/3)n} \cdot \left(2^{\tau/3} + \frac{2^{2(1-\tau/3)}}{3} \right).$$

Eine einfache Unterscheidung der Fälle $\tau = 0, 1, 2$ zeigt, dass der Term in Klammern seinen maximalen Wert von $7/3$ für $\tau = 0$ annimmt. Dies beweist, dass das hier konstruierte OBDD höchstens $(7/3) \cdot 2^{(4/3)n}$ Knoten hat. ∎

Die „Lücke“ zwischen der unteren Schranke von $\Omega(2^{n/2})$ und der hier gezeigten oberen Schranke ist immer noch recht groß. Es wäre daher wünschenswert, eine von beiden Schranken noch zu verbessern. Interessant an dem Beweis der oberen Schranke ist aber, dass man dabei keinen verwobene (engl. interleaved) Variablenordnung wählt – das ist die von Praktikern bevorzugte Variablenordnung, bei der die x_i- und y_i-Variablen immer abwechselnd gelesen werden.

KAPITEL 5

FBDDs

OBDDs sind durch die Variablenordnung sehr stark eingeschränkt. Bei FBDDs fällt diese Restriktion weg und es sind zahlreiche Funktionen mit exponentieller OBDD-Größe bekannt, die durch polynomiell große FBDDs dargestellt werden können (Wegener, 2000, nennt einige Beispiele). Naturgemäß ist es somit auch schwieriger, große untere Schranken für FBDDs als für OBDDs zu beweisen.

Bryant bewies seine untere Schranke für die OBDD-Größe der Multiplikation, indem er – in Kenntnis der Variablenordnung π – einen Faktor a geschickt konstant wählte. Auch wir haben im vorigen Kapitel diese Methode verwendet, dabei aber mehr Möglichkeiten für a zugelassen. Bei FBDDs scheint das Konstantsetzen eines Faktors nicht so einfach weiterzuhelfen, da es keine feste Variablenordnung gibt. Schließlich können auf jedem Berechnungspfad die Variablen in einer anderen Reihenfolge vorkommen. Die Beweismethode, die bei FBDDs typischerweise angewendet wird, basiert auf der Arbeit von Simon und Szegedy (1993). Man wählt zunächst einen Schnitt durch das FBDD, d.h. eine Menge von Kanten, sodass jeder Berechnungspfad eine Kante dieser Menge enthält. Wenn das FBDD klein ist, so gibt es eine Kante e, durch die viele Berechnungspfade verlaufen. Diese Kante e teilt nun wieder die Menge Z der Variablen in eine obere Menge Z^- und eine untere Menge Z^+ ein, wobei keine Variable aus Z^- auf einem Pfad von e zu einer Senke und keine Variable aus Z^+ auf einem Pfad von der Wurzel zu e vorkommt. Wenn man nun alle partiellen Belegungen mit Trägermenge Z^- betrachtet, so können – wie bei OBDDs – die durch sie induzierten partiellen Berechnungspfade nur dann die Kante e beinhalten, wenn die zugehörigen Subfunktionen gleich sind. Je kleiner das FBDD ist, desto kleiner ist somit der Anteil verschiedener Subfunktionen von partiellen Belegungen mit einer Trägermenge Z^-. Für den Beweis einer unteren Schranke muss man also für eine beliebige Aufteilung der Variablen in zwei Mengen Z^- und Z^+ beweisen, dass nur ein kleiner Anteil der partiellen Belegungen mit Trägermenge Z^- die gleiche Subfunktion definiert. Das Problem ist nun, dass – im Gegensatz zu OBDDs – die Partitionierung der Variablen von der Belegung der Variablen abhängt. Wir können daher nicht einen Faktor a in Kenntnis der Partitionierung wählen, denn die Wahl von a bestimmt mit, über welche Kante des Schnitts ein Berechnungspfad (x, a) verläuft und welche Zerlegung (Z^-, Z^+) man für diesen Pfad erhält. Also lassen sich weder Bryants noch unsere Beweisidee aus dem letzten Kapitel auf FBDDs übertragen.

Nachdem einige Anstrengungen unternommen wurden eine gute untere Schranken für die FBDD-Größe der Multiplikation zu zeigen, war schließlich Ponzio (1995, 1998) mit einer schwach exponentiellen unteren Schranke von $2^{\Omega(\sqrt{n})}$ erfolgreich. Der Beweis ist jedoch wesentlich komplizierter als Bryants OBDD-Beweis. Und auch hier wird auf Bitebene argumentiert. D.h. um zu zeigen, dass zwei Subfunktionen verschieden sind, werden *einzelne* noch freie Variablen untersucht und so fixiert, dass sie die Ungleichheit der Subfunktionen bezeugen.

Eine wesentliche Idee des letzten Kapitels war jedoch, die Belegungen der x-Variablen aus Z^- bzw. aus Z^+ direkt als ganze Zahlen x^- bzw. x^+ aufzufassen, sodass die Belegung aller Variablen die Summe $x^- + x^+$ darstellt. Dann konnten wir über die *Mengen* M^- und M^+ aller möglichen Zahlen x^- bzw. x^+ argumentieren. Genauer gesagt, wir konnten für einen festen Faktor a sowie zwei Zahlen $x, x' \in M^-$ einen Zeugen $z \in M^+$ finden, der beweist, dass sich die mittleren Bits von $a(x+z)$ und $a(x'+z)$ unterscheiden. Hierbei waren nur die Kardinalitäten der Mengen M^- und M^+ von Bedeutung und wir konnten somit von der Bitebene abstrahieren. Die zweite wesentliche Idee bestand darin, die Existenz vieler verschiedener Subfunktionen zu einer gegebenen Partitionierung der Variablen mithilfe des Überdeckungslemmas zu beweisen, dass wir für universelle Hashklassen hergeleitet haben. Wir wollen beide Ideen auch im Folgenden aufgreifen, um eine echt exponentielle untere Schranke von $\Omega\big(2^{n/4}\big)$ für die FBDD-Größe von $\mathrm{MUL}_{n-1,n}$ zu zeigen.

Übersicht über die Beweisidee

Die generelle Idee ist Folgende: Wie oben beschrieben legen wir einen Schnitt durch das FBDD so fest, dass jede Kante des Schnitts die Variablen aus $Z = X \cup Y$ in zwei Teile Z^- und Z^+ teilt, sodass $\left|Z^-\right| = k$ ist, für ein k in der Größenordnung von $n/2$. O.B.d.A. enthält Z^- dann mindestens $k/2$ x-Variablen. Jede Eingabe (x, y) lässt sich als $(x^- + x^+, y^- + y^+)$ darstellen, wobei x^- die durch die Belegung der x-Variablen aus Z^- definierte Zahl in $\mathbb{Z}_{2^n}$ ist, und x^+, y^-, y^+ analog definiert sind. Für x^- haben wir mindestens $2^{k/2}$ Möglichkeiten. Wir wählen nun ein beliebiges y^- fest und beweisen für alle verschiedenen x_1^-, x_2^-, dass die durch (x_1^-, y^-) und (x_2^-, y^-) definierten Subfunktionen von $\mathrm{MUL}_{n-1,n}$ verschieden sind. Also können die durch diese partiellen Belegungen induzierten partiellen Berechnungspfade nicht die gleiche Kante e des Schnitts enthalten. Insgesamt folgt, dass der Schnitt mindestens $2^{k/2}$ verschiedene Kanten haben muss, und wir erhalten eine untere Schranke von $\Omega\big(2^{k/2}\big)$.

Allgemein müssen wir folgendes Problem lösen: Gegeben sind zwei verschiedene Zahlen $x, x' \in \mathbb{Z}_{2^n}$ (die durch verschiedene Belegungen von Z^- gebildet werden), eine Menge $M \subseteq \mathbb{Z}_{2^n}$ (die Menge der Zahlen, die durch Belegungen von Z^+ gebildet werden) sowie eine Menge $A \subseteq \mathbb{Z}_{2^n}$ (die Menge der Faktoren $a = y^- + y^+$, die man für ein festes y^- und alle y^+ erhält). Um zu zeigen, dass die durch (x, y^-) und (x', y^-) definierten Subfunktionen von $\mathrm{MUL}_{n-1,n}$ verschieden sind, genügt es, einen Zeugen $(a, z) \in A \times M$ zu finden, sodass sich die mittleren Bits der Produkte $a(x+z)$ und $a(x'+z)$ unterscheiden.

Eine wichtige Vorarbeit für den Beweis der Existenz solcher Zeugen haben wir schon im letzten Kapitel mit Lemma 4.8 geleistet. Da dieses Lemma nur für ungerade Zahlen gilt, schränken wir uns auf eine Menge $A \subseteq \mathbb{Z}_{2^n}^*$ ein. Diese Einschränkung ist nicht problematisch, weil wir einfach die Subfunktion der Multiplikation betrachten können, bei der die Bits mit niedrigster Signifikanz beider Faktoren auf 1 gesetzt sind. Somit handelt es sich um die Multiplikation zweier ungerader Zahlen.

Um Lemma 4.8 erfolgreich anzuwenden, genügt es zu allen verschiedenen $x, x' \in \mathbb{Z}_{2^n}$ ein $a \in A$ und ein $z \in M$ zu finden, sodass einerseits

$$2 \;\leq\; \big(h^m_{a,0}(x) - h^m_{a,0}(x')\big) \bmod 2^m \;\leq\; 2^{m-1}$$

gilt und andererseits $\big(h^m_{a,0}(z) + h^m_{a,0}(x)\big) \bmod 2^m \;=\; 0$ ist. Wir finden diese Werte a und z in zwei Schritten. Im ersten Schritt schränken wir die Menge A auf eine möglichst große Teilmenge A' ein, für die gilt

$$\forall a \in A' : \quad 2 \;\leq\; \big(h^m_{a,0}(x) - h^m_{a,0}(x')\big) \bmod 2^m \;\leq\; 2^m - 2. \tag{5.1}$$

Ggf. müssen wir nun noch x und x' vertauschen, und können so die rechte Seite von (5.1) durch 2^{m-1} ersetzen. Damit ist der erste Teil der Voraussetzung von Lemma 4.8 für alle $a \in A'$ erfüllt. Als Nächstes wenden wir ein weiteres Überdeckungslemma an, dass wir auch durch die Universalität der multiplikativen Hashklasse herleiten. Dieses Überdeckungslemma sagt uns, dass für ein genügend großes Produkt $|A'| \cdot |M|$ gilt

$$\exists\, a \in A' : \quad h^m_{a,0}(M) \;=\; \mathbb{Z}_{2^m}. \tag{5.2}$$

Wir können also für ein geeignetes $a \in A'$ ein $z \in M$ mit $\left(h^m_{a,0}(z) + h^m_{a,0}(x)\right) \bmod 2^m = 0$ finden. Damit ist auch die zweite Voraussetzung von Lemma 4.8 erfüllt und wir haben die Existenz des Zeugen $(a, z) \in M$ mit $\langle a(x+z)\rangle^{n-1}_{n-1} \neq \langle a(x'+z)\rangle^{n-1}_{n-1}$ nachgewiesen.

Das Abstandslemma für die Multiplikation

Wir beginnen mit der Aussage, dass es zu jeder Menge $A \subseteq \mathbb{Z}^*_{2^n}$ eine große Teilmenge A' gibt, die die Voraussetzung (5.1) erfüllt. Wenn die Funktionen $h^m_{a,0}$ eine Zahl $x \in \mathbb{Z}_{2^n}$ auf den Wert $y \in \mathbb{Z}_{2^m}$ abbildet, so bedeutet dies, dass ax im Bereich $y \cdot 2^{n-m}, \ldots, (y+1) \cdot 2^{n-m} - 1$ liegt. Im Wesentlichen ist somit die Differenz $\left(h^m_{a,0}(x) - h^m_{a,0}(x')\right) \bmod 2^m$ durch die Differenz $(ax - ax') \bmod 2^n$ bestimmt. D.h., die erste Differenz liegt nur am „Rand“, also nahe bei 0 oder nahe bei 2^m, wenn die zweite Differenz auch nahe bei 0 bzw. nahe bei 2^n liegt. Um (5.1) zu zeigen, genügt es eine Menge A' zu bestimmen, für die alle $a(x - x') \bmod 2^n$ mit $a \in A'$ nicht in der Nähe von 0 oder 2^n liegen.

Lemma 5.1 *Sei $\delta = 2^k$ mit $0 \leq k \leq n-1$ und $A \subseteq \mathbb{Z}^*_{2^n}$. Für alle $x, x' \in \mathbb{Z}_{2^n}$ mit $x \neq x'$ gibt es eine Teilmenge $A' \subseteq A$ mit $\left|A'\right| \geq |A| - \delta$, sodass für jeden Faktor $a \in A'$ gilt*

$$\delta \;\leq\; \left(a(x - x')\right) \bmod 2^n \;\leq\; 2^n - \delta.$$

Beweis: Wir bestimmen für $d = (x' - x) \bmod 2^n$ eine obere Schranke für die Kardinalität der Mengen

$$\begin{aligned} A_1(d) &= \left\{a \in \mathbb{Z}^*_{2^n} \;\middle|\; 1 \leq (ad) \bmod 2^n < \delta\right\} \quad \text{und} \\ A_2(d) &= \left\{a \in \mathbb{Z}^*_{2^n} \;\middle|\; 2^n - \delta < (ad) \bmod 2^n < 2^n\right\}. \end{aligned}$$

Da es für $d \neq 0$ kein ungerades a mit $(ad) \bmod 2^n = 0$ gibt, genügt es

$$|A_1(d) \cup A_2(d)| \;\leq\; \delta$$

zu zeigen. Seien r ungerade und s so gewählt, dass $d = r \cdot 2^s$ ist (dies ist für $d \neq 0$ immer möglich). Im Beweis zu Theorem 3.10 (S. 20) haben wir bereits gezeigt, dass $(ad) \bmod 2^n$ für zufälliges $a \in \mathbb{Z}^*_{2^n}$ gleichverteilt über $\left\{1 \cdot 2^s, 3 \cdot 2^s, \ldots, \ldots, (2^{n-s} - 1) \cdot 2^s\right\}$ ist. Also folgt mit $\delta = 2^k$

$$\begin{aligned} |A_1(d)| &= 2^{n-1} \cdot \operatorname*{\mathbf{Prob}}_{a \in \mathbb{Z}^*_{2^n}}\left(0 \leq (ad) \bmod 2^n < \delta\right) \\ &= 2^{n-1} \cdot \frac{\left|\left\{0, \ldots, 2^k - 1\right\} \cap \{1 \cdot 2^s, 3 \cdot 2^s, \ldots, \ldots, (2^{n-s} - 1) \cdot 2^s\}\right|}{2^{n-s-1}} \\ &= 2^s \cdot \left\lfloor 2^k / 2^{s+1} \right\rfloor \;\leq\; 2^{k-1} \;=\; \delta/2. \end{aligned}$$

Da $A_2(d) = A_1(2^n - d)$ ist, folgt $|A_2(d)| = |A_1(2^n - d)| \leq \delta/2$. Somit erhalten wir schließlich $|A_1(d) \cup A_2(d)| \leq \delta$. ∎

Wir wollen dieses Ergebnis nun auf die Hashfunktionen $h^m_{a,0}$ übertragen. Dabei hilft uns folgende Bemerkung.

Bemerkung 5.2 *Seien $2 \leq m < n$, $z, z' \in \mathbb{Z}_{2^n}$ und $k \geq 1$ mit*

$$k \cdot 2^{n-m} \;\leq\; (z - z') \bmod 2^n \;\leq\; 2^{n-1}.$$

Dann gilt

$$k \;\leq\; \left(z \operatorname{div} 2^{n-m} - z' \operatorname{div} 2^{n-m}\right) \bmod 2^m \;\leq\; 2^{m-1}.$$

Beweis: Seien $y = z \operatorname{div} 2^{n-m}$, $y' = z' \operatorname{div} 2^{n-m}$, $\tau = z \bmod 2^{n-m}$ und $\tau' = z' \bmod 2^{n-m}$. Also gilt $z = y \cdot 2^{n-m} + \tau$ sowie $z' = y' \cdot 2^{n-m} + \tau'$ und es folgt

$$z - z' \;=\; (y - y') \cdot 2^{n-m} + \tau - \tau'.$$

Da $-2^{n-m} < \tau - \tau' < 2^{n-m}$ ist, gilt

$$z - z' - 2^{n-m} \;<\; (y - y') \cdot 2^{n-m} \;<\; z - z' + 2^{n-m}.$$

Nach der Voraussetzung $k \geq 1$ ist $2^{n-m} \leq (z - z') \bmod 2^n \leq 2^n - 2^{n-m}$. Somit können wir alle Terme der obigen Ungleichungen modulo 2^n nehmen, ohne dass sich etwas ändert. Wir erhalten

$$\left((y - y' - 1) \cdot 2^{n-m}\right) \bmod 2^n \;<\; (z - z') \bmod 2^n \;<\; \left((y - y' + 1) \cdot 2^{n-m}\right) \bmod 2^n.$$

Schätzen wir nun $(z - z') \bmod 2^n$ mit der Voraussetzung nach unten durch $k \cdot 2^{n-m}$ bzw. oben durch 2^{n-1} ab, so erhalten wir

$$\begin{aligned} \left((y - y' + 1) \cdot 2^{n-m}\right) \bmod 2^n \;&>\; k \cdot 2^{n-m} \quad \text{und} \\ \left((y - y' - 1) \cdot 2^{n-m}\right) \bmod 2^n \;&<\; 2^{n-1} \;=\; 2^{m-1} \cdot 2^{n-m}. \end{aligned}$$

Aus diesen beiden Ungleichungen folgt dann

$$(y - y' + 1) \bmod 2^m \;>\; k \quad \text{und} \quad (y - y' - 1) \bmod 2^m \;<\; 2^{m-1},$$

was schließlich $k \leq (y - y') \bmod 2^m \leq 2^{m-1}$ impliziert. ∎

Korollar 5.3 *Seien $2 \leq m < n$, $A \subseteq \mathbb{Z}^*_{2^n}$ und $x, x' \in \mathbb{Z}_{2^n}$ mit $x \neq x'$. Dann gibt es eine Teilmenge $A' \subseteq A$ mit $\left|A'\right| \geq |A| - 2^{n-m+1}$, sodass für jedes $a \in A'$ gilt*

$$2 \;\leq\; \left(h^m_{a,0}(x) - h^m_{a,0}(x')\right) \bmod 2^m \;\leq\; 2^m - 2.$$

Beweis: Wir wenden Lemma 5.1 mit dem Parameter $\delta = 2^{n-m+1}$ an. Also gibt es eine Menge $A' \subseteq A$, $\left|A'\right| \geq |A| - 2^{n-m+1}$, sodass für alle $a \in A'$ gilt

$$2^{n-m+1} \;\leq\; \left(ax - ax'\right) \bmod 2^n \;\leq\; 2^n - 2^{n-m+1}.$$

Aufgrund der Symmetrie dieser Ungleichungen gelten sie auch, wenn wir x und x' vertauschen. Da sich an der Behauptung für vertauschte x und x' auch nichts ändert können wir o.B.d.A. $2^{n-m+1} \leq (ax - ax') \bmod 2^n \leq 2^{n-1}$ annehmen und die Behauptung folgt dann sofort aus Bemerkung 5.2 und der Definition von $h^m_{a,0}$. ∎

Dieses Korollar liefert uns die Grundlage für den ersten Teil der Beweisidee, eine große Menge $A' \subseteq A$ zu finden, für die Voraussetzung (5.1) gilt (s. S. 36). Im nächsten Abschnitt beschreiben wir, wie wir für diese Menge A' und eine genügend große Menge M auch die zweite Voraussetzung (5.2) erfüllen können.

Ein weiteres Überdeckunslemma für universelle Hashklassen

Unser Ziel ist es, für eine Menge $A' \subseteq \mathbb{Z}_{2^n}^*$ und eine Menge $M \subseteq \mathbb{Z}_{2^n}$ einen Wert $a \in A'$ zu finden, sodass $h^m_{a,0}(M) = \mathbb{Z}_{2^m}$ ist. Da es sich bei $h^m_{a,0}$ um eine Hashfunktion einer universellen Hashklasse handelt, liegt es nahe, eine ähnliche Aussage zunächst für universelle Hashklassen zu beweisen.

Sei $\mathcal{H}$ eine universelle Hashklasse mit Universum U und Wertebereich R. Wir haben nun nur einen Teil der Hashfunktionen zur Verfügung, also eine Teilmenge $F \subseteq \mathcal{H}$. Zudem haben wir eine Menge $V \subseteq U$ gegeben. Ziel ist es, für möglichst kleine, beliebige Teilmengen F und V zu zeigen, dass es eine Hashfunktion $h \in F$ mit $h(V) = R$ gibt. Intuitiv ist es einleuchtend, dass man bei universellen Hashklassen $\mathcal{H}$ gute Ergebnisse erzielen kann. Die Universalität der Hashklasse garantiert nämlich, dass zwei verschiedene Schlüsselpaare aus V nur unter wenigen Hashfunktionen kollidieren. Somit ist die Gesamtzahl aller Kollisionen für alle Hashfunktionen und alle Schlüsselpaare aus V klein. Wenn es aber zu viele Hashfunktionen h mit $h(V) \subsetneqq R$ gibt, kommt es insgesamt zu zu vielen Kollisionen.

Lemma 5.4 *Sei $\mathcal{H}$ eine universelle Hashklasse mit Universum U und Wertebereich R, und sei $V \subseteq U$. Dann gilt für eine zufällig gewählte Hashfunktion $h \in \mathcal{H}$*

$$\mathbf{Prob}\big(h(V) = R\big) \;\geq\; 1 - \frac{(|R|-1)^2}{|V|}.$$

Beweis: Sei $r = |R|$ und k die Kardinalität von V. Im Beweis von Lemma 4.2 hatten wir bereits für $h \in \mathcal{H}$ die Anzahl Schlüsselpaare aus V, die unter h kollidieren, mit

$$\delta_h(V) = \big|\{(x,x') \in V \times V \mid x \neq x' \wedge h(x) = h(x')\}\big|$$

bezeichnet. Sei $\delta_H(V)$ für jedes $H \subseteq \mathcal{H}$ definiert als $\sum_{h \in H} \delta_h(V)$. Indem wir ausnutzen, dass $\mathcal{H}$ eine universelle Hashklasse ist, erhalten wir

$$\begin{aligned}
\delta_{\mathcal{H}}(V) &= \sum_{h \in \mathcal{H}} \big|\{(x,x') \in V \times V \mid x \neq x' \wedge h(x) = h(x')\}\big| \\
&= \sum_{\substack{x,x' \in V \\ x \neq x'}} \big|\{h \in \mathcal{H} \mid h(x) = h(x')\}\big| \;=\; \sum_{\substack{x,x' \in V \\ x \neq x'}} |\mathcal{H}| \cdot \mathop{\mathbf{Prob}}_{h \in \mathcal{H}}\big(h(x) = h(x')\big) \\
&\leq \sum_{\substack{x,x' \in V \\ x \neq x'}} |\mathcal{H}| \cdot \frac{1}{r} \;=\; |\mathcal{H}| \cdot \frac{k \cdot (k-1)}{r}
\end{aligned} \tag{5.3}$$

Wir betrachten nun die Menge F der Hashfunktionen aus $\mathcal{H}$, für die $h(V)$ nicht den gesamten Wertebereich abdeckt, d.h. $F = \{h \in \mathcal{H} \mid h(V) \neq R\}$. Wir berechnen untere Schranken für $\delta_F(V)$ und $\delta_{\mathcal{H}-F}(V)$ und vergleichen deren Summe dann mit der oberen Schranke für $\delta_{\mathcal{H}}(V)$ aus (5.3). Es ist V die disjunkte Vereinigung aller $h^{-1}(y) \cap V$ mit $y \in h(V)$, also

$$\sum_{y \in h(V)} |h^{-1}(y) \cap V| \;=\; k.$$

Nach Definition von F ist $|h(V)| \leq r-1$ für jeder Hashfunktion $h \in F$. Indem wir ausnutzen, dass die reelle Abbildung $(z_1, \ldots, z_{r-1}) \mapsto {z_1}^2 + \ldots + {z_{r-1}}^2$ unter der Nebenbedingung

$z_1 + \ldots + z_{r-1} = k$ ihr Minimum für $z_1 = \ldots = z_{r-1} = k/(r-1)$ hat, erhalten wir

$$\begin{aligned}
\delta_F(V) &= \sum_{h\in F} \delta_h(V) = \sum_{h\in F}\sum_{y\in h(V)} |h^{-1}(y)\cap V|\cdot\big(|h^{-1}(y)\cap V|-1\big)\\
&= \sum_{h\in F}\sum_{y\in h(V)} \big(|h^{-1}(y)\cap V|\big)^2 - \sum_{h\in F}\sum_{y\in h(V)} |h^{-1}(y)\cap V|\\
&\geq \sum_{h\in F}\sum_{y\in h(V)} \frac{k^2}{(r-1)^2} - \sum_{h\in F} k = |F|\cdot\frac{k^2}{r-1} - |F|\cdot k\\
&= |F|\cdot k\cdot\left(\frac{k}{r-1}-1\right). \qquad (5.4)
\end{aligned}$$

Für $h \in \mathcal{H} - F$ ist hingegen $|h(V)|$ gleich r anstatt $r-1$. Wir erhalten völlig analog zur obigen Rechnung, indem wir $|F|$ durch $|\mathcal{H} - F|$ und $r-1$ durch r substituieren,

$$\delta_{\mathcal{H}-F}(V) \geq |\mathcal{H}-F|\cdot k\cdot\left(\frac{k}{r}-1\right). \qquad (5.5)$$

Da nach Definition $\delta_{\mathcal{H}}(V) = \delta_F(V) + \delta_{\mathcal{H}-F}(V)$ ist, folgt aus den Ungleichungen (5.3)–(5.5)

$$|F|\cdot k\cdot\left(\frac{k}{r-1}-1\right) + |\mathcal{H}-F|\cdot k\cdot\left(\frac{k}{r}-1\right) \leq |\mathcal{H}|\cdot\frac{k\cdot(k-1)}{r}.$$

Indem wir $|\mathcal{H}-F|$ durch $|\mathcal{H}|-|F|$ ersetzen und nach $|F|$ auflösen, erhalten wir

$$\begin{aligned}
|F| &\leq \frac{|\mathcal{H}|\cdot(k-1)/r - |\mathcal{H}|\cdot(k/r-1)}{\big(k/(r-1)-1\big)-(k/r-1)} = |\mathcal{H}|\cdot\frac{k-1-k+r}{kr/(r-1)-k}\\
&= |\mathcal{H}|\cdot\frac{(r-1)^2}{kr-k(r-1)} = |\mathcal{H}|\cdot\frac{(r-1)^2}{k}.
\end{aligned}$$

Also ist bei zufälliger Wahl von $h \in \mathcal{H}$ die Wahrscheinlichkeit für das Ereignis $h \in F$ durch $(r-1)^2/k$ beschränkt. ■

Schließlich müssen wir das Ergebnis noch auf die Funktionen $h^m_{a,0}$ mit $a \in \mathbb{Z}^m_{2^n}$ übertragen. Da obiges Überdeckungslemma nur für die multiplikative Hashklasse gilt, d.h. für Funktionen $h^m_{a,b}$, muss man wieder den additiven Term b loswerden. Zur Erinnerung: Der Funktionswert von $h^m_{a,b}$ ist die Zahl, die durch die Bits an den Positionen $n-m, \ldots, n-1$ des Terms $ax+b$ dargestellt werden. Da b bei der multiplikativen Hashklasse aus $\mathbb{Z}_{2^{n-m}}$ gewählt wird, ist sein Einfluss auf diese Bits aber nur gering – eine Tatsache, die wir schon in Kapitel 4 bei Lemma 4.3 ausgenutzt haben.

Lemma 5.5 *Seien $A \subseteq \mathbb{Z}^*_{2^n}$ und $M \subseteq \mathbb{Z}_{2^n}$ mit $|A|\cdot|M| \geq 2^{n+2m+1}$ für ein $m \geq 0$. Dann gibt es ein $a \in A$, sodass $h^m_{a,0}(M) = \mathbb{Z}_{2^m}$ ist.*

Beweis: Da die multiplikative Hashklasse $\mathcal{M}_{n,m+1}$ universell ist (Theorem 3.10), ist für eine zufällig gewählte Hashfunktion $h^{m+1}_{a,b} \in \mathcal{M}_{n,m+1}$ gemäß Lemma 5.4 die Wahrscheinlichkeit für das Ereignis $h_{a,b}(M) \neq \mathbb{Z}_{2^{m+1}}$ durch $(2^{m+1}-1)^2/|M|$ beschränkt. Die Parameter (a,b) der Hashfunktionen $h^{m+1}_{a,b}$ aus $\mathcal{M}_{n,m+1}$ werden aus $\mathbb{Z}^*_{2^n}\times\mathbb{Z}_{2^{n-m-1}}$ gewählt. Also ist die Anzahl solcher Paare (a,b) mit $h_{a,b}(M) \neq \mathbb{Z}_{2^{m+1}}$ beschränkt durch

$$2^{2n-m-2}\cdot\frac{\big(2^{m+1}-1\big)^2}{|M|} < \frac{2^{2n+m}}{|M|} \leq |A|\cdot\frac{2^{2n+m}}{2^{n+2m+1}} = |A|\cdot 2^{n-m-1} = |A\times\mathbb{Z}_{2^{n-m-1}}|.$$

Demnach gibt es ein Paar $(a, b) \in A \times \mathbb{Z}_{2^{n-m-1}}$ mit $h_{a,b}^{m+1}(M) = \mathbb{Z}_{2^{m+1}}$.

Wir zeigen $h_{a,0}^{m}(M) = \mathbb{Z}_{2^m}$ für dieses a, also dass es für jedes $y \in \mathbb{Z}_{2^m}$ ein $x \in M$ mit $h_{a,0}^{M}(x) = y$ gibt. Sei $y \in \mathbb{Z}_{2^m}$ beliebig und $y' = 2y + 1 \in \mathbb{Z}_{2^{m+1}}$. Wegen $h_{a,b}^{m+1}(M) = \mathbb{Z}_{2^{m+1}}$ gibt es ein $x \in M$ mit $h_{a,b}^{m+1}(x) = y' = 2y + 1$, also

$$(2y + 1) \cdot 2^{n-m-1} \;\leq\; (ax + b) \bmod 2^n \;<\; (2y + 2) \cdot 2^{n-m-1}.$$

Da $0 \leq b < 2^{n-m-1}$ ist, erhalten wir

$$y \cdot 2^{n-m} \;<\; (ax) \bmod 2^n \;<\; (y + 1) \cdot 2^{n-m}.$$

Somit ist $h_{a,0}^{m}(x) = y$. ■

Dieses Lemma liefert den zweiten Teil zu unserer Beweisidee: Es zeigt, dass für genügend große Mengen A' und M Voraussetzung (5.2) gilt (s. S. 37). Wir können daher im folgenden Abschnitt unsere Idee realisieren und die untere Schranke für FBDDs formal beweisen.

Eine echt exponentielle untere Schranke für die FBDD-Größe der Multiplikation

Die allgemeine Idee, untere Schranken für FBDDs zu beweisen, die wir zu Beginn des Kapitels informal beschrieben haben, geht auf Simon und Szegedy (1993) zurück. Um die Technik anzugeben, benötigen wir folgende Definition.

Definition 5.6 Ein *Filter* $\mathcal{F}$ über einer Menge $X = \{x_1, \ldots, x_n\}$ von Variablen ist eine Teilmenge der Potenzmenge von X, sodass für jedes $S \in \mathcal{F}$ auch alle S' mit $S \subseteq S' \subseteq X$ Element von $\mathcal{F}$ sind. Die *Grenze* eines Filters $\mathcal{F}$ ist die Menge aller $B \subseteq X$ mit $B \notin \mathcal{F}$, für die eine Variable $x_i \in X$ existiert, sodass $B \cup \{x_i\} \in \mathcal{F}$ ist.

Sei G ein FBDD für die Funktion $f \in B_n$. Der Filter definiert formal den Schnitt durch G, den wir zu Beginn des Kapitels erwähnten. Dieser besteht aus den Kanten e, für die die Menge der Variablen, die nicht auf einem Pfad von der Wurzel zu e vorkommen, eine Grenze $B(e)$ des Filters bildet. Dann definieren alle partiellen Belegungen α mit Trägermenge $X - B(e)$, deren Berechnungspfade zur Kante e führen, die gleiche Subfunktion $f_{|\alpha}$. Diese Überlegung führt zu folgender Aussage.

Theorem 5.7 (Simon und Szegedy, 1993) *Sei $f \in B_n$ eine Funktion, die über die Variablen aus $X = \{x_1, \ldots, x_n\}$ definiert ist, und sei $\mathcal{F}$ ein Filter über X. Wenn es für jede Menge B aus der Grenze von $\mathcal{F}$ höchstens $k = 2^{n-|B|}/L$ partielle Belegungen $\alpha_1, \ldots, \alpha_k \in \mathcal{P}(X, X - B)$ mit $f_{|\alpha_1} = f_{|\alpha_2} = \ldots = f_{|\alpha_k}$ gibt, so hat jedes* FBDD *für f mindestens L Knoten.*

Wir können nun mit den bisherigen Ergebnissen dieses Kapitels die Technik von Simon und Szegedy auf die Funktion $\mathrm{MUL}_{n-1,n}$ anwenden.

Theorem 5.8 *Jedes* FBDD *für* $\mathrm{MUL}_{n-1,n}$ *hat eine Größe von mindestens* $2^{\lfloor (n-7)/4 \rfloor}$.

Beweis: Wir nehmen an, dass $n - 7$ ein Vielfaches von 4 ist, und beweisen eine untere Schranke von $2^{(n-7)/4}$. Seien $m = (n + 1)/4$ und $k = n - m + 1$ (mit der eben getroffenen Annahme also natürliche Zahlen). Weiterhin sei $\mathrm{MUL}'_{n-1,n}$ die Subfunktion von $\mathrm{MUL}_{n-1,n}$, bei der die Bits mit geringster Signifikanz beider Faktoren auf 1 gesetzt sind. Offensichtlich ist die FBDD-Größe von $\mathrm{MUL}'_{n-1,n}$ eine untere Schranke für die FBDD-Größe von $\mathrm{MUL}_{n-1,n}$. Das Konstantsetzen dieser Bits führt dazu, dass beide an der Multiplikation beteiligten Faktoren ungerade, also aus $\mathbb{Z}^*_{2^n}$ sind. Wir müssen aber darauf achten, dass die Funktion $\mathrm{MUL}'_{n-1,n}$ nur von $N = 2n - 2$ Variablen abhängt.

Die Funktion $\mathrm{MUL}'_{n-1,n}$ sei über die Variablen aus $Z = X \cup Y$ mit $X = \{x_1, \dots, x_{n-1}\}$ und $Y = \{y_1, \dots, y_{n-1}\}$ definiert. Wir betrachten folgenden Filter über Z:

$$\mathcal{F} = \left\{ Z' \subseteq Z \,\middle|\, |Z'| > 2k + 1 \right\}.$$

Sei $B \subseteq Z$ aus der Grenze von $\mathcal{F}$. Offensichtlich enthält B dann genau $2k + 1$ Variablen und es gilt $|Z - B| = N - 2k - 1$. Seien $Y^- = Y \cap (Z - B)$, $X^- = X \cap (Z - B)$ und $Y^+ = Y - Y^-$ sowie $X^+ = X - X^-$. Wir können o.B.d.A. $|Y^-| \le |X^-|$ annehmen, also

$$\left|Y^-\right| \le \left\lfloor \frac{|Z - B|}{2} \right\rfloor = \left\lfloor \frac{N - 2k - 1}{2} \right\rfloor = \left\lfloor \frac{2n - 2k - 3}{2} \right\rfloor = n - k - 2. \tag{5.6}$$

Wir beweisen nun, dass zwei Subfunktionen, die durch unterschiedliches Konstantsetzen der Variablen aus $Z - B$ entstehen, nur dann gleich sein können, wenn sich die entsprechenden Belegungen in den y-Variablen unterscheiden. Genauer gesagt, zeigen wir für alle $\alpha \in \mathcal{P}(Z, Y^-)$ und alle $\beta, \beta' \in \mathcal{P}(Z, X^-)$ mit $\beta \neq \beta'$, dass

$$\left(\mathrm{MUL}'_{n-1,n}\right)_{|\alpha\beta} \neq \left(\mathrm{MUL}'_{n-1,n}\right)_{|\alpha\beta'}$$

gilt. Offensichtlich gibt es dann höchstens $2^{|Y^-|}$ verschiedene partielle Belegungen γ mit Trägermenge $Z - B$, sodass die zugehörigen Subfunktionen $\left(\mathrm{MUL}'_{n-1,n}\right)_{|\gamma}$ alle gleich sind. Mit Theorem 5.7, Ungleichung (5.6) und der Wahl von $m = (n + 1)/4$ und $k = n - m + 1$ erhalten wir somit als untere Schranke für die FBDD-Größe von $\mathrm{MUL}'_{n-1,n}$

$$\frac{2^{N-|B|}}{2^{|Y^-|}} \ge 2^{N-(2k+1)-(n-k-2)} = 2^{n-k-1} = 2^{m-2} = 2^{(n-7)/4}.$$

Seien also $\alpha \in \mathcal{P}(Z, Y^-)$ und $\beta, \beta' \in \mathcal{P}(Z, X^-)$ mit $\beta \neq \beta'$. Es genügt zu zeigen, dass es eine partielle Belegung $\alpha' \in \mathcal{P}(Y, Y^+)$ sowie eine partielle Belegung $\lambda \in \mathcal{P}(X, X^+)$ gibt mit

$$\mathrm{MUL}'_{n-1,n}(\beta\lambda, \alpha\alpha') \neq \mathrm{MUL}'_{n-1,n}(\beta'\lambda, \alpha\alpha'). \tag{5.7}$$

Die Belegungen $\alpha\alpha'$, mit α' wie oben, definieren genau $2^{|Y^+|}$ verschiedene ungerade Zahlen $a \in \mathbb{Z}^*_{2^n}$ (indem man zusätzlich zu den durch $\alpha\alpha'$ festgesetzten Bits das Bit a_0 auf 1 setzt). Analog sei M die Menge der durch λ definierten ungeraden Zahlen, indem man zusätzlich $\lambda_0 = 1$ wählt und alle freien Bits auf 0 setzt. Schließlich seien $x = |\beta|$ und $x' = |\beta'|$. Offensichtlich gibt es genau dann so ein Paar (α', λ), das (5.7) erfüllt, wenn es ein Paar $(a, z) \in A \times M$ gibt, mit

$$\langle a(x + z)\rangle_{n-1}^{n-1} \neq \langle a(x' + z)\rangle_{n-1}^{n-1}. \tag{5.8}$$

Nach Konstruktion gilt $|A| = 2^{|Y^+|}$ sowie $|M| = 2^{|X^+|}$. Da $|Y^+| = |Y| - |Y^-|$ ist, erhalten wir mit Ungleichung (5.6) und mit der Wahl von $k = n - m + 1$

$$|A| \ge 2^{n-1-(n-k-2)} = 2^{k+1} = 2^{n-m+2}. \tag{5.9}$$

Da x, x' verschiedene Elemente aus $\mathbb{Z}_{2^n}$ sind, gibt es gemäß Korollar 5.3 eine Teilmenge $A' \subseteq A$ mit $|A'| \geq |A| - 2^{n-m+1} \geq 2^{n-m+1}$, sodass gilt

$$\forall a \in A' : \; 2 \;\leq\; \bigl(h^m_{a,0}(x) - h^m_{a,0}(x')\bigr) \bmod 2^n \;\leq\; 2^m - 2.$$

Da wir x und x' vertauschen können, nehmen wir o.B.d.A.

$$2 \;\leq\; h^m_{a,0}(x) - h^m_{a,0}(x') \;\leq\; 2^{m-1} \tag{5.10}$$

an. Weiterhin ist B die disjunkte Vereinigung von X^+ und Y^+, also $|A| \cdot |M| = 2^{|B|}$. Es folgt mit (5.9), $k = n - m + 1$ und $|B| = 2k + 1$

$$|M| \;=\; 2^{|B|}/|A| \;=\; 2^{2k+1}/|A| \;\geq\; 2^{2k-n+m-1} \;=\; 2^{n-m+1}.$$

Da $|A'| \geq 2^{n-m+1}$ ist, folgt mit der Wahl von $m = (n+1)/4$

$$\begin{aligned} |A'| \cdot |M| \;&\geq\; 2^{2n-2m+2} \;=\; 2^{2n-(n+1)/2+2} \;=\; 2^{n+2n/4+2/4+1} \\ &=\; 2^{n+2(n+1)/4+1} \;=\; 2^{n+2m+1}. \end{aligned}$$

Nun können wir Lemma 5.5 anwenden und erhalten ein $a \in A' \subseteq A$, sodass $h^m_{a,0}(M) = \mathbb{Z}_{2^m}$ ist. Somit gibt es ein $z \in M$, für das $\bigl(h^m_{a,0}(z) + h^m_{a,0}(x)\bigr) \bmod 2^m \;=\; 0$ ist, und zusammen mit den Ungleichungen aus (5.10) sind alle Voraussetzungen von Lemma 4.8 erfüllt. Die Anwendung dieses Lemmas liefert uns dann $\langle a(x+z)\rangle^{n-1}_{n-1} \neq \langle a(x'+z)\rangle^{n-1}_{n-1}$, also Aussage (5.8), welche zu zeigen war. ∎

KAPITEL 6

Semantische $(1, +k)$-BPs

Bisher konnten superpolynomielle untere Schranken für die Komplexität der Multiplikation nur bei Branchingprogrammmodellen gezeigt werden, die entweder deterministisch sind und bei denen auf jedem Pfad jede Variable nur einmal vorkommen darf oder die an starke Einschränkungen bzgl. der Reihenfolge der Variablentests gebunden sind (wie z.B. baumgesteuert oder stereotyp). Anders ausgedrückt konnte bisher keine exponentielle untere Schranke für die Multiplikation in einem Branchingprogrammmodell gezeigt werden, das echt allgemeiner als das FBDD-Modell ist.

Wir haben in den vorherigen Kapiteln aber ganz neue Aussagen über die Multiplikation hergeleitet, die für den Beweis unterer Schranken geeignet sind. Daher scheint es angebracht zu versuchen, mit diesen Aussagen auch bei allgemeineren Modelle exponentielle untere Schranken zu beweisen. In diesem Kapitel beginnen wir mit semantischen $(1, +k)$-BPs.

Leider sind $(1, +k)$-BPs bei weitem nicht so gut untersucht wie FBDDs oder OBDDs. Daher sind die Techniken zum Beweis unterer Schranken auch noch nicht so ausgereift. Exponentielle untere Schranken für explizit definierte Funktionen findet man für (syntaktische) $(1, +k)$-BPs bei Sieling (1996), Savický und Žák (1997b) oder Savický und Žák (2000) und für semantische $(1, +k)$-BPs bei Žák (1995), Savický und Žák (1997a) sowie Jukna und Razborov (1998). Dabei haben nur Jukna und Razborov (1998) natürliche Funktionen, nämlich lineare Codes, betrachtet. Die Funktionen der anderen Arbeiten wurden speziell so definiert, dass sich aus ihnen gut untere Schranken oder Hierarchieresultate herleiten ließen. Da keine der in den früheren Arbeiten verwendeten Techniken zu den Aussagen über die Multiplikation aus den letzten Kapiteln passt, werden wir im Folgenden eine neue Beweistechnik vorstellen, die wir speziell so entwerfen, dass wir die bisher gewonnenen Erkenntnisse über die Multiplikation ausnutzen können. Einige Ideen unserer Technik (insbes. die Idee zur nachfolgenden Definition 6.2) sind jedoch an die eben zitierten Beweise für semantische $(1, +k)$-BPs angelehnt.

6.1 Die Beweistechnik

Um die Technik zu beschreiben, benötigen wir noch weitere Definitionen.

Definition 6.1 Sei $X = \{x_1, \ldots, x_n\}$ eine Menge von Variablen. Für zwei partielle Belegungen $\alpha = \alpha_1 \ldots \alpha_n$ und $\beta = \beta_1 \ldots \beta_n$ von X bezeichne $D(\alpha, \beta)$ die Menge der Variablen $x_i \in X$, in denen sich α und β unterscheiden, also für die $\alpha_i \neq \beta_i$ ist.

Es sei daran erinnert, dass zu einer partiellen Belegung α der durch α induzierte Berechnungspfad in einem Branchingprogramm G der Pfad ist, der an der Wurzel beginnt, jeden mit

$x_i \in \mathcal{T}(\alpha)$ markierten Knoten über die mit α_i markierte Kante verlässt und auf dem ersten Knoten endet, der mit einer Variablen $x_j \notin \mathcal{T}(\alpha)$ markiert ist (s. Definition 4.5).

Definition 6.2 Sei G ein deterministisches Branchingprogramm über der Variablenmenge X und e eine Kante aus G. Die Menge $L_G(e)$ enthält alle Paare (α, β) partieller Belegungen von X mit folgenden drei Eigenschaften:

1. $\mathcal{T}(\alpha) = \mathcal{T}(\beta)$ und $\alpha \neq \beta$.
2. Die durch α und β induzierten partiellen Berechnungspfade p_α und p_β enthalten die Kante e.
3. Jede Variable $x_i \in D(\alpha, \beta)$, die auf einem Berechnungspfad einer Eingabe $\alpha\lambda$ oder $\beta\lambda$, $\lambda \in \mathcal{P}\big(X, X - \mathcal{T}(\alpha)\big)$, vorkommt, kommt auch auf beiden partiellen Berechnungspfaden p_α und p_β zwischen der Wurzel und e vor.

Mit L_G bezeichnen wir die Vereinigung aller $L_G(e)$.

Die Menge L_G ist eine Abwandlung der Definition sog. „forgetting pairs“ von Jukna und Razborov (1998). Die Idee dahinter ist folgende: Wenn $(\alpha, \beta) \in L_G(e)$ ist, so können die Berechnungspfade von zwei Eingaben $\alpha\lambda$ und $\beta\lambda$ nur dann zu verschiedenen Senken führen, wenn auf ihnen mindestens eine Variable aus $D(\alpha, \beta)$ zweimal vorkommt.

Bemerkung 6.3 *Sei G ein Branchingprogramm über einer Variablenmenge X für eine beliebige Funktion $f \in B_n$. Für alle $(\alpha, \beta) \in L_G$ und alle $\lambda \in \mathcal{P}\big(X, X - \mathcal{T}(\alpha)\big)$ mit $f(\alpha\lambda) \neq f(\beta\lambda)$ gibt es eine Variable in $D(\alpha, \beta)$, die mindestens zweimal auf dem Berechnungpfad von $\alpha\lambda$ vorkommt.*

Beweis: Seien $(\alpha, \beta) \in L_G(e)$ für eine beliebige Kante e und sei $\lambda \in \mathcal{P}\big(X, X - \mathcal{T}(\alpha)\big)$, sodass jede Variable aus $D(\alpha, \beta)$ nur einmal auf dem Berechnungspfad von $\alpha\lambda$ vorkommt. Seien p und p' die Berechnungspfade von $\alpha\lambda$ und $\beta\lambda$. Aus der Definition von $L_G(e)$ folgt sofort, dass jede Variable aus $D(\alpha, \beta) = D(\alpha\lambda, \beta\lambda)$ auf p zwischen der Wurzel und e vorkommt, also nicht auf dem Subpfad von e zur Senke. Da sich p und p' in e treffen, können sie sich somit nach e nicht mehr trennen und erreichen die gleiche Senke. Es folgt $f(\alpha\lambda) = f(\beta\lambda)$. ■

Findet man also $(\alpha, \beta) \in L_G$ sowie ein λ, sodass $f(\alpha\lambda) \neq f(\beta\lambda)$ ist, so muss mindestens eine Variable aus $\mathcal{T}(\alpha)$ auf dem Berechnungspfad von $\alpha\lambda$ zweimal vorkommen. Um zu beweisen, dass es ein „zu kleines“ $(1, +(k-1))$-BP nicht geben kann, müssen wir aber zeigen, dass in einem kleinen Branchingprogramm mindestens k Variablen auf einem Berechnungspfad zweimal vorkommen. Die Idee ist nun, für ein zu kleines $(1, +(k-1))$-BP eine Folge von sich ergänzenden partiellen Belegungen $\alpha_1, \dots, \alpha_k$ und $\beta_1, \dots, \beta_k$ derart zu finden, dass $(\alpha_1 \dots \alpha_i, \alpha_1 \dots \alpha_{i-1}\beta_i) \in L_G$ für alle $1 \le i \le k$ gilt. Danach beweist man, dass es einen Zeugen λ gibt, sodass für alle $1 \le i \le k$ gilt

$$f(\alpha_1 \dots \alpha_k\lambda) \neq f(\alpha_1 \dots \alpha_{i-1}\beta_i\alpha_{i+1} \dots \alpha_k\lambda).$$

Nun muss auf dem Berechnungspfad von $\alpha_1 \dots \alpha_k\lambda$ zu jedem α_i, $1 \le i \le k$, mindestens eine Variable aus $D(\alpha_i, \beta_i)$ zweimal vorkommen. Also kommen insgesamt k Variablen zweimal vor und man erhält einen Widerspruch zur Existenz eines kleinen $(1, +(k-1))$-BPs.

Auf diese Beweisidee können wir aber noch nicht unsere Aussagen über die Multiplikation anwenden. Denn wenn wir eine untere Schranke für $\mathrm{MUL}_{n-1,n}$ mit den Ideen aus Kapitel 5 zeigen wollen, so dürfen wir nur solche Subfunktionen betrachten, bei denen die zugehörigen partiellen Belegungen sich in einem Variablentyp (o.B.d.A. den y-Variablen) nicht unterscheiden. Wir müssen daher die partiellen Belegungen $\alpha_1, \ldots \alpha_k$ und $\beta_1, \ldots, \beta_k$ so wählen, dass α_i und β_i sich entweder nur in den x-Variablen oder nur in den y-Variablen unterscheiden.

Theorem 6.4 *Sei $f \in B_n$ definiert über die n Variablen aus $Z = X \cup Y$ für disjunkte Variablenmengen X und Y. Weiterhin seien $\ell, t \in \mathbb{N}$ mit $\ell \geq 4$ und $t\ell \leq n$. Wir betrachten beliebige disjunkte Teilmengen $V_1, \ldots, V_t \subseteq Z$ mit $|V_i| \leq \ell$, $1 \leq i \leq t$, und beliebige partielle Belegungen $\alpha_i, \beta_i \in \mathcal{P}(Z, V_i)$ mit $\alpha_i \neq \beta_i$, für die entweder $D(\alpha_i, \beta_i) \subseteq X$ oder $D(\alpha_i, \beta_i) \subseteq Y$ gilt. Angenommen, für alle V_i, α_i, β_i mit diesen Eigenschaften existiert eine Teilmenge $I \subseteq \{1, \ldots, t\}$ mit $|I| \geq k$ sowie eine partielle Belegung $\lambda \in \mathcal{P}\big(Z, Z - (V_1 \cup \ldots \cup V_t)\big)$, sodass gilt*

$$\forall i \in I: \; f(\alpha_1 \ldots \alpha_t \lambda) \neq f(\alpha_1 \ldots \alpha_{i-1} \beta_i \alpha_{i+1} \ldots \alpha_t \lambda).$$

Dann hat jedes semantische $(1, +(k-1))$-BP mindestens $2^{\ell/4-2} + 2$ Knoten.

Wir bereiten den Beweis dieses Theorems vor, indem wir zunächst zeigen, wie man in einem „zu kleinen“ $(1, +k)$-BP ein geeignetes Paar $(\alpha, \beta) \in L_G$ findet.

Lemma 6.5 *Sei $f \in B_n$ definiert über die n Variablen aus $Z = X \cup Y$ für disjunkte Mengen X und Y und sei $4 \leq \ell \leq n$. Wenn es ein Branchingprogramm G über Z für f mit höchstens $2^{\ell/4-2} + 1$ Knoten gibt, dann gibt es auch zwei partielle Belegungen α, β von Z mit folgenden Eigenschaften:*

(a) $(\alpha, \beta) \in L_G$.

(b) $|\mathcal{T}(\alpha)| = |\mathcal{T}(\beta)| \leq \ell$.

(c) $D(\alpha, \beta) \subseteq X$ *oder* $D(\alpha, \beta) \subseteq Y$.

Beweis: Sei $k = \lfloor (\ell + 3)/4 \rfloor \geq \ell/4$. Wir bezeichnen für eine Belegung a der x-Variablen und eine Belegung b der y-Variablen den durch (a, b) definierten Berechnungspfad in G mit $p_{a,b}$. Auf jedem solchen Berechnungspfad $p_{a,b}$ sei $e_{a,b}$ die letzte Kante des kürzesten Teilpfades von $p_{a,b}$, der an der Wurzel beginnt und auf dem entweder k verschiedene x-Variablen oder k verschiedene y-Variablen vorkommen. Wenn es eine solche Kante nicht gibt (weil der Pfad $p_{a,b}$ weder k verschiedene x-Variablen noch k verschiedene y-Variablen enthält), so sei $e_{a,b}$ die letzte Kante auf dem Pfad, also die Kante, die einen inneren Knoten mit der Senke des Berechnungspfads verbindet.

Offensichtlich hat G genau $2(|G| - 2)$ Kanten, da es zwei Senken mit Ausgangsgrad 0 und genau $|G| - 2$ Knoten mit Ausgangsgrad 2 gibt. Da es 2^n verschiedene Belegungen (a, b) von Z gibt, folgt mit dem Schubfachprinzip, dass eine Kante e existiert, sodass $e_{a,b} = e$ für mindestens

$$\frac{2^n}{2(|G| - 2)} > \frac{2^n}{2 \cdot 2^{\ell/4-2}} = 2^{n-\ell/4+1} \geq 2^{n-k+1}$$

Belegungen (a, b) gilt. Sei P die Menge aller Pfade $p_{a,b}$ mit $e_{a,b} = e$, d.h. es gilt mit obiger Rechnung $|P| > 2^{n-k+1}$. Da auf jedem Pfad $p \in P$ zwischen der Wurzel und e höchstens $2k - 1$ verschiedene Variablen vorkommen, gibt es eine Menge $P' \subseteq P$, $|P'| \geq |P|/2$, sodass auf jedem Pfad $p \in P'$ zwischen der Wurzel und e höchstens $\lfloor(2k-1)/2\rfloor = k-1$ verschiedene Variablen eines Typs, also o.B.d.A. y-Variablen vorkommen. Wir halten nun diejenige Belegung b der y-Variablen fest, für die es die meisten a mit $p_{a,b} \in P'$ gibt (also mindestens $|P'|/2^{|Y|}$ viele). Sei A die Menge der Belegungen der x-Variablen mit Berechnungspfaden $p_{a,b} \in P'$ für das fest gewählte b. Wir haben nun eine Belegung b der y-Variablen und eine Menge A von Belegungen der x-Variablen mit folgenden Eigenschaften konstruiert:

(i) $|A| \;\geq\; \dfrac{|P'|}{2^{|Y|}} \;\geq\; \dfrac{|P|/2}{2^{|Y|}} \;>\; 2^{n-|Y|-k} \;=\; 2^{|X|-k}.$

(ii) Jeder Pfad $p_{a,b}$ mit $a \in A$ enthält die Kante e und zwischen der Wurzel und der Kante e kommen auf $p_{a,b}$ höchstens $k-1$ verschiedene y-Variablen vor.

(iii) Die Kante e ist entweder inzident zu einer Senke oder auf jedem Pfad $p_{a,b}$, $a \in A$, kommen zwischen der Wurzel und e genau k verschiedene x-Variablen vor.

Wir betrachten nun zwei Fälle. Im ersten Fall gibt es ein $a \in A$, sodass $p_{a,b}$ weniger als k verschiedene x-Variablen zwischen der Wurzel und e enthält. Also ist e eingehende Kante einer Senke. Sei γ die partielle Belegung, die als Trägermenge genau die Variablen enthält, die auf dem Pfad $p_{a,b}$ vorkommen und deren induzierter partieller Berechnungspfad genau $p_{a,b}$ ist. Offensichtlich gilt dann $|\mathcal{T}(\gamma)| \leq 2k-2$. Wir wählen eine beliebige Variable $x_i \in X - \mathcal{T}(\gamma)$ und die partiellen Belegungen α und β mit Trägermenge $V = \mathcal{T}(\gamma) \cup \{x_i\}$, die alle Variablen aus $\mathcal{T}(\gamma)$ genauso wie γ belegen, aber zusätzlich die Variable x_i mit 0 bzw. mit 1 belegen. Offensichtlich haben dann α und β die gleiche Trägermenge V und es ist $D(\alpha, \beta) = \{x_i\}$. Da $p_{a,b}$ genau der durch α bzw. β induzierte partielle Berechnungspfad ist und x_i nicht auf diesem Pfad vorkommt, folgt $(\alpha, \beta) \in L_G$. Außerdem gilt

$$|\mathcal{T}(\alpha)| \;=\; |\mathcal{T}(\beta)| \;=\; |\mathcal{T}(\gamma)| + 1 \;\leq\; 2k-1 \;=\; 2 \cdot \left\lfloor \frac{\ell+3}{4} \right\rfloor - 1 \;\leq\; \frac{\ell+1}{2} \;\leq\; \ell.$$

Schließlich ist offensichtlich $D(\alpha, \beta) \subseteq X$ und die Eigenschaften (a)-(c) der Behauptung sind erfüllt.

Wir kommen nun zum zweiten Fall, bei dem die Pfade $p_{a,b}$ für alle $a \in A$ genau k verschiedene x-Variablen zwischen der Wurzel und e enthalten. Angenommen, alle Pfade $p_{a,b}$ mit $a \in A$ haben genau den gleichen Subpfad von der Wurzel zu e. Da auf diesem Subpfad genau k verschiedene x-Variablen vorkommen, unterscheiden sich die Belegungen $a \in A$ in den zugehörigen Belegungen der k x-Variablen nicht. Dies bedeutet aber $|A| \leq 2^{|X|-k}$ im Widerspruch zur Eigenschaft (i).

Also gibt es zwei verschiedene Belegungen $a, a' \in A$ und eine Variable $x_i \in D(a, a')$, die auf beiden Pfaden $p_{a,b}$ und $p_{a',b}$ zwischen der Wurzel und e vorkommt. Seien U und U' die Mengen der Variablen, die jeweils auf den Pfaden $p_{a,b}$ bzw. $p_{a',b}$ zwischen der Wurzel und e vorkommen. Zwar können U und U' verschiedene x- und y-Variablen enthalten, aber zumindest die Variable x_i ist in beiden Mengen vorhanden. Da auf jedem Pfad $p_{a,b}$ mit $a \in A$ vor e höchstens $2k-1$ verschiedene Variablen vorkommen, gilt $|U|, |U'| \leq 2k-1$. Mit $x_i \in U \cap U'$ und der Wahl von $k = \lfloor(\ell+3)/4\rfloor$ folgt dann für $V = U \cup U'$

$$|V| \;\leq\; |U| + |U'| - 1 \;\leq\; 4k-3 \;\leq\; \ell. \tag{6.1}$$

Seien nun $\alpha, \beta \in \mathcal{P}(Z, V)$ partielle Belegungen, deren nicht freie Variablen wie folgt belegt sind: Jede Variable $z_i \in U$ wird von α mit der Konstanten belegt, die der Belegung dieser Variablen durch die Eingabe (a, b) entspricht und jede Variable $z_i \in V - U$ wird von α mit der

Konstanten belegt, die der Belegung dieser Variablen durch die Eingabe (a', b) entspricht. Analog wird jede Variable $z_i \in U'$ von β mit der Konstanten belegt, die der Belegung dieser Variablen durch die Eingabe (a', b) entspricht und es wird jede Variable $z_i \in V - U'$ von β mit der Konstanten belegt, die der Belegung dieser Variablen durch die Eingabe (a, b) entspricht. Dann belegen offensichtlich α und β die Variablen aus $V - (U \cap U')$ gleich, d.h. $D(\alpha, \beta) \subseteq U \cap U'$. Zudem stimmen der durch α (bzw. β) induzierte partielle Berechnungspfad und der Berechnungspfad $p_{a,b}$ (bzw. $p_{a',b}$) zwischen der Wurzel und e überein. Da nach Konstruktion jede Variable aus $D(\alpha, \beta) \subseteq U \cap U'$ auf beiden Pfaden $p_{a,b}$ und $p_{a',b}$ zwischen der Wurzel und e vorkommt und zudem $\mathcal{T}(\alpha) = \mathcal{T}(\beta)$, aber $\alpha \neq \beta$ gilt (α und β unterscheiden sich zumindest in der Belegung der Variablen x_i), folgt also $(\alpha, \beta) \in L_G(e) \subseteq L_G$. Weiterhin gilt mit (6.1) $|\mathcal{T}(\alpha)| = |\mathcal{T}(\beta)| = |V| \leq \ell$ und zudem $D(\alpha, \beta) \subseteq X$, weil α und β in ihrer Belegung der nicht freien Variablen mit den Eingaben (a, b) bzw. (a', b) für das gleiche b übereinstimmen. Insgesamt haben wir die Behauptungen (a)-(c) gezeigt. ∎

Dieses Lemma liefert uns nun die Möglichkeit, in einem genügend kleinen Branchingprogramm ein Paar $(\alpha, \beta) \in L_G$ zu finden, sodass sich α und β entweder nur in den x-Variablen oder nur in den y-Variablen unterscheiden. Unsere Idee basiert aber darauf, mehrere partielle Belegungen $\alpha_1, \ldots, \alpha_t$ und $\beta_1, \ldots, \beta_t$ zu finden, sodass alle Paare $(\alpha_1 \ldots \alpha_i, \alpha_1 \ldots \alpha_{i-1}\beta_i)$ aus L_G sind. Dies ist möglich, indem man bei dem Branchingprogramm auf geeignete Weise die Variablen sukzessive konstant setzt.

Definition 6.6 Sei G ein Branchingprogramm über $X = \{x_1, \ldots, x_n\}$ für eine Funktion $f \in B_n$. Für eine partielle Belegung α bezeichnet $G_{|\alpha}$ das Branchingprogramm für die Subfunktion $f_{|\alpha}$, das man auf folgende Weise erhält: Für jeden mit einer Variablen $x_i \in \mathcal{T}(\alpha)$ markierten Knoten v wird jede Kante (u, v) durch eine Kante (u, v') mit der gleichen Markierung wie (u, v) ersetzt, wobei v' der α_i-Nachfolger von v ist. Anschließend kann v entfernt werden.

Es ist offensichtlich, dass $G_{|\alpha}$ tatsächlich die Subfunktion $f_{|\alpha}$ darstellt. Insbesondere erhalten wir aber für jedes Paar aus $L_{G_{|\alpha}}$ auch ein Paar aus L_G.

Bemerkung 6.7 *Sei G ein Branchingprogramm über einer Variablenmenge X und γ eine partielle Belegung von X. Dann gilt $(\alpha\gamma, \beta\gamma) \in L_G$ für jedes Paar $(\alpha, \beta) \in L_{G_{|\gamma}}$.*

Beweis: Zunächst bemerken wir, dass der Berechnungspfad einer Eingabe a in $G_{|\gamma}$ genau diejenigen Knoten des Berechnungspfads von $a\gamma$ in G enthält, die nicht mit Variablen aus $\mathcal{T}(\gamma)$ markiert sind.

Angenommen, die Behauptung gilt nicht, also $(\alpha\gamma, \beta\gamma) \notin L_G$ für ein Paar $(\alpha, \beta) \in L_{G_{|\gamma}}$. Seien $p_{\alpha\gamma}$ und $p_{\beta\gamma}$ die durch $\alpha\gamma$ bzw. $\beta\gamma$ induzierten partiellen Berechnungspfade in G und seien p_α und p_β die durch α und β induzierten partiellen Berechnungspfade in $G_{|\gamma}$. Weiterhin sei $e = (u, v)$ eine Kante in $G_{|\gamma}$ mit $(\alpha, \beta) \in L_{G_{|\gamma}}(e)$ und $c \in \{0, 1\}$ die Markierung von e. Offensichtlich ist der Knoten u auch in G vorhanden, und $p_{\alpha\gamma}$ und $p_{\beta\gamma}$ enthalten beide u sowie eine mit c markierte Kante $e' = (u, v')$. Da aber nach unserer Annahme $(\alpha\gamma, \beta\gamma) \notin L_G(e')$ ist, gibt es eine partielle Belegung $\lambda \in \mathcal{P}\big(X, X - \mathcal{T}(\alpha\gamma)\big)$, sodass o.B.d.A. auf dem Berechnungspfad von $\alpha\gamma\lambda$ eine Variable $x_i \in D(\alpha\gamma, \beta\gamma)$ vorkommt, die nicht auf $p_{\alpha\gamma}$ oder $p_{\beta\gamma}$ zwischen der Wurzel und e' vorkommt. Dann ist aber auch $x_i \in D(\alpha, \beta)$ und $x_i \notin \mathcal{T}(\gamma)$. Somit enthält der Berechnungspfad von $\alpha\lambda$ in $G_{|\gamma}$ auch die Variable x_i. Da nach unserer Annahme $(\alpha, \beta) \in L_{G_{|\gamma}}(e)$ ist, muss es auf p_α und auf p_β zwischen der Wurzel und

e jeweils einen mit x_i markierten Knoten geben. Dann liegen diese x_i-Knoten aber auch auf den Pfaden $p_{\alpha\gamma}$ bzw. $p_{\beta\gamma}$ zwischen der Wurzel und e'. Dies ist ein Widerspruch. ■

Wir haben nun alle notwendigen Methoden zum Beweis von Theorem 6.4 an der Hand.

Beweis von Theorem 6.4: Sei G ein $(1, +(k-1))$-BP mit $|G| \leq 2^{\ell/4-2} + 1$ für die Funktion $f \in B_n$. Wir konstruieren wie folgt für $1 \leq i \leq t$ geeignete Belegungen α_i, β_i mit Trägermenge V_i, sodass es keine Indexmenge $I \subseteq \{1, \ldots, t\}$ mit $|I| \geq k$ zusammen mit einer partiellen Belegung $\lambda \in \mathcal{P}\big(Z, Z - (V_1 \cup \ldots \cup V_t)\big)$ gibt, für die gilt

$$\forall i \in I : \quad f(\alpha_1 \ldots \alpha_t \lambda) \neq f(\alpha_1 \ldots \alpha_{i-1} \beta_i \alpha_{i+1} \ldots \alpha_t \lambda).$$

Wir wählen zunächst α_1, β_1 mit Trägermenge V_1 so, dass die Aussagen (a)-(c) aus Lemma 6.5 erfüllt sind. Also gilt $|V_1| \leq \ell$ und α_1 und β_1 unterscheiden sich entweder nur in den Belegungen der x-Variablen oder nur in den Belegungen der y-Variablen. Zudem gilt $(\alpha_1, \beta_1) \in L_G$.

Sei nun $1 \leq i < t$. Angenommen, für alle $1 \leq j \leq i$ sind α_j, β_j und V_j den Voraussetzungen des Theorems entsprechend gewählt. Wir betrachten nun das Branchingprogramm $G_{|\alpha_1 \ldots \alpha_i}$. Da $\left|V_j\right| \leq \ell$ für $1 \leq j < i$ gilt, folgt mit der Voraussetzung $t \cdot \ell \leq n$, dass es noch $n - i \cdot \ell \geq \ell$ freie Variablen aus $Z - (V_1 \cup \ldots \cup V_i)$ gibt. Wir können daher wieder Lemma 6.5 anwenden und erhalten so analog zur ersten Runde zwei weitere verschiedene partielle Belegungen $\alpha_{i+1}, \beta_{i+1}$ mit Trägermenge V_{i+1}, sodass $D(\alpha_{i+1}, \beta_{i+1})$ eine Teilmenge von X oder von Y ist, und zudem $(\alpha_{i+1}, \beta_{i+1}) \in L_{G_{|\alpha_1 \ldots \alpha_i}}$ gilt.

Seien schließlich alle α_i, β_i, V_i für $1 \leq i \leq t$ wie beschrieben gewählt. Sei weiterhin $a = \alpha_1 \ldots \alpha_t$ und $b_i = \alpha_1 \ldots \alpha_{i-1} \beta_i \alpha_{i+1} \ldots \alpha_t$. Da $(\alpha_i, \beta_i) \in L_{G_{|\alpha_1 \ldots \alpha_{i-1}}}$ ist, folgt aus Bemerkung 6.7 $(\alpha_1 \ldots \alpha_i, \alpha_1 \ldots \alpha_{i-1}\beta_i) \in L_G$ für alle $1 \leq i \leq t$. Mit der Definition von L_G ist dann offensichtlich, dass gilt

$$\forall 1 \leq i \leq t : \ (a, b_i) \in L_G. \tag{6.2}$$

Angenommen, die Behauptung stimmt nicht und es gibt nun eine Menge I von k Indizes aus $\{1, \ldots, t\}$ sowie eine partielle Belegung $\lambda \in \mathcal{P}\big(Z, Z - (V_1 \cup \ldots \cup V_t)\big)$, sodass gilt

$$\forall i \in I : \ f(a\lambda) \neq f(b_i \lambda).$$

Wir betrachten dann den Berechnungspfad p der Eingabe $a\lambda$. Da höchstens $k-1$ Variablen auf p mehr als einmal vorkommen und alle Variablenmengen V_i, $i \in I$, disjunkt sind, gibt es einen Index $i \in I$, sodass jede Variable aus V_i höchstens einmal auf p vorkommt. Da aber $D(a, b_i) = D(\alpha_i, \beta_i) \subseteq V_i$ ist und nach (6.2) das Paar $(a, b_i) \in L_G$ ist, folgt mit Bemerkung 6.3 $f(a\lambda) = f(b_i\lambda)$. Dies ist ein Widerspruch zur Annahme. ■

6.2 Eine untere Schranke für die Multiplikation

Die eben vorgestellte Beweistechnik wurde schon so formuliert, dass wir sie gut auf die Ergebnisse über die Multiplikation aus den vorherigen Kapiteln anwenden können. Wir wählen $t = 4k$ und ein geeignetes m. Seien $\alpha_1, \ldots, \alpha_t$ und $\beta_1, \ldots, \beta_t$ wie in Theorem 6.4. Da t genügend groß ist, können wir eine Menge J von $2k$ Indizes finden, sodass sich alle α_i und β_i mit $i \in J$ o.B.d.A. nur in den x-Variablen unterscheiden. Auf diese Weise definiert die Belegung der x-Variablen durch $\alpha_1 \ldots \alpha_t$ eine Zahl $x \in \mathbb{Z}_{2^n}$ und jede Belegung der x-Variablen durch

$\alpha_1 \dots \alpha_{i-1}\beta_i\alpha_{i+1} \dots \alpha_t$ definiert eine Zahl x_i. Entscheidend ist nun, dass $\alpha_1 \dots \alpha_t$ und alle $\alpha_1 \dots \alpha_{i-1}\beta_i\alpha_{i+1} \dots \alpha_t$ mit $i \in J$ die y-Variablen gleich belegen. Die möglichen Belegungen λ der noch freien x- und y-Variablen definieren dann Zahlen $z \in M$ und $a \in A$, sodass sich die Eingabe $\alpha_1 \dots \alpha_t\lambda$ als Faktorenpaar $(a, x + z)$ und die Eingaben $\alpha_1 \dots \alpha_{i-1}\beta_i\alpha_{i+1} \dots \alpha_t$ mit $i \in J$ als Faktorenpaare $(a, x_i + z)$ auffassen lassen. Wir wählen die Parameter so, dass A und M wieder genügend groß sind, sodass wir mit Korollar 5.3 und Lemma 5.5 ein $a \in A$ und ein $z \in M$ finden, sodass $h^m_{a,0}(M) = \mathbb{Z}_{2^m}$ ist und

$$2 \leq \left(h^m_{a,0}(x) - h^m_{a,0}(x_i)\right) \bmod 2^n \leq 2^m - 2.$$

für alle $i \in J$ gilt. Indem wir die Menge der J der Indizes weiter einschränken, finden wir k Indizes i, für die dann o.B.d.A. gilt

$$2 \leq \left(h^m_{a,0}(x) - h^m_{a,0}(x_i)\right) \bmod 2^n \leq 2^{m-1}.$$

Somit sind die Voraussetzungen für die Anwendung von Lemma 4.8 geschaffen und wir erhalten einen Zeugen $(a, z) \in A \times M$, sodass für die ausgewählten k Indizes i gilt

$$\langle a(x + z)\rangle^{n-1}_{n-1} \neq \langle a(x_i + z)\rangle^{n-1}_{n-1}.$$

Mit Theorem 6.4 folgt dann die untere Schranke für $\mathrm{MUL}_{n-1,n}$, die wir nun formulieren.

Theorem 6.8 *Jedes semantische $(1, +k)$-BP für $\mathrm{MUL}_{n-1,n}$ hat eine Größe von* $\Omega\left(2^{\frac{n}{48(k+1)}}\right)$.

Beweis: Wie auch schon beim Beweis der unteren Schranke für FBDDs (Theorem 5.8) zeigen wir die untere Schranke für die Subfunktion $\mathrm{MUL}'_{n-1,n}$, die das mittlere Bit des Produkts zweier ungeraden Zahlen darstellt, indem die Bits mit geringster Signifikanz, x_0 und y_0, auf 1 gesetzt werden. Sei also $Z = X \cup Y$ mit $X = \{x_1, \dots, x_n\}$ und $Y = \{y_1, \dots, y_n\}$ die Menge der Variablen, von denen $\mathrm{MUL}'_{n-1,n}$ abhängt.

Weiterhin seien $k \geq 1$, $\ell = \lfloor n/(12k) - 4/3 \rfloor$, $t = 4k$ und $m = 4k\ell + \lceil \log k \rceil + 4$. Wir beweisen die untere Schranke, indem wir Theorem 6.4 mit den eben genannten Parametern ℓ und t anwenden.

Seien α^i, β^i und $V_i \subseteq Z$ für $1 \leq i \leq t$ so, dass sie die Voraussetzungen aus Theorem 6.4 erfüllen. Wir wählen $V = V_1 \cup \ldots \cup V_t$. Da $D(\alpha^i, \beta^i)$ nach Voraussetzung entweder eine Teilmenge von X oder eine Teilmenge von Y ist, gibt es eine Menge $J \subseteq \{1, \dots, k\}$ von genau $t/2 = 2k$ Indizes, sodass o.B.d.A. $D(\alpha^i, \beta^i) \subseteq X$ für alle $i \in J$ gilt.

Wir spalten nun jedes α^i, $1 \leq i \leq t$, in die durch sie definierte partielle Belegung $\alpha^i_x \in \mathcal{P}(X, V \cap X)$ der x-Variablen sowie die durch sie definierte partielle Belegung $\alpha^i_y \in \mathcal{P}(Y, V \cap Y)$ der y-Variablen auf. Genauso verfahren wir mit β^i und erhalten β^i_x und β^i_y. Da für $i \in J$ $D(\alpha^i\beta^i) \subseteq X$ ist, gilt für diese Indizes $\alpha^i_y = \beta^i_y$.

Sei nun $x = \left|\alpha^1_x \dots \alpha^t_x\right|$ und für $i \in J$ sei $x_i = \left|\alpha^1_x \dots \alpha^{i-1}_x\beta^i_x\alpha^{i+1}_x \dots \alpha^t_x\right|$. Jede partielle Belegung $\lambda_x \in \mathcal{P}(X, X - V)$ der noch freien x-Variablen definiert dann eine ungerade Zahl $z \in \mathbb{Z}^*_{2^n}$, bei der zusätzlich zu den Variablen aus $X - V$ das Bit mit niedrigster Signifikanz gesetzt ist. Die vollständige Belegung $\alpha^1_x \dots \alpha^t_x\lambda_x$ entspricht dann der Zahl $x + z$, und wenn man α^i_x durch β^i_x ersetzt, erhält man die Zahl $x_i + z$. Sei M die Menge aller Zahlen $z \in \mathbb{Z}^*_{2^n}$, die man mit einem $\lambda_x \in \mathcal{P}(X, X - V)$ auf diese Weise erhält. Dann ist offensichtlich

$$|M| = 2^{|X|-|V\cap X|}. \tag{6.3}$$

Weiterhin definiert jede Belegung $\lambda_y \in \mathcal{P}(Y, Y - V)$ der noch freien y-Variablen zusammen mit $\alpha_y^1 \dots \alpha_y^t$ eine ungerade Zahl $a \in \mathbb{Z}_{2^n}^*$, indem man auch hier zusätzlich das Bit mit geringster Signifikanz setzt. Da $\alpha_y^j = \beta_y^j$ für $j \in J$ ist, definieren bei festem λ_y alle $\alpha_y^1 \dots \alpha_y^{j-1} \beta_y^j \alpha_y^{j+1} \dots \alpha_y^t \lambda_y$ mit $j \in J$ die gleiche Zahl a. Sei A die Menge aller Zahlen $a \in \mathbb{Z}_{2^n}^*$, die mit einem $\lambda_y \in \mathcal{P}(Y, Y - V)$ auf diese Weise definiert werden. Da es $2^{|Y|-|V \cap Y|}$ solche partiellen Belegungen λ_y gibt, gilt also

$$|A| = 2^{|Y|-|V\cap Y|} \geq 2^{|Y|-|V|} \geq 2^{n-1-t\cdot\ell}. \tag{6.4}$$

Wir beweisen im Folgenden, dass ein Paar $(\lambda_x, \lambda_y) \in \mathcal{P}(X, X - V) \times \mathcal{P}(Y, Y - V)$ und eine Teilmenge $I \subseteq J$ mit $|I| \geq k$ existieren, sodass für alle $i \in I$ gilt

$$\begin{aligned}&\mathrm{MUL}'_{n-1,n}(\alpha_x^1 \dots \alpha_x^t \lambda_x,\ \alpha_y^1 \dots \alpha_y^t \lambda_y)\\ &\qquad\qquad \neq \mathrm{MUL}'_{n-1,n}(\alpha_x^1 \dots \alpha_x^{i-1} \beta_x^i \alpha_x^{i+1} \dots \alpha_x^t \lambda_x,\ \alpha_y^1 \dots \alpha_y^t \lambda_y).\end{aligned}$$

Mit Theorem 6.4 folgt dann eine eine untere Schranke von $\Omega\big(2^{\ell/4}\big) = \Omega\big(2^{n/(48k)}\big)$ für die Größe eines $(1, +(k-1))$-BP für $\mathrm{MUL}'_{n-1,n}$.

Nach unserer obigen Konstruktion genügt es zu zeigen, dass ein Paar $(a, z) \in A \times M$ und eine Teilmenge $I \subseteq J$, $|I| \geq k$, existieren, sodass gilt

$$\forall i \in I :\ \big\langle a(x+z)\big\rangle_{n-1}^{n-1} \neq \big\langle a(x_i+z)\big\rangle_{n-1}^{n-1}. \tag{6.5}$$

Da $x \neq x_i$ für alle $i \in J$ gilt, können wir Korollar 5.3 auf alle Paare (x, x_i), $i \in J$, und die Menge A anwenden. Nach diesem Lemma ist für jedes $i \in J$ die Anzahl der $a \in A$, für die *nicht*

$$2 \leq \big(h_{a,0}^m(x) - h_{a,0}^m(x_i)\big) \bmod 2^n \leq 2^m - 2 \tag{6.6}$$

gilt, durch 2^{n-m+1} beschränkt. Da $|J| = 2k$ ist, gibt es höchstens $2k \cdot 2^{n-m+1}$ Elemente $a \in A$, sodass obige Ungleichung für mindestens eines der Paare (x, x_i) mit $i \in J$ nicht erfüllt ist. Also gibt es eine Menge $A' \subseteq A$ mit $|A'| \geq |A| - 2k \cdot 2^{n-m+1}$, sodass für alle $a \in A'$ und alle $i \in J$ die Ungleichungen (6.6) erfüllt sind. Mit Ungleichung (6.4) und mit der Wahl von $m = 4k\ell + \lceil \log k \rceil + 4$ und $t = 4k$ gilt

$$2k \cdot 2^{n-m+1} = 2^{\log k + 1} \cdot 2^{n-4k\ell-\lceil \log k \rceil - 4 + 1} \leq 2^{n-t\cdot\ell-2} \leq |A|/2.$$

Somit folgt $|A'| \geq |A|/2$. Wir bestimmen nun eine untere Schranke für das Produkt von $|A'|$ und $|M|$, indem wir die (Un)gleichungen (6.3) und (6.4) sowie $(V \cap X) \cup (V \cap Y) = V$, $\ell = \lfloor n/(12k) - 4/3 \rfloor$ und die schon eben verwendete Definition von t und m ausnutzen:

$$\begin{aligned}|A'| \cdot |M| &\geq |A| \cdot |M|/2 = 2^{|X-V\cap X|+|Y-V\cap Y|-1} = 2^{|Z|-|V|-1} \geq 2^{2n-t\cdot\ell-3}\\ &= 2^{n-4k\cdot\lfloor n/(12k)-4/3\rfloor} \geq 2^{2n-n/3+16k/3-3} = 2^{5n/3-32k/3+2k+42k/3-3}\\ &\geq 2^{5n/3-32k/3+2\lceil\log k\rceil+11} > 2^{n+8n/12-32k/3+2\lceil\log k\rceil+9}\\ &= 2^{n+8k(n/(12k)-4/3)+2\lceil\log k\rceil+9} \geq 2^{n+8k\ell+2\lceil\log k\rceil+9} = 2^{n+2m+1}.\end{aligned}$$

Mit dieser Ungleichung folgt aus Lemma 5.5, dass es ein $a \in A'$ gibt, sodass $h_{a,0}^m(M) = \mathbb{Z}_{2^m}$ ist. Sei dieses a von nun an fest.

Die Ungleichungen (6.6) besagen, dass für jedes $i \in J$ entweder

$$2 \leq \big(h_{a,0}^m(x) - h_{a,0}^m(x_i)\big) \bmod 2^m \leq 2^{m-1} \tag{6.7}$$

oder

$$2 \leq \left(h^m_{a,0}(x_i) - h^m_{a,0}(x)\right) \bmod 2^m \leq 2^{m-1} \tag{6.8}$$

gilt. Eine der beiden Aussagen gilt dann aber für mindestens die Hälfte aller Indizes aus J. Sei also $I \subseteq J$, $|I| = |J|/2 = k$ so, dass entweder (6.7) für alle $i \in I$ oder (6.8) für alle $i \in I$ gilt. Ist ersteres der Fall, so wählen wir $z \in M$ so, dass $\left(h^m_{a,0}(x) + h^m_{a,0}(z)\right) \bmod 2^m = 0$ ist (so ein z existiert, da $h^m_{a,0}(M) = \mathbb{Z}_{2^m}$ ist). Gilt hingegen (6.8) für alle $i \in I$, so wählen wir $z \in M$ so, dass $\left(h(x) + h(z)\right) \bmod 2^m = 2^{m-1} - 2$ gilt. In jedem Fall folgt dann mit Lemma 4.8 für alle $i \in I$

$$\langle a(x+z)\rangle^{n-1}_{n-1} < \langle a(x_i+z)\rangle^{n-1}_{n-1}$$

und wir haben Aussage (6.5) bewiesen. ■

Obiges Theorem besagt, dass es kein semantisches $(1,+k)$-BP für $\mathrm{MUL}_{n-1,n}$ mit polynomieller Größe gibt, solange $k = o(n/\log n)$ ist. Es gibt keine explizit definierte Funktion, für die eine superpolynomielle untere Schranken für $(1,+k)$-BPs mit asymptotisch größerem k gezeigt werden konnte. Beim Beweis der unteren Schranke konnten wir diesmal vollständig auf die in den letzten Kapiteln gewonnenen Erkenntnisse über die Multiplikation zurückgreifen. Um diese Erkenntnisse verwenden zu können, mussten wir aber eine Beweistechnik entwickeln, die zu den Aussagen über die Multiplikation passt.

KAPITEL 7

FBDDs mit eingeschränktem Nichtdeterminismus

Im vorherigen Kapitel haben wir gesehen, dass die Erkenntnisse über die Multiplikation aus den Kapiteln 4 und 5 schon ausreichen, auch für allgemeinere Modelle als FBDDs exponentielle untere Schranken zu beweisen. Die natürlichste Verallgemeinerung von FBDDs sind k-BPs und nichtdeterministische FBDDs. Der Beweis exponentieller unterer Schranken für diese Modelle ist daher ein nahe liegendes Ziel. Sowohl für nichtdeterministische FBDDs als auch für k-BPs werden untere Schranken üblicherweise mithilfe von kombinatorischen Rechtecken oder Pseudorechtecken bewiesen. Gerade für diese Technik, die auf Borodin, Razborov und Smolensky (1993) zurückgeht, scheinen aber unsere Erkenntnisse aus den ersten beiden Kapiteln über die Multiplikation nicht stark genug zu sein.

Es ist daher sinnvoll, als „Zwischenziel" zunächst etwas eingeschränktere Modelle zu betrachten, die mächtiger als FBDDs, aber eingeschränkter als k-BPs bzw. als nichtdeterministische FBDDs sind. Hier konzentrieren wir uns auf den Nichtdeterminismus und betrachten als Zwischenmodell sog. $(\vee, k)$-FBDDs, die Savický und Sieling (2000) definiert haben.

7.1 PBDDs und $(\vee, k)$-FBDDs

Partitioned BDDs, kurz PBDDs, stellen eine Verallgemeinerung von OBDDs dar und wurden vor allem wegen ihrer guten algorithmischen Eigenschaften untersucht. Sie wurden von Jain, Bitner, Fussell und Abraham (1992) definiert. Sei $\pi = (\pi_1, \dots, \pi_k)$ ein Vektor von Variablenordnungen und $w = (w_1, \dots, w_k)$ ein Vektor von Funktionen aus B_n mit $w_1 \vee \dots \vee w_k = 1$. Ein (k, w, π)-PBDD ist eine Familie G von k OBDDs $G_1, \dots, G_k$, wobei G_i die Variablenordnung π_i hat und für die durch G_i dargestellte Funktion f_i gilt $f_i = (f_1 \vee \dots \vee f_k) \wedge w_i$. Die durch G dargestellte Funktion ist $f = f_1 \vee \dots \vee f_k$.

Offensichtlich kann man ein k-PBDD auch als ein nichtdeterministisches FBDD mit $k - 1$ nichtdeterministischen Knoten auffassen. Diese nichtdeterministischen Knoten bilden dabei die inneren Knoten eines binären Baums, dessen k Blätter die Wurzeln der OBDDs $G_1, \dots, G_k$ sind. Der Vorteil von PBDDs liegt darin, dass sie einerseits gute algorithmische Eigenschaften haben und andererseits einige Funktionen mit exponentieller OBDD-Komplexität in polynomieller Größe darstellen können.

Weniger aus algorithmischen Gründen, sondern vielmehr, um den Einfluss der Menge von Nichtdeterminismus auf die Mächtigkeit von Branchingprogrammmodellen zu untersuchen, haben Savický und Sieling (2000) die Definition von PBDDs verallgemeinert.

Definition 7.1 Ein $(\vee, k)$-FBDD über $X = \{x_1, \ldots, x_n\}$ ist eine Familie G von k FBDDs $G_1, \ldots, G_k$ über X. Wenn G_i, $1 \le i \le k$, die Funktion $f_i \in B_n$ darstellt, so repräsentiert G die Funktion $f_1 \vee \ldots \vee f_k$. Die Größe des $(\vee, k)$-FBDDs G ist $|G| := |G_1| + \ldots + |G_k|$.

Diese Art von Nichtdeterminismus ist auch deswegen interessant, weil sie dem vom Turingmaschinenmodell bekannten „Rate-Verifikations-Nichtdeterminismus“ entspricht. D.h., es werden erst alle nichtdeterministischen Bits geraten und danach wird deterministisch weitergearbeitet. Während bei OBDDs diese Art des Rate-Verifikations-Nichtdeterminismus zu einem eingeschränkteren Modell bzgl. polynomieller Komplexität führt (vgl. Sauerhoff, 2003), ist die Frage, ob für beliebig (aber polynomiell) große k nichtdeterministische FBDDs allgemeiner als $(\vee, k)$-FBDDs sind, bisher ungelöst. Savický und Sieling (2000) haben für kleine k bewiesen, dass die Hierarchie der Klassen der durch polynomiell große $(\vee, k)$-FBDDs darstellbaren Funktionen bzgl. k echt ist, solange $k \le (2/3)\sqrt{\log n}$ gilt. Solche Ergebnisse belegen, dass eine minimale Menge an zusätzlichem Nichtdeterminismus schon zu einem mächtigereren Branchingprogrammmodell führt.

Ähnlich wie für $(1, +k)$-BPs werden wir im Folgenden eine neue Technik zum Beweis unterer Schranken für $(\vee, k)$-FBDDs entwerfen, mit deren Hilfe wir dann für alle $k = o(n/\log n)$ eine superpolynomielle untere Schranken für die Komplexität von $\mathrm{MUL}_{n-1,n}$ erhalten.

7.2 Untere Schranken für $(\vee, k)$-FBDDs

Wir wollen zum Beweis unterer Schranken für $(\vee, k)$-FBDDs auf eine ähnliche Methode zurückgreifen, wie wir sie in Kapitel 6 für $(1, +k)$-BPs entwickelt haben. Sei $G = \{G_1, \ldots, G_k\}$ ein „zu kleines“ $(\vee, k)$-FBDD für die Funktion $f \in B_n$. Von wesentlicher Bedeutung ist wieder die Definition der Menge L_G (s. Definition 6.2). Die Idee ist, für jedes einzelne deterministische FBDD G_i, $1 \le i \le k$, ein Paar $(\alpha_i, \beta_i) \in L_{G_i}$ zu finden, sodass es eine partielle Belegung λ gibt, mit

$$\forall\, 1 \le i \le k : \quad f(\alpha_1 \ldots \alpha_k \lambda) \;>\; f(\alpha_1 \ldots \alpha_{i-1} \beta_i \alpha_{i+1} \ldots \alpha_k \lambda).$$

Dann muss die Eingabe $\alpha_1 \ldots \alpha_k \lambda$ von mindestens einem FBDD G_i akzeptiert werden, d.h. für die von diesem FBDD dargestellte Funktion f_i gilt $f_i(\alpha_1 \ldots \alpha_k \lambda) = 1$. Da aber $(\alpha_i, \beta_i) \in L_{G_i}$ ist und auf jedem Pfad in G_i jede Variable nur einmal vorkommt, folgt $f_i(\alpha_1 \ldots \alpha_{i-1} \beta_i \alpha_{i+1} \ldots \alpha_k \lambda) = 1$ (mit Bemerkung 6.3) im Widerspruch zur Annahme. Wie auch im vorigen Kapitel besteht das Hauptproblem nun darin, die partiellen Belegungen α_i, β_i so zu finden, dass sie sich für alle $1 \le i \le k$ o.B.d.A. nur in den x-Variablen unterscheiden. Bei der unteren Schranke für $(1, +k)$-BPs hatten wir uns dadurch beholfen, dass wir doppelt so viele Paare (α_i, β_i) auswählten, die sich jeweils nur in der Belegung eines Variablentyps unterschieden und dann passende k Paare erhielten, die sich alle im gleichen Variablentyp unterschieden. Hier funktioniert dieser Trick nicht, da wir jedes ausgewählte Paar (α_i, β_i) einem FBDD G_i zuordnen müssen.

Stattdessen wählen wir von Beginn an die Paare (α_i, β_i) so, dass die Trägermenge von α_i und β_i nur x-Variablen enthält. Dadurch können wir aber nicht mehr garantieren, dass sich die durch α_i und β_i induzierten partiellen Berechnungspfade überhaupt in einer Kante treffen, geschweige denn, dass $(\alpha_i, \beta_i) \in L_{G_i}$ ist. Stattdessen schränken wir zusätzlich die Menge A der möglichen y-Belegungen auf eine Menge A' ein, sodass für alle $a \in A'$ gilt $(\alpha_i a, \beta_i a) \in L_{G_i}$. Dass das möglich ist, zeigt folgendes Lemma.

Um die Unterscheidung zwischen x- und y-Variablen zu verdeutlichen, schreiben wir im Folgenden Eingaben für eine boolesche Funktion, die über einer Variablenmenge $Z = X \cup Y$ definiert ist, immer als Paare (a, b). Dabei ist a eine Belegung der x-Variablen und b eine Belegung der y-Variablen.

Lemma 7.2 *Sei G ein* FBDD *über der Variablenmenge $Z = X \cup Y$ mit $|X| = n$ und $|Y| = n'$, das eine Funktion $f \in B_{n+n'}$ darstellt. Sei weiterhin $1 \leq \ell \leq n$ so, dass $|G| \leq 2^{\ell-1} + 1$ gilt, und sei $A \subseteq \{0,1\}^{n'}$ eine Menge von Belegungen der y-Variablen. Dann gibt es eine Teilmenge $A' \subseteq A$ mit $|A'| \geq |A| \cdot 2^{1-2\ell}$ sowie zwei verschiedene partielle Belegungen $\alpha, \beta \in \mathcal{P}(X, V)$ mit $V \subseteq X$ und $|V| = \ell$, sodass gilt*

$$\forall a \in A' : \ \big((\alpha, a), (\beta, a)\big) \in L_G.$$

Für den Beweis des Lemmas benötigen wir folgende kombinatorische Hilfsaussage.

Behauptung 7.3 *Sei $D = (d_{i,j})$, $1 \leq i \leq r$, $1 \leq j \leq s$, eine Matrix über $\{0, 1\}$ mit mindestens $\epsilon \cdot r \cdot s$ Einseinträgen $(0 \leq \epsilon \leq 1)$. Dann gibt es zwei verschiedene Zeilen i, i' sodass für mehr als $\epsilon \cdot s(\epsilon - 1/r)$ Spalten j gilt $d_{i,j} = d_{i',j} = 1$.*

Beweis: Sei für eine Spalte j und zwei verschiedene Zeilen i, i'

$$\delta_j(i, i') = \begin{cases} 1 & \text{für } d_{i,j} = d_{i',j} = 1 \text{ und} \\ 0 & \text{sonst.} \end{cases}$$

Sei weiterhin k_j die Anzahl Einsen in Spalte j. Nach Voraussetzung ist $\sum_{j=1}^{s} k_j \geq \epsilon \cdot r \cdot s$. Da die Funktion $(k_1, \ldots, k_s) \mapsto \sum_{j=1}^{s} k_j(k_j - 1)$ mit reellen k_j unter der Nebenbedingung $\sum_{j=1}^{s} k_j = K$ ihr Minimum für $k_j = K/s$, $1 \leq j \leq s$, annimmt, folgt

$$\sum_{j=1}^{s} \binom{k_j}{2} = \sum_{j=1}^{s} \frac{k_j(k_j - 1)}{2} \geq \sum_{j=1}^{s} \frac{\epsilon r(\epsilon r - 1)}{2} = \frac{\epsilon rs(\epsilon r - 1)}{2}.$$

Obige Summe beschreibt offensichtlich genau die Anzahl von Kombinationen i, i', j mit $i < i'$, für die $d_{i,j} = d_{i',j} = 1$ gilt. Wir erhalten also den gleichen Wert, wenn wir über alle $\delta_j(i, i')$ summieren:

$$\sum_{1 \leq i < i' \leq r} \sum_{j=1}^{s} \delta_j(i, i') = \sum_{j=1}^{s} \binom{k_j}{2} \geq \frac{\epsilon rs(\epsilon r - 1)}{2}.$$

Nach dem Schubfachprinzip gibt es dann zwei Zeilen i, i', $i < i'$, sodass

$$\sum_{j=1}^{s} \delta_j(i, i') \geq \frac{\epsilon rs(\epsilon r - 1)}{2 \cdot \binom{r}{2}} = \frac{\epsilon rs(\epsilon r - 1)}{r(r - 1)} > \epsilon s(\epsilon - 1/r).$$

Somit gilt für mehr als $\epsilon s(\epsilon - 1/r)$ Spalten j, dass $d_j(i, j) = d_j(i', j) = 1$ ist. ■

Beweis von Lemma 7.2: Sei w die Wurzel von G. Für jeden Knoten $v \neq w$ von G sei v^+ die Menge der Variablen $x_i \in X$, für die es einen Pfad von v zu einer Senke gibt, der einen x_i-Knoten enthält. Analog sei v^- die Menge der Variablen $x_i \in X$, die auf einem

Pfad von w nach v vorkommen. Bei der Definition von v^+ und v^- werden die Start- und Endknoten der Pfade mit berücksichtigt, sodass, falls v selbst mit einer Variablen x_i markiert ist, $v^- \cap v^+ = \{x_i\}$ gilt. Für die Wurzel w sei $w^+ = X$ sowie $w^- = \{x_i\}$, falls w mit x_i markiert ist, und $w^- = \emptyset$, falls w mit einer y-Variablen markiert ist.

Wir definieren einen *Schnitt* durch das FBDD: Sei $\mathcal{C}$ die Menge der Kanten $e = (u, v)$ für die $u^+ > n - \ell$ und $v^+ \leq n - \ell$ gilt. Für jede Kante $e = (u, v)$ ist offensichtlich $v^+ \subseteq u^+$. Da zudem $w^+ = X$ für die Wurzel w und $s^+ = \emptyset$ für jede Senke s gilt, enthält jeder Berechnungspfad genau eine Kante aus dem Schnitt $\mathcal{C}$.

Für jede Kante $e = (u, v) \in \mathcal{C}$ sei $V_e \subseteq X$ eine beliebige ℓ-elementige Menge für die $u^- \subseteq V_e \subseteq X - v^+$ gilt. So eine Menge existiert, denn es ist $u^- \subseteq X - v^+$ und nach Konstruktion des Schnitts gilt $|u^+| > n-\ell$ und $|v^+| \leq n-\ell$, was $|u^-| \leq \ell$ und $|X - v^+| > \ell$ impliziert. Die Definition von V_e stellt sicher, dass V_e alle x-Variablen enthält, die auf einem beliebigen Pfad von w nach u vorkommen, aber keine x-Variable, die auf einem Pfad von v zu einer Senke vorkommt. D.h., dass es für jede feste Belegung a der y-Variablen nur von der partiellen Belegung $\beta \in \mathcal{P}(X, V_e)$ abhängt, ob der Berechnungspfad einer Eingabe $(\beta\gamma, a)$ mit $\gamma \in \mathcal{P}(X, X - V_e)$ die Kante e enthält.

Wir betrachten nun die Menge $K(e)$ aller Paare (β, a) mit $\beta \in \mathcal{P}(X, V_e)$ und $a \in A$, deren induzierte partielle Berechnungspfade die Kante e enthalten. Mit dem Schubfachprinzip gibt es eine Kante $e \in \mathcal{C}$, sodass die Berechnungspfade von mindestens $2^n|A|/|\mathcal{C}|$ Eingaben (x, a) mit $a \in A$ die Kante e enthalten. Wir halten nun diese Kante e für den Rest des Beweises fest. Da es $2^{n-\ell}$ verschiedene Belegungen der Variablen aus $X - V_e$ gibt, gibt es mindestens $2^n|A|/(2^{n-\ell}|\mathcal{C}|)$ Eingaben (x, a), die sich in den Variablen aus $X - V_e$ nicht unterscheiden und deren Berechnungspfade die Kante e enthalten. Also gibt es genauso viele Paare $(\beta, a) \in \mathcal{P}(X, V_e) \times A$, deren induzierte partielle Berechnungspfade die Kante e enthalten (es sei daran erinnert, dass es für festes $a \in A$ nur von der Belegung $\beta \in \mathcal{P}(X, V_e)$ abhängt, ob der durch (β, a) induzierte partielle Berechnungspfad e enthält). Es folgt daher

$$|K(e)| \geq \frac{2^n \cdot |A|}{2^{n-\ell} \cdot |\mathcal{C}|} = \frac{2^\ell \cdot |A|}{|\mathcal{C}|}. \tag{7.1}$$

Wenn man jede Kante $e \in \mathcal{C}$ durch eine Kante ersetzt, die an einem neuen Blatt b_e endet, so kann man in G einen Baum mit höchstens $|G|-2$ inneren Knoten mit Ausgangsgrad 1 oder 2 und $|\mathcal{C}|$ Blättern einbetten. Da so ein Baum mit k inneren Knoten höchstens $k + 1$ Blätter hat, folgt $|\mathcal{C}| \leq |G| - 1$. Wir erhalten also mit Ungleichung (7.1) und der Voraussetzung $|G| \leq 2^{\ell-1} + 1$

$$|K(e)| \geq \frac{2^\ell \cdot |A|}{|G| - 1} \geq 2|A|. \tag{7.2}$$

Wir wählen nun $V = V_e$ und beweisen die Behauptung für diese Menge V. Sind (α, a) und (β, a) aus $K(e)$ mit $\alpha \neq \beta$, so gilt nach Konstruktion $\mathcal{T}(\alpha, a) = \mathcal{T}(\beta, a)$ und die durch (α, a) und (β, a) induzierten partiellen Berechnungspfade enthalten die Kante e. Da e eine Kante des Schnitts $\mathcal{C}$ ist, kommen auf allen Pfaden, die über e verlaufen, Variablen aus $D\big((\alpha, a), (\beta, a)\big) \subseteq V_e$ höchstens zwischen der Wurzel und e vor. Also folgt $\big((\alpha, a), (\beta, a)\big) \in L_G(e)$. Es genügt somit, zwei verschiedene partielle Belegungen $\alpha, \beta \in \mathcal{P}(X, V)$ zu finden, sodass für eine genügend große Teilmenge $A' \subseteq A$ sowohl (α, a) als auch (β, a) in $K(e)$ sind.

Diese erhalten wir aber ganz einfach aus Behauptung 7.3: Betrachten wir die Matrix $D = (d_{\alpha,a})$ mit $\alpha \in \mathcal{P}(X, V)$ und $a \in A$, für die genau dann $d_{\alpha,a} = 1$ gilt, wenn $(\alpha, a) \in K(e)$ ist.

Offensichtlich hat diese Matrix 2^ℓ Zeilen, $|A|$ Spalten und $|K(e)|$ Einseinträge. Es folgt mit

$$\epsilon = \frac{|K(e)|}{2^\ell \cdot |A|} \geq \frac{2|A|}{2^\ell \cdot |A|} = 2^{1-\ell}$$

aus Behauptung 7.3, dass es zwei verschiedene Zeilen α, β gibt, sodass $d_{\alpha,a} = d_{\beta,a} = 1$ für mindestens

$$\epsilon \cdot |A| \cdot (\epsilon - 2^{-\ell}) \geq 2^{1-\ell} \cdot |A| \cdot (2^{1-\ell} - 2^{-\ell}) = |A| \cdot 2^{1-2\ell}$$

Spalten a gilt. Gemäß der Definition der Matrix D bilden die mit diesen Spalten assoziierten Elemente $a \in A$ dann eine Menge $A' \subseteq A$, $\left|A'\right| \geq |A| \cdot 2^{1-2\ell}$, sodass (α, a) und (β, a) für alle $a \in A'$ Elemente von $K(e)$ sind. Somit gilt für alle $a \in A'$ auch $\big((\alpha, a), (\beta, a)\big) \in L_G$. ■

Dieses Lemma stellt den Kern unserer Technik zum Beweis unterer Schranken für $(\vee, k)$-FBDDs dar. Im folgenden Abschnitt werden wir die Beweistechnik vollständig angeben und analysieren.

Die Beweistechnik

Sei $f \in B_n$ eine boolesche Funktion, für deren $(\vee, k)$-FBDD-Größe wir eine untere Schranke beweisen wollen. Wir beschreiben die Technik durch ein Spiel, das zwei Spieler – Alice und Bob – miteinander spielen. Das Spiel besteht aus k Runden; in jeder Runde darf zuerst Alice und dann Bob eine Entscheidung treffen. Am Ende gewinnt genau einer der beiden Spieler. Wir legen die Regeln des Spiels so fest, dass Alice in Kenntnis eines kleinen $(\vee, k)$-FBDDs für f immer gewinnen kann. Wenn es eine *Gewinnstrategie* für Bob gibt – also eine Strategie, mit der Bob immer gewinnen kann, egal welche Entscheidungen Alice trifft – so kann es kein kleines $(\vee, k)$-FBDD für f geben.

Wir entwerfen das Spiel so, dass es für Funktionen geeignet ist, die – wie die Multiplikation – von zwei Variablentypen abhängen. Sei $f \in B_{n+n'}$ über der Variablenmenge $Z = X \cup Y$ definiert, wobei X eine Menge von n x-Variablen und Y eine Menge von n' y-Variablen ist. Die Regeln des Spiels hängen von der Funktion f sowie den Parametern $k, \ell \in \mathbb{N}$ ab. Das Spiel mit den Parametern k, ℓ für die Funktion f nennen wir $\mathcal{G}(f, k, \ell)$.

Bevor die erste Runde beginnt, darf Bob eine Menge $A_0 \subseteq \{0,1\}^{n'}$ möglicher Belegungen der y-Variablen festlegen. Danach beginnt die erste Runde. Alice beginnt in der i-ten Runde, $1 \leq i \leq k$, indem sie eine Teilmenge $A'_i \subseteq A_{i-1}$ mit $|A'_i| \geq 2^{1-2\ell}|A_{i-1}|$ wählt. Dann darf sie noch bis zu ℓ x-Variablen auswählen, die sie noch in keiner früheren Runde gewählt hat, d.h. eine Teilmenge $V_i \subseteq X - (V_1 \cup \ldots \cup V_{i-1})$ mit $|V_i| \leq \ell$, sowie zwei verschiedene partielle Belegungen $\alpha_i, \beta_i \in \mathcal{P}(X, V_i)$ der x-Variablen. Danach beendet Bob die Runde, indem er eine beliebige Teilmenge $A_i \subseteq A'_i$ sowie eine der beiden von Alice bestimmten partiellen Belegungen auswählt, also ein $\gamma_i \in \{\alpha_i, \beta_i\}$.

Wir müssen abschließend noch festlegen, wer das Spiel gewinnt. Sei γ_i^* das von Bob nicht ausgewählte Element der i-ten Runde, d.h. das Element aus $\{\alpha_i, \beta_i\} - \{\gamma_i\}$. Weiterhin seien $V = V_1 \cup \ldots \cup V_k$, $c = \gamma_1 \ldots \gamma_k$ und $c_i = \gamma_1 \ldots \gamma_{i-1}\gamma_i^*\gamma_{i+1} \ldots \gamma_k$. Bob gewinnt das Spiel, wenn es eine partielle Belegung $b \in \mathcal{P}(X, X - V)$ sowie ein $a \in A_k$ gibt mit

$$f(bc, a) = 1 \quad \text{und} \quad \bigvee_{i=1}^{k} f(bc_i, a) = 0. \tag{7.3}$$

Andernfalls gewinnt Alice das Spiel.

Theorem 7.4 *Seien $f \in B_{n+n'}$ und $k, \ell \in \mathbb{N}$ mit $1 \leq k\ell < n$. Wenn es beim Spiel $\mathcal{G}(f, k, \ell)$ eine Gewinnstrategie für Bob gibt, dann hat jedes $(\vee, k)$-FBDD für f ein Größe von mindestens $2^{\ell-1} + 2$.*

Beweis: Sei $G = \{G_1, \ldots, G_k\}$ ein $(\vee, k)$-FBDD über der Variablenmenge $Z = X \cup Y$ mit $|X| = n$ und $|Y| = n'$ für die Funktion f. Wir beweisen das Theorem, indem wir $|G| \leq 2^{\ell-1} + 1$ annehmen und für diesen Fall eine Gewinnstrategie für Alice beschreiben. Es sei an die Bezeichnungen bei der Beschreibung des Spiels erinnert.

Bob wählt zu Beginn eine beliebige Menge $A_0 \subseteq \{0, 1\}^{n'}$. Wir zeigen, dass Alice ihre Entscheidungen in solch einer Weise treffen kann, dass nach der i-ten Runde gilt

$$\forall a \in A_i : \big((\gamma_1 \ldots \gamma_i, a), (\gamma_1 \ldots, \gamma_{i-1}\gamma_i^*, a)\big) \in L_{G_i}. \tag{7.4}$$

Für die erste Runde wenden wir Lemma 7.2 auf G_1 und die Menge A'_0 an (die Voraussetzung $|G_1| \leq 2^{\ell-1}+1$ ist nach unserer Annahme $|G| \leq 2^{\ell-1}+1$ erfüllt). Somit erhalten wir $V_1 \subseteq X$, $\alpha_1, \beta_1 \in \mathcal{P}(X, V_1)$ und $A'_1 \subseteq A_0$, sodass für alle $a \in A'_1$ gilt $\big((\alpha_1, a), (\beta_1, a)\big) \in L_{G_1}$. Diese Objekte stellen dann den Spielzug von Alice dar, der mit $|V_1| = \ell$ und $|A'_1| \geq 2^{1-2\ell} \cdot |A_0|$ regelkonform ist. Danach wählt Bob $A_1 \subseteq A'_1$ sowie ein $\gamma_1 \in \{\alpha_1, \beta_1\}$. Offensichtlich ist dann (7.4) für $i = 1$ erfüllt.

In der $(i + 1)$-ten Runde betrachten wir das FBDD $G' = (G_{i+1})_{|\gamma_1 \ldots \gamma_i}$ (s. Definition 6.6). Wir wenden wieder Lemma 7.2 an, aber diesmal auf G' und die Menge A_i. So erhalten wir analog zur ersten Runde regelkonforme Objekte $V_{i+1} \subseteq X$, $\alpha_{i+1}, \beta_{i+1} \in \mathcal{P}(X, V_{i+1})$ sowie $A'_{i+1} \subseteq A_i$. Hierfür ist zu beachten, dass G' ein FBDD über $X - (V_1 \cup \ldots \cup V_i)$ ist, also über eine Menge von mindestens $n - ki \geq \ell$ freien Variablen, und das $\mathcal{T}(\alpha) = \mathcal{T}(\beta) = V_{i+1}$ offensichtlich disjunkt zu $V_1 \cup \ldots \cup V_i$ ist. Gemäß Lemma 7.2 gilt zudem $\big((\alpha_{i+1}, a), (\beta_{i+1}, a)\big) \in L_{G'}$ für alle $a \in A'_{i+1}$ und mit Bemerkung 6.7 folgt $\big((\gamma_1 \ldots \gamma_i \alpha_{i+1}, a), (\gamma_1 \ldots \gamma_i \beta_{i+1}, a)\big) \in L_{G_{i+1}}$ für diese a. Wenn also anschließend Bob ein $\gamma_{i+1} \in \{\alpha_{i+1}, \beta_{i+1}\}$ sowie $A_{i+1} \subseteq A'_{i+1}$ ausgewählt hat, ist (7.4) auch für $i + 1$ erfüllt.

Wir haben gezeigt, dass Alice das Spiel so spielen kann, dass nach k Runden Aussage (7.4) für alle $1 \leq i \leq k$ gilt. Seien nun $c = \gamma_1 \ldots \gamma_k$ und $c_i = \gamma_1 \ldots \gamma_{i-1}\gamma_i^*\gamma_{i+1} \ldots \gamma_k$. Wir beweisen, dass Alice das Spiel gewonnen hat, also dass (7.3) nicht erfüllt ist. Seien $b \in \mathcal{P}\big(X, X - (V_1 \cup \ldots \cup V_k)\big)$ und $a \in A_k$ beliebig. Wir zeigen

$$f(bc, a) = 0 \quad \text{oder} \quad \bigvee_{i=1}^{k} f(bc_i, a) = 1. \tag{7.5}$$

Für den Fall $f(bc, a) = 0$ ist nichts zu zeigen; sei also $f(bc, a) = 1$. Dann gibt es offensichtlich ein FBDD $G_j \in \{G_1, \ldots, G_k\}$, in dem der Berechnungspfad der Eingabe (bc, a) die 1-Senke erreicht. Da $a \in A_k$ ist und nach den Regeln des Spiels $A_1 \supseteq A_2 \supseteq \ldots \supseteq A_k$ gilt, ist a auch Element von A_j. Nach der j-ten Runde war (7.4) erfüllt, also gilt $\big((\gamma_1 \ldots \gamma_j, a), (\gamma_1 \ldots \gamma_j^*, a)\big) \in L_G$. Da es sich aber bei G_j um ein FBDD handelt, bei dem auf jedem Berechnungspfad jede Variable nur einmal vorkommt, folgt aus Bemerkung 6.3, dass $f(bc, a) = f(bc_j, a)$ ist. Somit ist $\vee_{i=1}^{k} f(bc_i, a) = 1$ und (7.5) ist gezeigt. Somit hat Alice das Spiel gewonnen. ∎

7.3 Eine untere Schranke für die Multiplikation

Um eine untere Schranke für die $(\vee, k)$-FBDD-Größe der Multiplikation zu zeigen, müssen wir für das Spiel $\mathcal{G}(\mathrm{MUL}_{n-1,n}, k, \ell)$ eine Gewinnstrategie für Bob angeben. Wenn das Spiel zu Ende gespielt ist, hat man partielle Belegungen der x-Variablen $\gamma_1 \dots \gamma_k$ und $\gamma_1 \dots \gamma_{i-1}\gamma_i^*\gamma_{i+1} \dots \gamma_k$ bestimmt, die wir wie üblich als ganze Zahlen x bzw. x_i auffassen können. Die möglichen Belegungen der noch freien x-Variablen erzeugen dann wieder eine Menge M von Zahlen $z \in \mathbb{Z}_{2^n}$, sodass für eine solche Belegung λ die Bistrings $c\lambda$ und $c_i\lambda$ den Zahlen x und $x_i + z$ entsprechen. Zudem wird durch den Verlauf des Spiels eine Menge A_k von möglichen Belegungen der y-Variablen bestimmt. Diese Belegungen entsprechen einer Menge A von Zahlen aus $\mathbb{Z}_{2^n}$.

Bobs Strategie sollte also darauf abzielen, dass man am Schluss aus $A \times M$ ein Paar (a, z) auswählen kann, sodass das mittlere Bit von $a(x+z)$ gesetzt ist, während die mittleren Bits von $a(x_i + z)$ alle 0 sind. Dies kann Bob wie üblich mit Lemma 4.8 erreichen, wenn die Voraussetzungen $h^m_{a,0}(M) = \mathbb{Z}_{2^m}$ und $2 \le \big(h^m_{a,0}(x) - h^m_{a,0}(x_i)\big) \bmod 2^m \le 2^{m-1}$ für alle $1 \le i \le k$ erfüllt sind. Die erste Voraussetzung ist gemäß Lemma 5.5 einfach zu erreichen, wenn am Ende des Spiels noch genügend Elemente aus A und M zur Verfügung stehen. Ein kleines Problem tritt mit der zweiten Voraussetzung auf, da sie für alle $1 \le i \le k$ gelten muss. Hier kann aber Bob eingreifen, indem er in jeder Runde die Menge $A_i \subseteq A'_i$ geschickt einschränkt und das γ_i aus $\{\alpha_i, \beta_i\}$ geeignet auswählt, sodass am Schluss auch diese Voraussetzung erfüllt ist.

Theorem 7.5 *Jedes* $(\vee, k)$-FBDD *für* $\mathrm{MUL}_{n-1,n}$ *hat eine Größe von mindestens* $\Omega\big(2^{n/(7k)}\big)$.

Beweis: Wir geben für das Spiel $\mathcal{G}(\mathrm{MUL}_{n-1,n}, k, \ell)$ mit $1 \le \ell \le (n-4)/(7k) - 3/7$ eine Gewinnstrategie für Bob an. Mit Theorem 7.4 folgt dann die Behauptung.

Sei $m = k(2\ell + 1) + 1$. Bevor die erste Runde beginnt, wählt Bob als Menge A_0 die Menge aller Bitstrings $a \in \mathbb{B}_n$ mit $a_0 = 1$, sodass genau die ungeraden Zahlen aus $\mathbb{Z}^*_{2^n}$ mit den Strings aus A_0 dargestellt werden können. Danach trifft Bob seine Entscheidungen in solch einer Weise, dass nach der i-ten Runde für $1 \le i \le k$ folgende Invarianten erfüllt sind:

$$|A_i| \ \ge\ 2^{n-1-i(2\ell+1)} \tag{7.6}$$

$$\forall a \in A_i: \quad 2^{n-m+1} \ \le\ \big(|a| \cdot (|\gamma_i| - |\gamma_i^*|)\big) \bmod 2^n \ \le\ 2^{n-1}. \tag{7.7}$$

Wir beweisen zunächst, dass Bob seine Entscheidungen so treffen kann, dass diese Invarianten nach jeder Runde i erfüllt sind. Danach zeigen wir, dass unter dieser Bedingung Bob das Spiel gewinnt.

Für $i = 0$ gilt $|A_0| = \big|\mathbb{Z}^*_{2^n}\big| = 2^{n-1}$. Also ist Invariante (7.6) erfüllt. Sei nun (7.6) für ein $i \in \{0, \dots, k-1\}$ erfüllt. Wir zeigen, dass Bob seine Entscheidungen so treffen kann, dass beide Invarianten (7.6) und (7.7) nach der $(i+1)$-ten Runde erfüllt sind. Seien also V_{i+1}, $\alpha_{i+1}, \beta_{i+1} \in \mathcal{P}(X, V_{i+1})$ und A'_{i+1} die Objekte, die Alice in der $(i+1)$-ten Runde gewählt hat. Nach den Regeln des Spiels gilt $|A'_{i+1}| \ge 2^{1-2\ell}|A_i|$ und weil (7.6) nach der i-ten Runde erfüllt war, folgt

$$|A'_{i+1}| \ \ge\ 2^{1-2\ell} \cdot 2^{n-1-i(2\ell+1)} \ =\ 2^{n+1-(i+1)(2\ell+1)}. \tag{7.8}$$

Da nach den Regeln des Spiels α_{i+1} und β_{i+1} verschieden sein müssen, gilt offensichtlich auch $|\alpha_{i+1}| \ne |\beta_{i+1}|$. Wir wenden Lemma 5.1 mit $\delta = 2^{n-m+1}$ an. Demnach gibt es eine Teilmenge $A^*_{i+1} \subseteq A'_{i+1}$ mit $\big|A^*_{i+1}\big| \ge |A'_{i+1}| - 2^{n-m+1}$, sodass für jedes $a \in A^*_{i+1}$ gilt

$$2^{n-m+1} \ \le\ \big(|a| \cdot (|\alpha_i| - |\beta_i|)\big) \bmod 2^n \ \le\ 2^n - 2^{n-m+1}.$$

Nun gilt für jedes $a \in A^*_{i+1}$ entweder

$$2^{n-m+1} \;\leq\; \big(|a| \cdot (|\alpha_i| - |\beta_i|)\big) \bmod 2^n \;\leq\; 2^{n-1}$$

oder

$$2^{n-m+1} \;\leq\; \big(|a| \cdot (|\beta_i| - |\alpha_i|)\big) \bmod 2^n \;\leq\; 2^{n-1}.$$

Daher gilt entweder ersteres oder letzteres für mindestens die Hälfte aller Elemente $a \in A^*_{i+1}$ und wir können ein entsprechendes $\gamma_i \in \{\alpha_i, \beta_i\}$ und das zugehörige $\gamma_i^* \in \{\alpha_i, \beta_i\} - \{\gamma_i\}$ auswählen, sodass es eine Teilmenge $A_{i+1} \subseteq A^*_{i+1}$ mit $|A_{i+1}| \geq |A^*_{i+1}|/2$ gibt, für die Invariante (7.7) erfüllt ist. Für die Kardinalität dieser Menge A_{i+1} erhalten wir dann mit Ungleichung (7.8) und der Wahl von $m = k(2\ell + 1) + 1$ sowie mit $i + 1 \leq k$

$$\begin{aligned}|A_{i+1}| \;&\geq\; \frac{|A^*_{i+1}|}{2} \;\geq\; \frac{|A'_{i+1}| - 2^{n-m+1}}{2} \;\geq\; \frac{2^{n+1-(i+1)(2\ell+1)} - 2^{n-k(2\ell+1)}}{2}\\ &\geq\; 2^{n-(i+1)(2\ell+1)} - 2^{n-(i+1)(2\ell+1)-1} \;\geq\; 2^{n-1-(i+1)(2\ell+1)}.\end{aligned}$$

Somit ist für A_{i+1} auch Invariante (7.6) erfüllt.

Es bleibt also nur noch zu beweisen, dass Bob das Spiel gewinnt, wenn die Invarianten (7.6) und (7.7) nach jeder Runde erfüllt sind. Sei $c = \gamma_1 \ldots \gamma_k$ und $c_i = \gamma_1 \ldots \gamma_{i-1}\gamma_i^*\gamma_{i+1} \ldots \gamma_k$. Nach den Gewinnregeln (7.3) des Spiels müssen wir zeigen, dass es ein $b \in \mathcal{P}\big(X, X - (V_1 \cup \ldots \cup V_k)\big)$ sowie ein $a \in A_k$ gibt, sodass gilt

$$\mathrm{MUL}_{n-1,n}(bc, a) \;=\; 1 \quad \text{und} \quad \forall 1 \leq i \leq k : \mathrm{MUL}_{n-1,n}(bc_i, a) \;=\; 0.$$

Sei A die Menge aller $|a|$ mit $a \in A_k$, M die Menge aller $|b| \in \mathbb{Z}_{2^n}$ mit $b \in \big(X, X - (V_1 \cup \ldots \cup V_k)\big)$ und $x = |c| \in \mathbb{Z}_{2^n}$ sowie $x_i = |c_i| \in \mathbb{Z}_{2^n}$ für $1 \leq i \leq k$. Dann müssen wir also zeigen, dass es ein $z \in M$ sowie ein $a \in A$ gibt, sodass

$$\langle a(x + z)\rangle^{n-1}_{n-1} \;=\; 1 \quad \text{und} \quad \forall 1 \leq i \leq k : \langle a(x_i + z)\rangle^{n-1}_{n-1} \;=\; 0 \tag{7.9}$$

ist.

Mit Bemerkung 4.6 gilt für alle $1 \leq i \leq k$

$$x - x_i \;=\; |\gamma_1 \ldots \gamma_k| - |\gamma_1 \ldots \gamma_{i-1}\gamma_i^*\gamma_{i+1} \ldots \gamma_k| \;=\; |\gamma_i| - |\gamma_i^*|.$$

Also folgt aus Invariante (7.7) für alle $a \in A$

$$2 \cdot 2^{n-m} \;\leq\; \big(a(x - x_i)\big) \bmod 2^n \;\leq\; 2^{n-1}.$$

Da $h^m_{a,0}(x) = \big((ax) \bmod 2^n\big) \operatorname{div} 2^{n-m}$ ist, erhalten wir nun mit Bemerkung 5.2

$$\forall\, a \in A :\; 2 \;\leq\; \big(h^m_{a,0}(x) - h^m_{a,0}(x_i)\big) \bmod 2^m \;\leq\; 2^{m-1}. \tag{7.10}$$

Invariante (7.6) sorgt dafür, dass A_k und somit auch A viele Elemente enthält, also

$$|A| \;=\; |A_k| \;\geq\; 2^{n-1-k(2\ell+1)}.$$

Da nach den Regeln des Spiels $|V_i| \leq \ell$ gilt, ist $|X - (V_1 \cup \ldots \cup V_k)| \geq n - k\ell$. Somit gilt $|M| \geq 2^{n-k\ell}$ und mit $m = k(2\ell + 1) + 1$ und $\ell \leq (n - 4)/(7k) - 3/7$ folgt

$$\begin{aligned}|A| \cdot |M| \;&\geq\; 2^{n-1-k(2\ell+1)+n-k\ell} \;=\; 2^{2n-3k\ell-k-1} \;=\; 2^{2n-3(n-4)/7+9k/7-k-1}\\ &=\; 2^{n+4n/7+2k/7+5/7} \;=\; 2^{n+4(n-4)/7-12k/7+2k+3}\\ &=\; 2^{n+4k\ell+2k+3} \;=\; 2^{n+2m+1}.\end{aligned}$$

Wir können nun Lemma 5.5 anwenden und erhalten ein $a \in A$, sodass $h^m_{a,0}(M) = \mathbb{Z}_{2^m}$ gilt. Dann gibt es ein $z \in M$ für das $\bigl(h^m_{a,0}(z) + h^m_{a,0}(x)\bigr) \bmod 2^m = 2^{m-1}$ ist. Wir wenden für alle $1 \le i \le k$ Lemma 4.8 an, indem wir jeweils $x' = x_i$ wählen. Mit (7.10) ist die Voraussetzung dieses Lemmas erfüllt, sodass für alle $1 \le i \le k$

$$\langle a(x+z)\rangle_{n-1}^{n-1} \ > \ \langle a(x_i+z)\rangle_{n-1}^{n-1}$$

folgt. Dies ist aber offensichtlich äquivalent zu Aussage (7.9), welche zu zeigen war. ■

Die soeben bewiesene Aussage liefert für $k = 1$, d.h. für FBDDs, eine untere Schranke von $\Omega\bigl(2^{n/7}\bigr)$. Sie ist also nicht viel schlechter als unsere untere Schranke von $\Omega\bigl(2^{n/4}\bigr)$ aus Kapitel 5. Wir erhalten für alle $k = o(n/\log n)$ noch superpolynomielle untere Schranken. Zudem kann man die unteren Schranken auch auf FBDDs übertragen, bei denen nichtdeterministischen Knoten an beliebiger Stelle vorkommen dürfen – allerdings sind dann nur sehr wenige nichtdeterministische Knoten erlaubt.

Angenommen, es gibt ein FBDD G der Größe L mit t nichtdeterministischen Knoten, das die Funktion f darstellt. Sei v ein nichtdeterministischer Knoten mit ausgehenden Kanten (v, v_1) und (v, v_2). Wir bilden nun zwei FBDDs G_1 und G_2, indem wir in G_i alle Kanten (u, v) durch Kanten (u, v_i) ersetzen. Mit anderen Worten: Wir kodieren die möglichen nichtdeterministischen Entscheidung einer Berechnung am Knoten v fest ein und erhalten so zwei neue FBDDs G_1 und G_2. Wenn G_1 und G_2 nun die Funktionen f_1 und f_2 darstellen, so stellt G offensichtlich die Funktion $f_1 \vee f_2$ dar. Auf gleiche Weise können wir die nichtdeterministischen Entscheidungen aller t nichtdeterministischen Knoten fest einkodieren und erhalten so 2^t FBDDs, sodass f die Disjunktion der von ihnen dargestellten Funktionen ist. Wenn es also ein nichtdeterministisches FBDD der Größe L mit t nichtdeterministischen Knoten für die Funktion f gibt, so gibt es auch ein $(\vee, 2^t)$-FBDD der Größe $L \cdot 2^t$ für f.

Korollar 7.6 *Jedes nichtdeterministische* FBDD *für* $\mathrm{MUL}_{n-1,n}$*, das höchstens t nichtdeterministische Knoten hat, hat eine Größe von* $2^{\Omega(n/2^t)-t}$. *Dies ist superpolynomiell für* $t = \log n - \log\log n - \omega(1)$.

Beweis: Angenommen für jede positive Konstante c gibt es ein nichtdeterministisches FBDD mit t nichtdeterministischen Knoten der Größe $2^{cn/2^t-t}$ für die Funktion $\mathrm{MUL}_{n-1,n}$. Mit der obigen Konstruktion erhält man dann für $k = 2^t$ ein $(\vee, k)$-FBDD der Größe $2^{cn/k}$ für $\mathrm{MUL}_{n-1,n}$. Dies ist für $c < 1/7$ und genügend großes n ein Widerspruch zu Theorem 7.5. ■

Übrigens konnte bis heute für keine Funktion $f \in B_n$ bewiesen werden, dass sie polynomielle Größe für nichtdeterministische FBDDs, aber exponentielle Größe für FBDDs mit höchstens $O(\log n)$ nichtdeterministischen Knoten hat (vgl. Wegener, 2000, Problem 10.5). Korollar 7.6 wesentlich zu verbessern, scheint also ähnlich schwierig zu sein, wie eine untere Schranke für allgemeine nichtdeterministische FBDDs zu beweisen.

KAPITEL 8

Schluss

Wie schwer ist es zu multiplizieren? Wir können nicht erwarten, auf diese eingangs gestellte Frage in naher Zukunft für irgendein allgemeines Rechenmodell wie Schaltkreise oder Branchingprogramme eine vollständige Antwort zu finden. Wir können aber versuchen, nach und nach zu umfassendereren Erkenntnissen über die Multiplikation zu gelangen und neue Techniken zum Nachweis oberer und unterer Schranken zu entwickeln, um auf diese Weise die Komplexität der Multiplikation besser einzugrenzen. Die Ergebnisse der vorliegenden Arbeit stellen einen weiteren Schritt in diese Richtung dar.

Das für die Praxis relevanteste hier vorgestellte Ergebnis ist sicherlich die untere Schranke von $2^{n/2}/61-4$ für OBDDs. Sie beweist erstmals, dass OBDDs zur Verifikation von Multiplikationsschaltkreisen realistischer Größe (z.B. 128-Bit Multiplizierer) ungeeignet sind und bestätigt die seit längerem gehegte Vermutung, dass die wahre OBDD-Größe der Multiplikation wesentlich größer als $2^{n/8}$ ist. Es wird zwar allgemein angenommen, dass die wahre OBDD-Komplexität der Multiplikation mindestens 2^n beträgt, aber unsere obere Schranke aus Theorem 4.10 für die Funktion $\mathrm{MUL}^a_{n-1,n}$ zeigt, dass man für den Beweis unterer Schranken in dieser Größendordnung nicht mehr über das Konstantsetzen eines Faktors argumentieren kann, sondern einen einen anderen Ansatz benötigt. Mit der oberen Schranke von $O\big(2^{(4/3)n}\big)$ für $\mathrm{MUL}_{n-1,n}$ haben wir versucht, die Lücke zwischen unterer und oberer Schranke zumindest etwas weiter zu schließen.

Zum Beweis der unteren Schranke für OBDDs haben wir neue Eigenschaften über die Multiplikation verwendet, die sich aus der bekannten Tatsache ableiten lassen, dass die linearen Funktionen über $\mathbb{Z}/2^n\mathbb{Z}$ eine universelle Hashklasse bilden. Diese Ideen haben wir in Kapitel 5 weiter ausgebaut, um die erste echt exponentielle untere Schranke von $\Omega\big(2^{n/4}\big)$ für die FBDD-Größe des mittleren Bits der Multiplikation zu beweisen. Entscheidend war auch für diesen Beweis ein Überdeckungslemma für universelle Hashklassen (Lemma 5.4), das sich auf die Multiplikation übertragen ließ (Lemma 5.5).

In den Kapiteln 6 und 7 haben wir uns weniger mit den Eigenschaften der Multiplikation beschäftigt, als viel mehr damit, wie sich die bereits in den früheren Kapiteln gewonnenen Erkenntnisse für den Beweis unterer Schranken in allgemeineren Branchingprogrammmodellen ausnutzen lassen. Durch die Entwicklung neuer Beweistechniken konnten wir sowohl für semantische $(1,+k)$-BPs als auch für $(\vee,k)$-FBDDs untere Schranken von $2^{\Omega(n/k)}$ für die Komplexität der Multiplikation herleiten. Dies bedeutet, dass diese Funktion sich für $k=o(n/\log n)$ nicht durch semantische $(1,+k)$-BPs oder $(\vee,k)$-FBDDs polynomieller Größe darstellen lässt. Bis heute sind selbst für syntaktische $(1,+k)$-BPs mit $k=\Omega(n/\log n)$ für keine explizit definierte Funktion superpolynomielle untere Schranken bekannt. Bei $(\vee,k)$-FBDDs gibt es natürlich exponentielle untere Schranken für beliebig große k, aber es gibt für $k=\Omega(n/\log n)$ keinen Beweis einer superpolynomiellen unteren Schranke für eine Folge von Funktionen, der nicht auch gleichzeitig eine superpolyonomielle untere Schranke für allgemeine nichtdeterministische FBDDs impliziert.

Neben dem grundsätzlichen Interesse daran, möglichst viel über eine so fundamentale Funktion wie die Multiplikation zu erfahren, hatten wir eingangs die Analyse der Komplexität von wichtigen Funktionen in Branchingprogrammmodellen vor allem mit zwei weiteren Argumenten motiviert: Zum einen kann man durch den Beweis unterer Schranken zu Erkenntnissen über wichtige Eigenschaften der Funktion gelangen und zum anderen müssen ggf. neue Beweistechniken entwickelt oder bekannte Beweistechniken verbessert werden, damit sie auf die speziellen Eigenschaften der betrachteten Funktion angewendet werden können. Beides ist hier am Beispiel der Multiplikation erfolgt.

Die Ideen zum Beweis unterer Schranken für $(1, +k)$-BPs und $(\vee, k)$-FBDDs, die wir hier entwickelt haben, konnten außerhalb dieser Arbeit noch auf eine ähnliche Weise formalisiert und auf andere Funktionen angewendet werden (Woelfel, 2002a). Dies hat zur Definition von sog. „k-wise ℓ-mixed“ Funktionen geführt, für die eine Komplexität von $2^{\Omega(\ell)}$ sowohl bei semantischen $(1, +k)$-BPs als auch bei nichtdeterministischen FBDDs mit höchstens k Knoten nachgewiesen wurde. Dadurch konnte z.B. bewiesen werden, dass es Folgen explizit definierter Funktionen gibt, die polynomiell große FBDDs mit k nichtdeterministischen Knoten, aber keine polynomiell großen FBDDs mit $k-1$ nichtdeterministischen Knoten haben (für $k = o(n^{1/3}/\log^{2/3} n)$). Zudem konnten die zuvor bekannten Hierarchieresultate für semantische $(1, +k)$-BPs (vgl. Kapitel 2, S. 7) verbessert werden.

Da die Erkenntnisse über die Multiplikation aus den Kapiteln 4 und 5 recht allgemein sind, konnten sie auch bei anderen, in dieser Arbeit nicht behandelten Branchingprogrammmodellen zum Beweis unterer Schranken verwendet werden. So haben Bollig, Waack und Woelfel (2002) sowie Bollig und Woelfel (2002) unter Verwendung der hier vorgestellten Eigenschaften der Multiplikation exponentielle untere Schranken für $\mathrm{MUL}_{n-1,n}$ bei sog. well-structured graphgesteuerten nichtdeterministischen FBDDs bewiesen (bei Bollig, Waack und Woelfel handelt es sich dabei um XOR-Nichtdeterminismus). Die Eigenschaft „well-structured“ schränkt zwar graphgesteuerte FBDDs weiter ein, aber das Modell ist dennoch echt allgemeiner als das der deterministischen FBDDs. Schließlich konnten kürzlich Sauerhoff und Woelfel (2003) eine untere Schranke von $2^{\Omega(n \cdot k^{-2} 3^{-4k})}$ für die Größe von nichtdeterministischen (syntaktischen) k-BPs beweisen, die das mittlere Bit der Multiplikation darstellen. Für den Beweis dieses Ergebnisses waren jedoch die hier vorgestellten Aussagen über die Multiplikation zu schwach und es mussten weiterführende Eigenschaften der Funktion ausgenutzt werden. Die entscheidende Idee, diese Eigenschaften aus der Tatsache abzuleiten, dass die multiplikative Hashklasse universell ist, wurde jedoch auch dort wieder aufgegriffen.

Sieht man von nichtdeterministischen oder unbedeutenderen Branchingprogrammmodellen ab, bleibt also bzgl. der Komplexität des mittleren Bits der Multiplikation vor allem eine Frage offen: Welche Länge muss ein polynomiell großes Branchingprogramm für die Funktion $\mathrm{MUL}_{n-1,n}$ mindestens haben? Für eine Variante von Branchingprogrammen, bei denen an einem Knoten statt nur einer Variable ein ganzer Block von Variablen gelesen werden muss, finden sich bei Sauerhoff und Woelfel (2003) auch exponentielle untere Schranken für längenbeschränkte Branchingprogramme. Diese Schranken lassen sich aber nicht auf das hier verwendete binäre Modell übertragen, sodass trotz der exponentiellen unteren Schranke für k-BPs noch offen ist, ob es vielleicht ein Branchingprogramm der Länge $4n$ für $\mathrm{MUL}_{n-1,n}$ mit polynomieller Größe gibt. Vielleicht liegt aber auch die wahre Mindestlänge eines polynomiell großen Branchingprogramms für $\mathrm{MUL}_{n-1,n}$ in der Nähe der (mit der Schulmethode) trivialen oberen Schranke von $O(n^2)$? Neuere Überlegungen lassen vermuten, dass dem nicht so ist. So gibt es einen (noch unfertigen) Ansatz, ein randomisiertes k-BP mit $k = (\log n)^{O(1)}$ polynomieller Größe zu konstruieren, dass mit polynomiell kleiner Fehlerwahrscheinlichkeit die Funktion $\mathrm{MUL}_{n-1,n}$ berechnet. Natürlich würde ein Resultat in dieser Richtung noch nichts über deterministische Branchingprogramme aussagen – hier sind sicherlich noch weitere Untersuchungen angebracht.

Auch wenn noch einige Fragen offen sind, so ist man doch inzwischen mit dem Beweis großer unterer Schranken für $\mathrm{MUL}_{n-1,n}$ fast bei den allgemeinsten Modellen erfolgreich, für die sich überhaupt superpolynomielle untere Schranken für explizit definierte Funktionen beweisen lassen. Für allgemeinere Branchingprogrammmodelle müssen also erst grundsätzlich Techniken zum Beweis unterer Schranken entwickelt werden, bevor man der Antwort auf die Frage „wie schwer ist es zu multiplizieren“ wieder etwas näher kommen kann.

Anhang

Nachtrag zum Beweis von Lemma 4.8

Wir beweisen hier die Aussagen (b)-(d) aus Lemma 4.8. Die Notation wird aus dem Beweis von Aussage (a) auf S. 27f. übernommen.

Aussage (b): Aus den Gleichungen (4.3) und (4.5) sowie der Voraussetzung $(y+k) \bmod 2^m = 2^{m-1}$ folgt

$$ax + az \equiv \big(y+k\big) \cdot 2^{n-m} + \delta + \tau \equiv 2^{n-1} + \delta + \tau \pmod{2^n}.$$

Da $\delta + \tau \in \mathbb{Z}_{2^{n-m}}$ ist, ist $\delta + \tau < 2^{n-1}$ und somit $(ax+az) \bmod 2^n \geq 2^{n-1}$. Also folgt $\langle a(x+z)\rangle_{n-1}^{n-1} = 1$. Wir zeigen nun $\langle a(x'+z)\rangle_{n-1}^{n-1} = 0$. Indem wir die Gleichungen (4.4) und (4.5) sowie $(y+k) \bmod 2^m = 2^{m-1}$ benutzen, erhalten wir

$$\begin{aligned} ax' + az &\equiv (y'+k) \cdot 2^{n-m} + \delta' + \tau \\ &\equiv \big(y + k - (y - y')\big) \cdot 2^{n-m} + \delta' + \tau \\ &\equiv 2^{n-1} + (y'-y) \cdot 2^{n-m} + \delta' + \tau \pmod{2^n}. \end{aligned} \tag{8.1}$$

Nach Voraussetzung ist $2 \leq (y-y') \bmod 2^m \leq 2^{m-1}$, woraus

$$2^{n-1} \leq \big((y'-y)\cdot 2^{n-m}\big) \bmod 2^n \leq 2^n - 2 \cdot 2^{n-m}$$

folgt. Wir setzen dies in (8.1) ein und erhalten mit $\delta' + \tau \in \big\{0, \ldots, 2 \cdot (2^{n-m} - 1)\big\}$

$$0 \leq (ax' + az) \bmod 2^n \leq 2^{n-1} - 2.$$

Also folgt $\langle a(x'+z)\rangle_{n-1}^{n-1} = 0$.

Aussage (c): Aus den Gleichungen (4.4) und (4.5) sowie der Voraussetzung $(y'+k) \bmod 2^m = 2^{m-1} - 2$ folgt

$$ax' + az \equiv \big(y'+k\big) \cdot 2^{n-m} + \delta' + \tau \equiv 2^{n-1} - 2 \cdot 2^{n-m} + \delta' + \tau \pmod{2^n}.$$

Wegen $\delta' + \tau \in \mathbb{Z}_{2^{n-m}}$ folgt $(ax'+az) \bmod 2^n < 2^{n-1}$ also $\langle a(x'+z)\rangle_{n-1}^{n-1} = 0$.

Wir zeigen nun $\langle a(x+z)\rangle_{n-1}^{n-1} = 1$. Indem wir die Gleichungen (4.3) und (4.5) sowie $(y'+k) \bmod 2^m = 2^{m-1} - 2$ benutzen, erhalten wir

$$\begin{aligned} ax + az &\equiv (y+k) \cdot 2^{n-m} + \delta + \tau \\ &\equiv \big(y' + k - (y' - y)\big) \cdot 2^{n-m} + \delta + \tau \\ &\equiv 2^{n-1} - 2 \cdot 2^{n-m} + (y-y') \cdot 2^{n-m} + \delta + \tau \pmod{2^n}. \end{aligned}$$

Mit der Voraussetzung $2 \le (y - y') \bmod 2^m \le 2^{m-1}$ und $\delta + \tau \in \big\{0, \dots, 2 \cdot (2^{n-m} - 1)\big\}$ folgt

$$2^{n-1} \;\le\; (ax + az) \bmod 2^n \;\le\; 2^n - 2.$$

Also ist $\langle a(x+z)\rangle_{n-1}^{n-1} = 1$.

Aussage (d): Aus den Gleichungen (4.4) und (4.5) sowie der Voraussetzung $(y' + k) \bmod 2^m = 2^m - 2$ folgt

$$ax' + az \;\equiv\; \big(y' + k\big) \cdot 2^{n-m} + \delta' + \tau \;\equiv\; 2^n - 2 \cdot 2^{n-m} + \delta' + \tau \pmod{2^n}.$$

Wegen $\delta', \tau \in \mathbb{Z}_{2^{n-m}}$ folgt

$$2^n - 2 \cdot 2^{n-m} \;\le\; (ax' + az) \bmod 2^n \;\le\; 2^n - 2,$$

also $\langle a(x'+z)\rangle_{n-1}^{n-1} = 1$.

Indem wir die Gleichungen (4.3) und (4.5) sowie $(y' + k) \bmod 2^m = 2^m - 2$ benutzen, erhalten wir

$$\begin{aligned} ax + az \;&\equiv\; (y + k) \cdot 2^{n-m} + \delta + \tau \\ &\equiv\; \big(y' + k - (y' - y)\big) \cdot 2^{n-m} + \delta + \tau \\ &\equiv\; 2^n - 2 \cdot 2^{n-m} + (y - y') \cdot 2^{n-m} + \delta + \tau \pmod{2^n}. \end{aligned}$$

Mit der Voraussetzung $2 \le (y - y') \bmod 2^m \le 2^{m-1}$ und $\delta + \tau \in \big\{0, \dots, 2 \cdot (2^{n-m} - 1)\big\}$ folgt

$$0 \;\le\; (ax + az) \bmod 2^n \;\le\; 2^{n-1} - 2.$$

Also ist $\langle a(x+z)\rangle_{n-1}^{n-1} = 0$. ∎

Symbolverzeichnis

$\oplus$	XOR-Operation, Seite 8.
$\langle\rangle_\ell^k$	Für $z \in \mathbb{Z}_{2^n}$ ist $\langle z\rangle_\ell^k$ der String, der aus den Bits an den Positionen $\ell, \ldots, k$ der Binärdarstellung von z besteht, Seite 10.
$\|\,\|$	Für $x \in \mathbb{B}_n$ ist $\|x\|$ die Interpretation von x als Zahl in $\mathbb{Z}_{2^n}$, Definition 2.5, Seite 10.
$(1, +k)$-BP	S. Definition 2.4, Seite 6.
$(\vee, k)$-FBDD	S. Definition 7.1, Seite 56.
$0^k, 1^k, *^k$	Für $c \in \{0, 1, *\}$ ist c^k die Folge $c \ldots c$ von k Symbolen c, Seite 11.
$\mathrm{ADD}_{i,n}$	Boolesche Funktion, die das i-te Ausgabebit der Summe zweier n-Bit Zahlen darstellt, Definition 2.6, Seite 10.
ADD_n	Boolesche Funktion, die alle Ausgabebits der Summe zweier n-Bit Zahlen darstellt, Definition 2.6, Seite 10.
$\mathbb{B}_n$	Die Menge der Binärdarstellungen von Zahlen aus $\mathbb{Z}_{2^n}$, Definition 2.5, Seite 10.
B_n	Abkürzung für $B_{n,1}$, Seite 3.
$B_{n,m}$	Klasse der booleschen Funktionen $f : \{0, 1\}^n \to \{0, 1\}^m$, Seite 3.
$\mathrm{BP}(f)$	Größe des minimalen Branchingprogramms für f, Definition 2.1, Seite 3.
k-BP	S. Definition 2.4, Seite 6.
BP	Abk. für Branchingprogramm, Definition 2.1, Seite 3.
$D(\alpha, \beta)$	Für zwei partielle Belegungen α, β ist dies die Menge der Variablen in denen sich α und β unterscheiden, Definition 6.1, Seite 45.
div	Ganzzahlige Division ohne Rest, Seite 16.
DIV_n	Funktion, die die n signifikantesten Bits des Quotienten zweier als Binärdarstellung gegebener Zahlen berechnet, Seite 12.
$f(M)$	Bild der Menge M unter der Funktion f, Seite 23.

f^{-1} .... Urbild der Abbildung f, also $f^{-1}(y) = \{x \mid f(x) = y\}$.

$f_{|\alpha}$ .... Subfunktion der Funktion f für die partielle Belegung α, Definition 4.5, Seite 26.

FBDD .... Free Binary Decision Diagram, Definition 2.3, Seite 5.

$h^m_{a,b}$ .... Funktion der multiplikativen Hashklasse mit Wertebereich $\mathbb{Z}_{2^m}$, Definition 3.9, Seite 18.

$\mathcal{M}_{n,m}$ .... Multiplikative Hashklasse mit Universum $\mathbb{Z}_{2^n}$ und Wertebereich $\mathbb{Z}_{2^m}$, Definition 3.9, Seite 18.

$\mathrm{MUL}_{i,n}$ .... Boolesche Funktion, die das i-te Ausgabebit des Produkts zweier n-Bit Zahlen darstellt, Definition 2.8, Seite 10.

MUL_n .... Boolesche Funktion, die alle Ausgabebits des Produkts zweier n-Bit Zahlen darstellt, Definition 2.8, Seite 10.

$\mathcal{P}(X, X')$ .... Menge der partiellen Belegungen von X mit Trägermenge X', Definition 4.5, Seite 26.

$\mathcal{T}(\alpha)$ .... Trägermenge einer partiellen Belegung α, Definition 4.5, Seite 26.

$\mathbb{Z}/n\mathbb{Z}$ .... Restklassenring der ganzen Zahlen modulo n.

$\mathbb{Z}_k$ .... Die Menge $\{0, \ldots, k-1\}$, Definition 2.5, Seite 10.

$\mathbb{Z}_k^*$ .... Menge der invertierbaren Elemente des Rings $\mathbb{Z}/k\mathbb{Z}$, Seite 18.

Literaturverzeichnis

F. Ablayev und M. Karpinski (1998). A lower bound for integer multiplication on randomized read-once branching programs. Technischer Bericht TR98-011, Electronic Colloquium on Computational Complexity.

M. Ajtai (1999). A non-linear time lower bound for boolean branching programs. In *Proceedings of the 40th Annual IEEE Symposium on Fountations of Computer Science*, S. 60–70.

S. B. Akers (1978). Binary decision diagrams. *IEEE Transactions on Computers*, Band C-27, S. 509–516.

N. Alon und W. Maass (1988). Meanders and their applications in lower bounds arguments. *Journal of Computer and System Sciences*, Band 37, S. 118–129.

A. Andersson, T. Hagerup, S. Nilsson und R. Raman (1995). Sorting in linear time? In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, S. 427–436.

A. Andreev, J. Baskakov, A. Clementi und J. Rolim (1999). Small pseudo-random sets yield hard functions: New tight explict lower bounds for branching programs. In *Proceedings of the 26th International Colloquium on Automata, Languages, and Programming*, LNCS 1644, S. 179–189.

V. Arvind und M. Mahajan (1996). Applications of universal hashing in complexity theory. In *Proceedings of the 6th National Seminar on Theoretical Computer Science*, S. 57–83.

M. Atici und D. R. Stinson (1996). Universal hashing and multiple authentication. In *Advances in Cryptology – CRYPTO '96*, LNCS 1109, S. 16–30.

P. Beame, M. Saks, X. Sun und E. Vee (2003). Time-space tradeoff lower bounds for randomized computation of decision problems. *Journal of the ACM*, Band 50, S. 169–179.

B. Bollig (2001). Restricted nondeterministic read-once branching programs and an exponential lower bound for integer multiplication. *RAIRO Theoretical Informatics and Applications*, Band 35, S. 149–162.

B. Bollig, S. Waack und P. Woelfel (2002). Parity graph-driven read-once branching programs and an exponential lower bound for integer multiplication. In *Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science*, S. 83–94.

B. Bollig und I. Wegener (1996). Read-once projections and formal circuit verification with binary decision diagrams. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 1046, S. 491–502.

B. Bollig und I. Wegener (1998). A very simple function that requires exponential size read-once branching programs. *Information Processing Letters*, Band 66, S. 53–57.

B. Bollig und I. Wegener (1999). Asymptotically optimal bounds for OBDDs and the solution of some basic OBDD problems. Technischer Bericht TR99-048, Electronic Colloquium on Computational Complexity (vgl. auch Bollig und Wegener, 2000).

B. Bollig und I. Wegener (2000). Asymptotically optimal bounds for OBDDs and the solution of some basic OBDD problems. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming*, S. 187–198.

B. Bollig und P. Woelfel (2001). A read-once branching program lower bound of $\Omega(2^{n/4})$ for integer multiplication using universal hashing. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, S. 419–424.

B. Bollig und P. Woelfel (2002). A lower bound technique for nondeterministic graph-driven read-once-branching programs and its applications. In *Mathematical Foundations of Computer Science: 27th International Symposium*, LNCS 2420, S. 131–142.

A. Borodin, A. Razborov und R. Smolensky (1993). On lower bounds for read-k-times branching programs. *Computational Complexity*, Band 3, S. 1–18.

R. E. Bryant (1985). Symbolic manipulation of boolean functions using a graphical representation. *Proceedings of the 22nd ACM/IEEE Design Automation Conference*, S. 688–694.

R. E. Bryant (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, Band C-35, S. 677–691.

R. E. Bryant (1991). On the complexity of VLSI implementations and graph representations of boolean functions with applications to integer multiplication. *IEEE Transactions on Computers*, Band 40, S. 205–213.

J. L. Carter und M. N. Wegman (1979). Universal classes of hash functions. *Journal of Computer and System Sciences*, Band 18, S. 143–154.

A. Cobham (1966). The recognition problem for the set of perfect squares. In *Conference Record of 1966 Seventh Annual Symposium on Switching and Automata Theory*, S. 78–87.

M. Dietzfelbinger (1996). Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, S. 569–580.

M. Dietzfelbinger, T. Hagerup, J. Katajainen und M. Penttonen (1997). A reliable randomized algorithm for the closest-pair problem. *Journal of Algorithms*, Band 25, S. 19–51.

J. Gergov (1994). Time-space tradeoffs for integer multiplication on various types of input oblivious sequential machines. *Information Processing Letters*, Band 51, S. 265–269.

J. Gergov und C. Meinel (1994). Efficient analysis and manipulation of OBDDs can be extended to FBDDs. *IEEE Transactions on Computers*, Band 43, S. 1197–1209.

O. Giel (2001). Branching program size is almost linear in formula size. *Journal of Computer and System Sciences*, Band 63, S. 222–235.

G. D. Hachtel und F. Somenzi (1996). *Logic Synthesis and Verification Algorithms*. Kluwer Academic Publishers.

K. Iwama und H. Morizumi (2002). An explicit lowser bound of $5n - o(n)$ for boolean circuits. In *Mathematical Foundations of Computer Science: 27th International Symposium*, LNCS 2420, S. 353–364.

J. Jain, J. Bitner, D. S. Fussell und J. A. Abraham (1992). Functional partitioning for verification and related problems. In *Brown MIT VLSI Conference*, S. 210–226.

S. Jukna und A. Razborov (1998). Neither reading few bits twice nor reading illegally helps much. *Discrete Applied Mathematics*, Band 85, S. 223–238.

C. Y. Lee (1959). Representation of switching circuits by binary-decision programs. *The Bell Systems Technical Journal*, Band 38, S. 985–999.

Y. Mansour, N. Nisan und P. Tiwari (1993). The computational complexity of universal hashing. *Theoretical Computer Science*, Band 107, S. 121–133.

W. Masek (1976). A fast algorithm for the string-editing problem and decision graph complexity. M. Sc. Thesis. MIT, Cambridge, MA.

Y. Matias und U. Vishkin (1991). On parallel hashing and integer sorting. *Journal of Algorithms*, Band 12, S. 573–606.

C. Meinel (1990). Polynomial size ω-branching programs and their computational power. *Information and Computation*, Band 85, S. 163–182.

S. Minato (1996). *Binary Decision Diagrams and Applications for VLSI CAD*. Kluwer Academic Publishers.

E. I. Nečiporuk (1966). A Boolean function. *Soviet Mathematics Doklady*, Band 7, S. 999–1000.

E. A. Okol'nishnikova (1993). On lower bounds for branching programs. *Siberian Advances in Mathematics*, Band 3, S. 152–166.

S. Ponzio (1995). A lower bound for integer multiplication with read-once branching programs. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, S. 130–139.

S. Ponzio (1998). A lower bound for integer multiplication with read-once branching programs. *SIAM Journal on Computing*, Band 28, S. 798–815.

M. Sauerhoff (2003). Guess-and-verify versus unrestricted nondeterminism for OBDDs and one-way turing machines. *Journal of Computer and System Sciences*, Band 66, S. 473–495.

M. Sauerhoff, I. Wegener und R. Werchner (1999). Relating branching program size and formula size over the full binary basis. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 1563, S. 57–67.

M. Sauerhoff und P. Woelfel (2003). Time-space tradeoff lower bounds for integer multiplication and graphs of arithmetic functions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, S. 186–195.

P. Savický und D. Sieling (2000). A hierarchy result for read-once branching programs with restricted parity nondeterminism. In *Mathematical Foundations of Computer Science: 25th International Symposium*, LNCS 1893, S. 650–659.

P. Savický und S. Žák (1997a). A hierarchy for $(1, +k)$-branching programs with respect to k. In *Mathematical Foundations of Computer Science: 22nd International Symposium*, LNCS 1295, S. 478–487.

P. Savický und S. Žák (1997b). A lower bound on branching programs reading some bits twice. *Theoretical Computer Science*, Band 172, S. 293–301.

P. Savický und S. Žák (2000). A read-once lower bound and a $(1, +k)$-hierarchy for branching programs. *Theoretical Computer Science*, Band 238, S. 347–362.

A. Schönhage und V. Strassen (1971). Schnelle Multiplikation großer Zahlen. *Computing*, Band 7, S. 281–292.

C. E. Shannon (1949). The synthesis of two-terminal switching circuits. *The Bell Systems Technical Journal*, Band 28, S. 59–98.

D. Sieling (1996). New lower bounds and hierarchy results for restricted branching programs. *Journal of Computer and System Sciences*, Band 53, S. 79–87.

D. Sieling (2002). The complexity of minimizing and learning OBDDs and FBDDs. *Discrete Applied Mathematics*, Band 122, S. 263–282.

D. Sieling und I. Wegener (1993). NC-algorithms for operations on binary decision diagrams. *Parallel Processing Letters*, Band 48, S. 139–144.

D. Sieling und I. Wegener (1995). Graph driven BDDs – a new data structure for Boolean functions. *Theoretical Computer Science*, Band 141, S. 283–310.

J. Simon und M. Szegedy (1993). A new lower bound theorem for read-only-once branching programs and its applications. In *Advances in Computational Complexity Theory*, Band 13 von *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, S. 183–193. AMS.

J. S. Thathachar (1998). On separating the read-k-times branching program hierarchy. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, S. 653–662.

I. Wegener (1987). *The Complexity of Boolean Functions*. Wiley-Teubner, erste Auflage.

I. Wegener (1988). On the complexity of branching programs and decision trees for clique functions. *Journal of the ACM*, Band 35, S. 461–471.

I. Wegener (1993). Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions. *Information Processing Letters*, S. 85–87.

I. Wegener (2000). *Branching Programs and Binary Decision Diagrams - Theory and Applications*. SIAM.

M. N. Wegman und J. L. Carter (1979). New classes and applications of hash functions. In *Proceedings of the 20th Annual IEEE Symposium on Fountations of Computer Science*, S. 175–182.

P. Woelfel (1999). Efficient strongly universal and optimally universal hashing. In *Mathematical Foundations of Computer Science: 24th International Symposium*, LNCS 1672, S. 262–272.

P. Woelfel (2000). Klassen universeller Hashfunktionen mit ganzzahliger Arithmetik. Diplomarbeit, Universität Dortmund.

P. Woelfel (2001). New bounds on the OBDD-size of integer multiplication via universal hashing. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 2010, S. 563–574.

P. Woelfel (2002a). A lower bound technique for restricted branching programs and applications. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 2285, S. 431–442.

P. Woelfel (2002b). On the complexity of integer multiplication in branching programs with multiple tests and in read-once branching programs with limited nondeterminism. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, S. 80–89.

B. Yang, Y.-A. Chen, R. E. Bryant und D. R. O'Hallaron (1998). Space- and time-efficient BDD construction via working set control. In *Proceedings of the Asia South-Pacific Design Automation Conference*, S. 423–432.

S. Žák (1984). An exponential lower bound for one-time-only branching programs. In *Mathematical Foundations of Computer Science: 11th International Symposium*, LNCS 176, S. 562–566.

S. Žák (1995). A superpolynomial lower bound for $(1,+k(n))$-branching programs. In *Mathematical Foundations of Computer Science: 20th International Symposium*, LNCS 969, S. 319–325.