technische universität
dortmund

Diplomarbeit

Analyse der BDD-Komplexität des höchstwertigen Bits der Multiplikation

Marc Gillé

9. Februar 2011

Gutachter:
Prof. Dr. Beate Bollig
Dr. André Gronemeier

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl II - Effiziente Algorithmen und Komplexitätstheorie
http://ls2-www.cs.tu-dortmund.de/

Ich danke meiner Familie, die mir das Informatik-Studium ermöglicht hat. Ich danke Alexander Munteanu und Christian Wiener für die schöne gemeinsame Studienzeit, für das Korrekturlesen und für die vielen Anregungen und Hilfestellungen. Darüber hinaus bedanke ich mich bei Dr. André Gronemeier für die Zweitbegutachtung dieser Diplomarbeit und für die vielen hilfreichen Verbesserungsvorschläge. Ein besonderer Dank geht an Prof. Dr. Beate Bollig für die erstklassige Betreuung und für die vielen interessanten und weiterführenden Gespräche, die mir bei der Erstellung der Diplomarbeit sehr geholfen haben. Zu guter Letzt danke ich meiner Freundin Anna Bury, die mich immer unterstützt und motiviert hat.

Inhaltsverzeichnis

1 Einleitung

Für die Informatik ist es von großer Bedeutung, die Komplexität von fundamentalen arithmetischen Funktionen zu bestimmen. Schließlich kommen sie in fast jedem Algorithmus vor und beeinflussen somit die Laufzeit von Algorithmen. Außerdem besitzt jeder Prozessor Befehle zum Addieren, Subtrahieren, Multiplizieren und Dividieren. Daher besteht ein großes Interesse zu wissen, wie schnell diese arithmetischen Funktionen berechnet werden können oder wie groß Schaltkreise für diese Funktionen sind.

In dieser Diplomarbeit werden wir uns mit der Multiplikation beschäftigen. Aus der Schule ist uns mit der schriftlichen Multiplikation bereits ein Algorithmus zur Berechung der Mutliplikation von zwei Zahlen mit n Stellen bekannt. Dieser Algorithmus benötigt $\Theta(n^2)$ arithmetische Operationen. Es sind aber noch bessere (dafür aber komplexere) Algorithmen für die Multiplikation bekannt: Bereits 1971 wurde der Algorithmus von Schönhage-Strassen [SS71] mit einer Laufzeit von $O(n \log n \log\log n)$ entworfen. Dieser Algorithmus war für mehr als 35 Jahre der schnellste bekannte Algorithmus zum Multiplizieren. 2007 veröffentlichte M. Fürer [Für07] eine Verbesserung des Algorithmus von Schönhage-Strassen mit einer Laufzeit[1] von $(n \log n)2^{O(\log^* n)}$. In [DKSS08] wurde ein Algorithmus angegeben, der eine Laufzeit von $O((n \log n)2^{O(\log^* n)})$ hat, und im Gegensatz zu dem Algorithmus von Fürer keine Arithmetik über den komplexen Zahlen sondern modulare Arithmetik verwendet. Bereits Schönhage und Strassen haben in [SS71] vermutet, dass der beste (bisher unbekannte) Algorithmus eine Laufzeit von $O(n \log n)$ hat. Das Resultat von Fürer bestärkt diese Vermutung. Bis heute ist es aber ein offenes Problem, ob ein Algorithmus für die Multiplikation mit einer asymptotischen Laufzeit von $O(n \log n)$ existiert.

Obere Schranken für die Schaltkreiskomplexität von der Multiplikation lassen sich gut

[1] $\log^* n = \min\{i \geq 0 \mid \log^i n \leq 1\}$

mit Hilfe der oben genannten Algorithmen zeigen. So liefert das Resultat von Fürer ein Schaltkreis der Größe $(n \log n)2^{O(\log^* n)}$ (siehe [Für07]). Dahingegen sind gute untere Schranken für allgemeine boolesche Schaltkreise mit den heutigen Techniken sehr schwierig zu zeigen. So sind die besten unteren Schranke für explizit definierte Funktionen[2] „nur“ linear in der Eingabelänge. Daher betrachtet man eingeschränkte Schaltkreismodelle, um dafür bessere untere Schranken zeigen zu können. Die Komplexitätsklasse AC^i mit $i \geq 0$ beinhaltet alle booleschen Funktionen, die mit Schaltkreisen der Größe $O(poly(n))$ und Tiefe $O(\log^i n)$ berechnet werden können, die AND- und OR-Bausteine mit unbeschränktem Eingangsgrad und NOT-Bausteine benutzen dürfen. In der Komplexitätsklasse NC^i mit $i > 0$ liegen Funktionen, die mit Schaltkreisen der Größe $O(poly(n))$ und Tiefe $O(\log^i n)$ berechnet werden können und die Schaltkreise dürfen die Bausteine AND, OR mit Eingangsgrad 2 und NOT benutzen. Als Threshold-Schaltkreis bezeichnen wir Schaltkreise, die ein zusätzlichen Threshold-Baustein verwenden dürfen, der Eins als Ergebnis liefert, wenn die Eingabe eine feste Anzahl an Einsen überschreitet. Die Komplexitätsklasse $TC^{0,i}$ besteht aus den Funktionen, die mit Threshold-Schaltkreisen der Größe $O(poly(n))$ und Tiefe i berechnet werden können. Aus dem Schönhage-Strassen-Algorithmus folgt, dass die Multiplikation in NC^1 liegt. Die Multiplikation ist auch in der Klasse $TC^{0,3}$ [SR94] aber nicht in der Klasse $TC^{0,2}$ [HMP+87]. Es ist auch bekannt, dass die Multiplikation nicht in AC^0 liegt.

Ein anderes wichtiges Berechnungsmodell für boolesche Funktionen sind sogenannte *Branchingprogramme* bzw. *Binary Decision Diagrams* (BDDs). Die Größe von einem minimalen BDD für eine Funktion f liegt zwischen der Schaltkreisgröße und der Formelgröße für f. Der Logarithmus der Größe des BDDs entspricht ungefähr dem Platzbedarf einer nichtuniformen Turingmaschine für die Funktion f (siehe [Weg00] für eine detaillierte Einführung in BDDs). Auch für allgemeine BDDs können keine guten unteren Schranken für explizit definierte Funktionen gezeigt werden, aber für eingeschränkte Modelle – insbesondere für *Ordered Binary Decision Diagrams* (OBDDs) – stehen sehr gute Methoden zur Verfügung, um große untere Schranken zu beweisen. Zusätzlich sind für OBDDs viele

[2] Wir nennen eine Funktion $f : \{0,1\}^n \to \{0,1\}$ explizit definiert, wenn f eine gewisse Struktur hat, z.B. wenn $f^{-1}(1) \in NP$ ist. Es kann mit einem Abzählargument gezeigt werden, dass fast alle Funktionen eine hohe Schaltkreiskomplexität haben (wenn wir wirklich *alle* Funktionen betrachten). In [Weg87] Kapitel 5.6 findet man eine ausführliche Behandlung der Bezeichnung „explizit definiert“.

praktische Operationen effizient durchführbar (siehe [Bry86]). Daher finden sich OBDDs in zahlreichen Anwendungen wie z.B. dem Model-Checking und der Schaltkreisverifikation wieder.

Für OBDDs sind viele schwierige Funktionen bekannt. So hat Bryant in [Bry91] gezeigt, dass ein OBDD für das *mittlere Bit* der Multiplikation exponentielle Größe haben muss. Mit dem mittleren Bit bezeichnen wir das Bit im Produkt $x \cdot y$ mit der Wertigkeit $n-1$, wobei x und y Binärzahlen der Länge n sind. Über die Komplexität des mittleren Bits ist vieles bekannt. In [Woe05] hat P. Woelfel die aktuell beste untere Schranke von $\Omega(2^{n/2})$ für die Größe von deterministischen OBDDs für das mittlere Bit der Multiplikation gezeigt. Für ein allgemeineres BDD-Modell – den sogenannten *Free Binary Decision Diagrams* (FBDD) – wurde in [BW01] eine untere Schranke von $\Omega(2^{n/4})$ bewiesen. Beide Beweise für die unteren Schranken nutzen Methoden und Resultate für universelle Hashklassen. Auch wenn eine begrenzten Nutzung von Nichtdeterminismus oder das mehrfache Testen von Variablen erlaubt ist, können exponentielle untere Schranken gezeigt werden [Woe02]. Die Methoden für die Beweise der unteren Schranken für das mittlere Bit sind sogar so stark, dass für allgemeine BDDs eine untere Schranke von $\Omega(n^{3/2}/\log n)$ gezeigt werden kann [WW05].

In der Diplomarbeit werden wir uns mit einem anderen Ergebnisbit beschäftigen: dem *höchstwertigen Bit* der Multiplikation. D.h. das Bit im Produkt von $x \cdot y$ mit der Wertigkeit $2n-1$. Im Gegensatz zu dem mittleren Bit ist die Komplexität des höchswertigen Bits für viele BDD-Modelle noch unklar. In [Saw06] wurde ein Zusammenhang zwischen der Platzkomplexität von symbolischen Graphalgorithmen[3] für Erreichbarkeitsprobleme und der deterministischen OBDD-Komplexität des höchstwertigen Bits bewiesen. Es wurde auch für einen sehr speziellen Fall eine exponentielle untere Schranke für die deterministische OBDD-Größe für das höchstwertige Bit gezeigt. In [Bol10] hat B. Bollig die bisher beste exponentielle untere Schranke von $\Omega(2^{n/45})$ für alle deterministischen OBDDs bewiesen. Wir werden dieses Resultat verwenden, um für eine allgemeinere Klasse von OBDDs – den *randomisierten OBDDs* – eine exponentiell große untere Schranke für die Größe zu beweisen. Aus diesem Ergebnis können wir leicht folgern, dass auch randomisierte OBDDs

[3] In symbolischen Graphalgorithmen werden Graphen durch die charakteristische Funktion ihrer Kantenmenge beschrieben, wobei die Knoten binär kodiert sind und der Zugriff auf diese boolesche Funktion z.B. mit OBDDs erfolgt.

für das mittlere Bit der Multiplikation exponentielle Größe haben müssen. Dieses Ergebnis verbessert die bisherige untere Schranke für die Größe von randomisierten OBDDs für das mittlere Bit der Multiplikation von $2^{\Omega(n/\log n)}$ aus [Abl03]. Die randomisierte OBDD-Komplexität des höchstwertigen Bits der Multiplikation war bisher offen.

Die Diplomarbeit ist folgendermaßen aufgebaut: In Kapitel 2 erarbeiten wir die benötigten Grundlagen, die wir zum Beweis der unteren Schranken benötigen. Dazu führen wir erstmal formal die BDDs ein und definieren, was (randomisierte) OBDDs sind. In den nächsten beiden Unterabschnitten zeigen wir den Zusammenhang von der (randomisierten) *Kommunikationskomplexität* einer Funktion und der (randomisierten) OBDD-Größe. Im dritten Kapitel behandeln wir ausführlich ein Resultat über die randomisierte Kommunikationskomplexität der *Greater-Than*-Funktion aus [SV08] von P. Sen und S. Venkatesh. Die Grundlagen für dieses Resultat beschreiben wir im ersten Unterabschnitt des dritten Kapitels. Unter anderem bekommen wir einen Einblick in die Informationstheorie und beweisen das sogenannte *Minimax-Theorem* für randomisierte Kommunikationsprotokolle. Im zweiten Unterabschnitt beweisen wir das *Round-Elimination-Lemma* mit dessen Hilfe wir im letzten Unterabschnitt eine lineare untere Schranke für die randomisierte Kommunikationskomplexität der Greater-Than-Funktion zeigen. Im vierten Kapitel beweisen wir unser Hauptresultat dieser Diplomarbeit. Dazu zeigen wir zuerst, wie sich die Komplexität der einzelnen Ergebnisbits der Multiplikation mit einer geeigneten Reduktionsmethode untereinander übertragen lässt. In Abschnitt 4.3 konstruieren wir eine Reduktion der Greater-Than-Funktion auf die Berechnung des höchstwertigen Bits. Mit den Ergebnissen aus Kapitel 3 können wir dann eine exponentielle untere Schranke für die Größe von randomisierten OBDDs beweisen. Aus den Ergebnissen am Anfang des Kapitels 4 können wir im letzten Unterabschnitt diese Schranke auch auf das mittlere Bit der Multiplikation übertragen. Im letzten Kapitel fassen wir kurz unsere Ergebnisse zusammen und diskutieren offene Fragen bzgl. der BDD-Komplexität des höchstwertigen Bits der Multiplikation.

2 Grundlagen

In diesem Kapitel erläutern wir, was ein Binary Decision Diagram (BDD) bzw. ein (randomisiertes) OBDD (Ordered Binary Decision Diagram) ist. Um untere Schranken für die Größe von randomisierten OBDDs zu beweisen, brauchen wir den Begriff der Kommunikationskomplexität einer Funktion. Daher werden wir in einem Kapitel einige Begriffe aus der Kommunikationskomplexität erläutern. Im letzten Unterkapitel gehen wir auf den Begriff der Rechteckreduktion ein, was uns ähnlich zu der polynomiellen Reduktion in der NP-Vollständigkeitstheorie erlaubt, untere/obere Schranken der OBDD-Größe von Funktionen zu übertragen und zeigen, wie die Kommunikationskomplexität und die OBDD-Größe sich zueinander verhalten.

2.1 Randomisierte OBDDs

Sei $f : \{0,1\}^n \to \{0,1\}$ eine boolesche Funktion. Ein Binary Decision Diagram (BDD) oder Branchingprogramm (BP) ist eine Darstellung von f, mit deren Hilfe $f(x)$ algorithmisch berechnet werden kann.

Definition 2.1.1 (BDD)**.** *Ein* BDD *ist ein gerichteter azyklischer Graph mit genau einem Wurzelknoten (Startknoten), der keine eingehenden Kanten besitzt, und zwei* Senken*, die keine ausgehenden Kanten besitzen. Jeder Knoten, der keine Senke ist, heißt* innerer *Knoten und hat genau zwei ausgehende Kanten. Jeder innere Knoten ist mit einer Variable x_i markiert und die ausgehenden Kanten sind mit* 0 *bzw.* 1 *markiert. Die Senken sind ebenfalls mit* 0 *bzw.* 1 *markiert. Jede Eingabe $x \in \{0,1\}^n$ definiert einen* Berechnungspfad *von dem Startknoten zu einer Senke, in dem an einem Knoten x_i die Kante gewählt wird, die mit dem Wert von x_i markiert ist. Das BDD berechnet eine Funktion $f : \{0,1\}^n \to \{0,1\}$, wenn für alle Eingaben $x \in \{0,1\}^n$ der Berechnungspfad von x an der Senke endet, die*

mit dem Funktionswert $f(x)$ markiert ist. Die Größe $|G|$ eines BDDs G ist die Anzahl der inneren Knoten von G.

In der Praxis sind viele Operationen für BDDs zu aufwendig. Z. B. sind der Erfüllbarkeitstest (d.h. existiert eine Eingabe, die auf 1 abgebildet wird) und das Minimierungsproblem (d.h. finde ein BDD mit minimaler Größer für f) NP-schwer (siehe [Weg00]). In [Bry86] hat Bryant für eine eingeschränkte Variante von BDDs, den sogenannten OBDDs, gezeigt, dass viele dieser praktischen Operationen effizient durchführbar sind. Somit haben sich OBDDs als eine nützliche Datenstruktur zur Repräsentation von booleschen Funktionen bewährt.

Definition 2.1.2 (OBDD)**.** *Sei π eine Permutation (Variablenordnung) auf $\{1,\dots,n\}$. Ein π-OBDD auf einer Variablenmenge $X = \{x_1,\dots,x_n\}$ ist ein BDD mit folgender zusätzlichen Anforderung:*

- *Wenn eine Kanten von einem Knoten mit Beschriftung x_i zu einem Knoten mit Beschriftung x_j existiert, dann muss $\pi(i) < \pi(j)$ gelten.*

Dies bedeutet auch, dass auf jedem Pfad vom Startknoten zu einer Senke jede Variable höchstens einmal getestet werden kann.

Mit Hilfe von Randomisierung können deterministische OBDDs auf naheliegende Weise erweitert werden. Die folgende Definition geht zurück auf Ablayev und Karpinski (siehe [AK96]).

Definition 2.1.3. *Sei $Z = \{z_1,\dots,z_r\}$ eine Menge von Zufallsvariablen und π eine Variablenordnung auf $X \cup Z$. Sei G ein π-OBDD für die Funktion $g : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}$. G heißt dann randomisiertes OBDD (mit unbeschränktem Fehler) für eine Funktion $f : \{0,1\}^n \to \{0,1\}$, falls für alle $x \in \{0,1\}^n$ gilt*

$$\Pr_{z\in\{0,1\}^r}[g(x,z) = f(x)] > \frac{1}{2},$$

wobei z uniform zufällig aus $\{0,1\}^r$ gezogen wird.

Der Fehler eines randomisierten OBDDs für eine Eingabe $x \in \{0,1\}^n$ ist $\Pr_{z\in\{0,1\}^r}[g(x,z) \neq f(x)]$. Analog zu randomisierten Turingmaschinen unterscheiden wir bei randomisierten OBDDs zwischen folgenden Fehlertypen:

- G ist ein randomisiertes OBDD für f mit *zweiseitigem Fehler* von höchstens ε mit $0 \leq \varepsilon < \frac{1}{2}$, falls für alle $x \in \{0,1\}^n$ gilt

$$\Pr[g(x,z) \neq f(x)] \leq \varepsilon$$

- G ist ein randomisiertes OBDD für f mit *einseitigem Fehler* von höchstens ε mit $0 \leq \varepsilon < 1$, falls für alle $x \in \{0,1\}^n$ folgendes gilt:

$$\text{Falls } f(x) = 0, \text{ dann gilt } \Pr[g(x,z) = 1] = 0.$$

$$\text{Falls } f(x) = 1, \text{ dann gilt } \Pr[g(x,z) = 0] \leq \varepsilon.$$

Wir interessieren uns für den Fall, dass ε konstant ist, und sprechen dann von *beschränktem Fehler*. Wenn die Fehlerwahrscheinlichkeit polynomiell nah an den trivialen Fehlerschranken ist, dann kann man durch probability amplification nur eine konstante Fehlerwahrscheinlichkeit erhalten, wenn das entstehende OBDD exponentielle Größe hat. Einen Beweis des folgenden Lemmas findet man in [Sau98].

Lemma 2.1.4 (probability amplification)**.** *Sei $f : \{0,1\}^n \to \{0,1\}$ eine Funktion.*

- *Sei $\varepsilon \in [0,1)$ und G ein randomisiertes π-OBDD für f mit einseitigem Fehler von höchstens ε. Dann kann ein randomisiertes π-OBDD H für f mit einseitigem Fehler von höchstens ε^m und Größe $|H| = O(|G|^m)$ konstruiert werden.*

- *Sei $\varepsilon \in [0, \frac{1}{2})$ und G ein randomisiertes π-OBDD für f mit zweiseitigem Fehler von höchstens ε. Weiter sei $0 \leq \varepsilon' \leq \varepsilon$. Dann kann ein randomisiertes π-OBDD H für f mit zweiseitigem Fehler von höchstens ε' und Größe $|H| = O(|G|^m)$ mit $m = O\left((\frac{1}{2} - \varepsilon)^{-2} \log \frac{1}{\varepsilon'}\right)$ konstruiert werden.*

Ist die Anzahl an Wiederholungen konstant, ist die Größe des resultierenden OBDDs H polynomiell in der Größe von G.

Eine naheliegende Erweiterung von OBDDs sind k-OBDDs, bei denen Variablen in bestimmter Art und Weise öfters getestet werden dürfen.

Definition 2.1.5. *Ein (randomisiertes) k-OBDD mit einer Variablenordnung π ist ein BDD, dessen Knoten in Ebenen $L_1, \ldots, L_k$ unterteilt werden können, so dass Kanten entweder zwischen Knoten einer Ebene oder von Ebene L_i zu L_j mit $j > i$ verlaufen. Innerhalb*

jeder Ebene L_i halten die Kanten die übliche OBDD Ordnung bzgl. π ein. Im Falle eines randomisierten k-OBDDs darf keine Zufallsvariable mehrfach auf einem Pfad getestet werden.

2.2 Kommunikationskomplexität

In diesem Kapitel orientieren wir uns an den Begriffen und Definitionen von Kushilevitz und Nisan ([KN97]).

Das Problem in der Kommunikationskomplexitätstheorie besteht darin, dass zwei Parteien (i.A. als Alice und Bob bezeichnet) eine Funktion $f : X \times Y \to \{0,1\}$ berechnen müssen, wobei Alice nur die Eingabe $x \in X$ und Bob die Eingabe $y \in Y$ kennt. D.h., Alice und Bob müssen im Allgemeinen miteinander kommunizieren, um den Funktionswert korrekt zu berechnen. In diesem Szenario ist die Komplexität einer Funktion gleich der Anzahl an Bits, die für die Kommunikation benötigt werden, um den Funktionswert zu berechnen.

Formaler definieren wir ein deterministisches Protokoll P für f, wobei Alice oder Bob mit der ersten Nachricht, die nur von der jeweiligen Eingabe abhängig ist, beginnt und abwechselnd weitere Nachrichten versendet werden, die von den bisherigen Nachrichten und der entsprechenden Eingabe abhängig sein dürfen. Als letzte Nachricht wird der Funktionswert $f(x,y)$ gesendet. Die Kommunikationskomplexität des Protokolls P ist die maximale Anzahl an kommunizierten Bits. Die deterministische Kommunikationskomplexität von f ist gleich der minimalen Kommunikationskomplexität eines deterministischen Protokolls.

Bei einem randomisierten Protokoll haben Alice und Bob noch Zugriff auf Zufallsbits, d.h., die Nachrichten hängen zusätzlich noch von den Zufallsbits ab und die Ausgabe des Protokolls ist eine Zufallsvariable. Wir unterscheiden bei randomisierten Protokollen zwischen *öffentlichen* und *privaten Zufall*. In Protokollen mit privatem Zufall (*private coins*) haben Alice und Bob ihre eigenen Zufallsbits und kennen nicht den Ausgang der Zufallsbits des anderen. Beim öffentlichen Zufall (*public coins*) haben Alice und Bob Zugriff auf eine gemeinsame Quelle von Zufallsbits und kennen somit den Ausgang aller im Protokoll benutzten Zufallsbits. Sei P ein randomisiertes Protokoll für f. Wir schreiben $P(x,y,r_A,r_B)$ als Ausgabe des Protokolls P bei Eingabe (x,y) und bei verwendeten Zufallsbit r_A für

Alice und r_B für Bob. Für eine Eingabe $(x, y) \in X \times Y$ definieren wir den Fehler des Protokolls mit

$$\varepsilon_P(x, y) := \Pr_{r_A, r_B}[P(x, y, r_A, r_B) \neq f(x, y)]$$

wobei die Zufallsbits unabhängig uniform zufällig gezogen werden. Das Maximum von $\varepsilon_P(x, y)$ über alle Eingaben (x, y) ist der (worst-case) Fehler des Protokolls P für f. Auch die Anzahl der Bits für die Kommunikation kann von den Zufallsvariablen abhängen, daher bezeichnen wir die Komplexität eines randomisierten Protokolls P als maximale Anzahl an ausgetauschten Bits über alle Eingaben und möglichen Zufallsbits. Mit $R_\varepsilon(f)$ bezeichnen wir die Komplexität von f als minimale Komplexität eines randomisierten Protokolls, welches f mit Fehler höchstens ε berechnet. Mit $R_\varepsilon^{pub}(f)$ bezeichnen wir die Komplexität von f bei Protokollen, die öffentlichen Zufall benutzen. Offensichtlich gilt $R_\varepsilon^{pub}(f) \leq R_\varepsilon(f)$, da aus einem Protokoll mit privatem Zufall und Zufallsbitstrings r_A und r_B ein Protokoll mit öffentlichem Zufall konstruiert werden kann, indem der gemeinsame Zufallsbitstring auf die Konkatenation von r_A und r_B gesetzt wird.

Eine weitere Eigenschaft von Protokollen, die insbesondere für den Zusammenhang zu der OBDD-Größe wichtig ist, ist die Anzahl an Runden eines Protokolls. In einem k-Runden Protokoll dürfen Alice und Bob sich für jede Eingabe höchstens k Nachrichten zusenden und danach muss der Funktionswert versendet werden. Insbesondere interessieren wir uns für 1-Runden Protokolle, d. h., z. B. Alice sendet eine einzige Nachricht an Bob, der dann direkt den Funktionswert ausgeben muss. Mit $R_\varepsilon^k(f)$ bezeichnen wir die minimale Komplexität eines k-Runden Protokolls, welches f mit Fehler höchstens ε berechnet. Im Spezialfall $k = 0$ muss der Spieler, der mit der ersten Nachricht anfängt, direkt den Funktionswert berechnen, ohne mit seinem Partner vorher Informationen ausgetauscht zu haben. Ein 0-Runden Protokoll kann für nicht triviale Funktionen keine Fehlerwahrscheinlichkeit kleiner als $1/2$ haben.

Beobachtung 2.2.1. *Sei $f : X \times Y \to \{0, 1\}$ eine Funktion mit $|X| \geq 2$ und $|Y| \geq 1$. Falls $x_0, x_1 \in X$ und $y \in Y$ mit $f(x_0, y) \neq f(x_1, y)$ existieren, dann kann ein 0-Runden Protokoll, bei dem Bob den Funktionswert ausgeben muss, nur eine Fehlerwahrscheinlichkeit von mindestens $1/2$ besitzen. Analog gilt die Aussage, wenn Alice die Nachricht senden muss und die Rollen von X und Y vertauscht sind.*

Angenommen es gibt ein Protokoll für f mit Fehler kleiner als $1/2$, dann muss die

Fehlerschranke sowohl für die Eingabe (x_0, y) als auch (x_1, y) gelten. Da Bob aber die beiden Fälle nicht unterscheiden kann, wird er für einen der beiden Fälle eine Fehlerwahrscheinlichkeit von mehr als 1/2 haben müssen (da $f(x_0, y) = 1 - f(x_1, y)$). Dies steht im Widerspruch zur Fehlerschranke. D.h., es bleibt Bob nichts besseres übrig als den Funktionswert durch einen Münzwurf zu entscheiden.

2.3 Untere Schranken für randomisierte OBDDs

In diesem Abschnitt wollen wir zwei Methoden zum Beweis von unteren Schranken für die randomisierte OBDD-Größe vorstellen. Wir zeigen, dass die randomisierte Kommunikationskomplexität einer Funktion in gewisser Weise eine untere Schranke für die randomisierte OBDD-Größe ist. D.h., wir sind an unteren Schranken für die randomisierte Kommunikationskomplexität von Funktionen interessiert. Wir definieren uns dazu eine Reduktionstechnik für Kommunikationsprotokolle, die uns ähnlich zu der polynomiellen Reduktion für Entscheidungsprobleme erlaubt, untere bzw. obere Schranken zu übertragen. Meistens sind wir nicht direkt in der Lage, eine untere Schranke für die Funktion, die mit einem randomisierten OBDD dargestellt werden soll, anzugeben. Mit Hilfe der Reduktion können wir dann zeigen, dass die gesamte Funktion mindestens so schwierig ist wie ein Subproblem und können für dieses Subproblem eine untere Schranke bestimmen. Der gebräuchliche Reduktionstyp für dieses Problem ist die sogenannte Rechteckreduktion.

Definition 2.3.1 (Rechteckreduktion)**.** *Seien $f : X_f \times Y_f \to \{0,1\}$ und $g : X_g \times Y_g \to \{0,1\}$ beliebige Funktionen. Das Paar (φ_1, φ_2) von Funktionen $\varphi_1 : X_f \to X_g$ und $\varphi_2 : Y_f \to Y_g$ heißt Rechteckreduktion von f auf g (kurz: Reduktion), falls*

$$g(\varphi_1(x), \varphi_2(y)) = f(x, y) \quad \text{für alle } (x, y) \in X_f \times Y_f.$$

Wir sagen dann f ist reduzierbar auf g.

Die Trennung der Reduktion in zwei Funktionen ist wichtig, um so die Schranken übertragen zu können. Im Falle, dass f auf g reduziert werden kann und es ein randomisiertes Protokoll für g gibt, können wir leicht mit Hilfe der Reduktion ein randomisiertes Protokoll für f angeben, in dem Alice ihre Eingabe für f mit φ_1 in eine Eingabe für g transformiert und Bob analog seine Eingabe mit φ_2 umwandelt. Dann befolgen die beiden das Protokoll

für g und erhalten so mit der gleichen Anzahl an versendeten Bits das Ergebnis für f. Falls nun φ_1 oder φ_2 von der anderen Eingabe abhängig gewesen wäre, könnte das alte Protokoll nicht ohne weiteres übernommen werden.

Beobachtung 2.3.2. *Seien f und g wie in 2.3.1 definiert und f reduzierbar auf g. Angenommen, es existiert ein randomisiertes Protokoll P für g. Dann existiert auch ein randomisiertes Protokoll für f mit den gleichen Eigenschaften wie P, d.h. insbesondere mit der gleichen Anzahl an versendeten Bits und der gleichen Fehlerwahrscheinlichkeit. Also gilt z.B. $R_\varepsilon^k(f) \leq R_\varepsilon^k(g)$ für alle $0 \leq \varepsilon < \frac{1}{2}$ und alle $k \in \mathbb{N}_0$.*

Auf den ersten Blick ist nicht direkt ersichtlich, wo die Kommunikationskomplexität für die OBDD-Größe eine Rolle spielt. Angenommen, wir haben ein π-OBDD G für f für eine Variablenordnung π. Wir können nun mit Hilfe von G ein Kommunikationsprotokoll für die *partitionierte Version* $f^{\pi,p}$ mit $1 \leq p \leq n-1$ angeben, die wie folgt definiert ist. (siehe [Sau98])

Definition 2.3.3. *Sei $f : \{0,1\}^n \to \{0,1\}$ eine beliebige Funktion definiert auf einer Variablenmenge $X = \{x_1, ..., x_n\}$. Sei π eine Variablenordnung auf X. Sei $1 \leq p \leq n-1$ und $L := \{x_{\pi(1)}, ..., x_{\pi(p)}\}$ und $R := \{x_{\pi(p+1)}, ..., x_{\pi(n)}\}$. Definiere die Funktion $f^{\pi,p} : \{0,1\}^p \times \{0,1\}^{n-p} \to \{0,1\}$. Wir bezeichnen $f^{\pi,p}$ als die partitionierte Version von f in Bezug auf π und p.*

Mit Hilfe von G kann nun folgendes Protokoll für $f^{\pi,p}$ angegeben werden:

- Alice verfolgt den Pfad innerhalb von G für ihren Teil der Eingabe. Dies ist möglich, da ihre Variablen vor den Variablen von Bob in G getestet werden.
- Alice sendet Bob die Knotennummer des Knotens, in dem der Pfad endet.
- Von diesem Knoten aus verfolgt Bob für seine Eingabe den Pfad weiter, bis er an einer Senke angelangt ist. Bob sendet die Markierung der Senke als Funktionswert.

Da G für jede Eingabe die Variablenordnung π einhalten muss, können Alice und Bob so den Funktionswert für jede Eingabe korrekt berechnen. Die Anzahl an ausgetauschten Bits ist $\lceil \log |G| \rceil$. In [Sau98] wurde dieser Zusammenhang zwischen Kommunikationskomplexität und OBDD-Größe allgemein für randomisierte k-OBDDs bewiesen.

Lemma 2.3.4. *Sei $g : \{0,1\}^n \to \{0,1\}$ eine Funktion und π eine Variablenordnung auf $X = \{x_1, \ldots, x_n\}$. Angenommen, es gibt eine Funktion $f : X \times Y \to \{0,1\}$ und einen Parameter $1 \leq p \leq n-1$, so dass f reduziert werden kann auf $g^{\pi,p}$. Sei G ein randomisiertes k-OBDD mit Variablenordung π und zweiseitigem Fehler höchstens ε, das g repräsentiert. Dann gilt*

$$\lceil \log |G| \rceil \geq R_\varepsilon^{2k-1}(f)/(2k-1)$$

In [BW96] wurde ein weiteres Reduktionskonzept eingeführt, das insbesondere für deterministische BDD-Modelle benutzt werden kann, um die Komplexität von Funktionen zu vergleichen.

Definition 2.3.5 (Read-Once-Projektion)**.** *Seien $f = (f_n)_{n\in\mathbb{N}}$ und $g = (g_n)_{n\in\mathbb{N}}$ zwei Folgen von booleschen Funktionen, d.h. $f_n : \{0,1\}^n \to \{0,1\}$ und $g_n : \{0,1\}^n \to \{0,1\}$. Dann ist f eine* Read-Once-Projektion *von g, falls es ein Polynom $p(n)$ gibt, so dass für alle $n \in \mathbb{N}$ und für alle Eingaben $x \in \{0,1\}^n$ für f eine entsprechende Eingabe $y \in \{0,1\}^{p(n)}$ für g existiert, so dass*

1. *$f_n(x_0, \ldots, x_{n-1}) = g_{p(n)}(y_0, \ldots, y_{p(n)-1})$ und*
2. *$y_i \in \{0, 1, x_0, \overline{x_0}, \ldots, x_{n-1}, \overline{x_{n-1}}\}$ für alle $0 \leq i < p(n)$ gilt und*
3. *falls $y_i \in \{x_k, \overline{x_k}\}$, dann folgt $y_j \notin \{x_k, \overline{x_k}\}$ für alle $i \neq j$ (Read-Once-Eigenschaft).*

Wir schreiben dann $f \leq_{rop} g$ oder $f_n \leq_{rop} g_{p(n)}$.

Für viele BDD-Varianten lassen sich analog zu der Rechteckreduktion mit Hilfe der Read-Once-Projektion die BDD-Größe zweier Funktionen vergleichen. Angenommen es gilt $f \leq_{rop} g$. Wenn es nun z.B. ein deterministisches OBDD G für die Funktion g_n gibt, dann können wir das OBDD wie folgt transformieren, damit es die Funktion f_n berechnet.

1. Wir löschen die Variablenknoten in G, die in der Projektion konstant auf $c \in \{0,1\}$ gesetzt sind, und verbinden die Vorgängerknoten mit dem c-Nachfolger des gelöschten Knotens.
2. Für alle negierten Variablen vertauschen wir die Markierung der ausgehenden Kanten des Variablenknotens.

Da f eine Read-Once-Projektion von g ist, erhält man ein OBDD, das die Funktion f_n berechnet und nicht größer als das OBDD G ist. Diese Eigenschaft können wir auch für randomisierte OBDDs nachweisen.

Beobachtung 2.3.6. *Sei $f_n \leq_{rop} g_{p(n)}$. Falls es ein randomisiertes OBDD G mit Fehler ε gibt, das g berechnet, dann existiert auch ein randomisiertes OBDD F mit Fehler ε für f und es gilt $|F| \leq |G|$.*

Die Transformation des randomisierten OBDDs G geht genauso wie zuvor. Sowohl die Konstantsetzungen als auch die Negation ändern nichts an der Fehlerwahrscheinlichkeit des randomisierten OBDDs.

3 Randomisierte Ein-Runden-Kommunikationskomplexität von GT

Dieses Kapitel beinhaltet eine ausführliche Darstellung der randomisierten Ein-Runden-Kommunikationskomplexität der *Greater-Than*-Funktion (GT) anhand der Resultate von P. Sen und S. Venkatesh in [SV08]. P. Bro Miltersen, N. Nisan, S. Safra und A. Wigderson haben in [MNSW98] bereits eine lineare untere Schranke für die randomisierte Ein-Runden-Kommunikationskomplexität von GT gezeigt. Sie haben dafür eine Technik verwendet, die es erlaubt, die Anzahl der benutzten Runden auf Kosten der Fehlerwahrscheinlichkeit und der Eingabegröße zu verkleinern. Diese Technik haben sie im sogenannten *Round-Elimination-Lemma* (REL) festgehalten. Das Ergebnis von Sen und Venkatesh ist eine stärkere Version des REL. Sie haben nicht nur die erreichten Schranken verbessert, auch ihr Beweis des REL ist eleganter und intuitiver.

> The original REL is sort of obsolete now - the bounds obtained by Sen and Venkatesh are better and the proof is more elegant. [1]

Während Miltersen et al. das REL mit kombinatorischen Mitteln bewiesen haben, benutzen Sen und Venkatesh Methoden aus der Informationstheorie.

Im ersten Abschnitt in diesem Kapitel fassen wir bekannte Resultate für die Kommunikationskomplexität von GT kurz zusammen. Im zweiten Abschnitt werden wir alle nötigen Grundlagen für den Beweis des REL herausarbeiten. Insbesondere zeigen wir ähnlich zum *Minimax-Prinzip von Yao* für randomisierte Algorithmen, wie die Komplexität von randomisierten Protokollen und deterministischen Protokollen mit einer Eingabeverteilung

[1] Auszug einer E-Mail von P. Bro Miltersen

zusammenhängen. Außerdem geben wir eine ausführliche Einführung in die informationstheoretischen Grundlagen, die wir für das REL benötigen. Im vorletzten Abschnitt beweisen wir das REL von Sen und Venkatesh und wenden es im letzten Abschnitt auf die GT Funktion an, um die lineare untere Schranke für die randomisierte Ein-Runden-Komplexität zu zeigen.

3.1 Allgemeine Kommunikationskomplexität von GT

Wie der Name der *greater than* Funktion schon vermuten lässt, bekommt die Funktion zwei Bitstrings als Eingabe und entscheidet, ob die Binärzahl des ersten Bitstrings echt größer als die Binärzahl des zweiten Bitstrings ist. Formal machen wir folgende Definition.

Definition 3.1.1. *Sei $n \in \mathbb{N}$. Die Funktion $GT_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ ist definiert durch $GT_n(a,b) = 1 :\Leftrightarrow |a| > |b|$, wobei $|x| := \sum_{i=0}^{n-1} 2^i \cdot x_i$ die durch $x \in \{0,1\}^n$ dargestellte natürliche Zahl ist.*

Mit $\overline{GT}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ bezeichnen wir die Negation von GT_n, d.h. $\overline{GT}_n(a,b) = 1 \Leftrightarrow GT_n(a,b) = 0$. Das bedeutet also $\overline{GT}_n(a,b) = 1$ genau dann, wenn $|a|$ kleiner oder gleich $|b|$ ist.

Das Kommunikationsproblem für GT_n ist dann folgendes:

- Alice erhält alle Bits des ersten Eingabebitstrings $a \in \{0,1\}^n$.
- Bob erhält alle Bits des zweiten Eingabebitstrings $b \in \{0,1\}^n$.
- Ziel: Berechnung von $GT_n(a,b)$.

Jedes Kommunikationsproblem lässt sich lösen, indem Alice ihre Eingabe einfach an Bob sendet und Bob dann den Funktionswert berechnet und an Alice verschickt. Im Falle von deterministischen Kommunikationsprotokollen kann man sehr leicht zeigen, dass dieses triviale Protokoll auch optimal für GT_n ist. D.h., die deterministische Kommunikationskomplexität von GT_n ist genau $n+1$ (siehe z.B. [KN97]).

Kann uns Randomisierung bei der Berechnung von GT_n helfen? Falls wir nur einseitigen Fehler für das Kommunikationsprotokoll erlauben, lautet die Antwort „Nein“. Dies liegt daran, dass selbst die Kommunikationskomplexität für nichtdeterministische Protokolle

(welche man sich z.B. als randomisierte Protokolle mit unbeschränktem einseitigen Fehler vorstellen kann) linear ist (siehe [KN97]). Falls aber zweiseitiger Fehler erlaubt ist, können wir ein Protokoll angeben, das GT_n mit $O(\log^2 n)$ Bits berechnet. Dieses Protokoll nutzt es aus, dass das Testen zweier Bitstrings auf Gleichheit im randomisierten Modell einfach ist. Mit einer Hash-Funktion erzeugen Alice und Bob einen Fingerabdruck (*Fingerprinting*-Methode) ihrer Eingabe, der nur logarithmisch viele Bits benötigt. Wenn die Hash-Funktion geeignet gewählt ist, sind diese Fingerabdrücke gleich, falls die Eingaben gleich sind, und mit hoher Wahrscheinlichkeit ($\geq 1 - 1/n$) nicht gleich, falls die Eingabebitstrings verschieden sind. Insgesamt braucht man $O(\log n)$ Bits für die Auswahl der Hash-Funktion und $O(\log n)$ Bits für den Fingerabdruck (siehe [KN97]). Die Idee für das Protokoll für GT ist eine binäre Suche nach dem höchstwertigen Bit der Eingabebitstrings, das verschieden ist. D.h., wir testen zuerst, ob die beiden Bitstrings bestehend aus jeweils den $n/2$ höchstwertigen Bits gleich sind. Falls der Test negativ ausfällt, unterteilen wir diese $n/2$ höchstwertigen Bits in zwei Hälften und fahren rekursiv fort. Sonst unterteilen wir die $n/2$ niedrigstwertigen Bits. Falls einmal alle Tests positiv ausgehen, verwerfen wir die Eingabe. Falls die binäre Suche bei einer einzelnen Bitposition angelangt ist, werden die Bits dieser Position verglichen und entsprechend verworfen oder akzeptiert. Falls die Eingabebitstrings gleich sind, wird auf jeden Fall verworfen, da die Gleichheitstests mit Wahrscheinlichkeit 1 positiv ausgehen. Falls die Bitstrings verschieden sind, kann nur ein Fehler passieren, falls ein Test positiv ausgeht, obwohl die Teilstrings verschieden waren. D.h., die Fehlerwahrscheinlichkeit ist nach oben beschränkt durch die Wahrscheinlichkeit, dass mindestens ein Test einen Fehler macht. Nach der Vereinigungsschranke ist diese Wahrscheinlichkeit höchstens $\frac{\log n}{n}$ und damit klein genug. Es ist sogar möglich, ein Protokoll anzugeben, das eine konstante Fehlerwahrscheinlichkeit hat dafür aber nur $O(\log n)$ Bits benötigt (siehe [KN97]).

Das Protokoll zur Berechnung von GT_n braucht $\log n$ Runden. Zum Beweis von exponentiellen unteren Schranken der randomisierten OBDD Größe einer Funktion f brauchen wir nach Lemma 2.3.4 lineare *Ein-Runden*-Kommunikationskomplexität für f. Daher hoffen wir, dass die $\log n$ Runden wirklich nötig sind, um die Anzahl an ausgetauschten Bits zu senken. Wie wir erfreulicherweise am Ende des Kapitels sehen werden, kann es kein randomisiertes Ein-Runden-Protokoll mit sublinearer Kommunikationskomplexität für GT_n

geben.

3.2 Benötigte Grundlagen

Für den Beweis des REL benötigen wir einige Grundlagen, die wir in diesem Abschnitt herausarbeiten werden. Zuerst betrachten wir eine Methode, um unter anderem untere Schranken für die Komplexität von randomisierten Protokollen zu beweisen. Zu diesem Zweck betrachten wir deterministische Protokolle mit einer Wahrscheinlichkeitsverteilung auf den Eingaben und deren Fehlerwahrscheinlichkeit bzgl. der Eingabeverteilung bestimmt wird. (siehe [KN97])

Definition 3.2.1. *Sei $f : X \times Y \to \{0,1\}$, μ eine Wahrscheinlichkeitsverteilung auf $X \times Y$ und $\varepsilon > 0$. Die* (μ,ε)-verteilungsbezogene Kommunikationskomplexität $D_{\mu,\varepsilon}(f)$ ((μ,ε)-distributional communication complexity) *ist die minimale Komplexität eines deterministischen Kommunikationsprotokolls für f, dessen Fehlerwahrscheinlichkeit bzgl. μ höchstens ε ist. Das bedeutet, das Protokoll berechnet mindestens einen $1-\varepsilon$ Anteil der Eingaben gewichtet mit μ korrekt.*

Als einfaches Beispiel betrachten wir $D_{uniform,1/4}(GT_n)$, wobei $uniform$ die Gleichverteilung auf $\{0,1\}^n \times \{0,1\}^n$ ist. Wir betrachten das folgende sehr einfache Kommunikationsprotokoll für die Eingabe $(a,b) \in \{0,1\}^n \times \{0,1\}^n$: Alice sendet das höchstwertige Bit a_{n-1} ihrer Eingabe an Bob. Bob akzeptiert genau dann, wenn $a_{n-1} > b_{n-1}$ ist. Wenn $a_{n-1} \neq b_{n-1}$ ist, dann berechnet das Protokoll den Wert von $GT_n(a,b)$ korrekt. Falls $a_{n-1} = b_{n-1}$ ist, gibt das Protokoll für mehr als die Hälfte der Eingaben dieser Art korrekt 0 aus. Das Protokoll berechnet also für mehr als 3/4 der Eingaben die Ausgabe korrekt. Da wir die Gleichverteilung gewählt haben, ist die Fehlerwahrscheinlichkeit somit kleiner 1/4. Damit gilt $D_{uniform,1/4}(GT_n) \leq 2$.

Im nächsten Unterabschnitt werden wir sehen, wie uns diese Protokolle helfen können, untere Schranken für die randomisierte Kommunikationskomplexität zu beweisen.

3.2.1 Minimax-Prinzip für randomisierte Kommunikationsprotokolle

Wenn wir deterministische Protokolle für bestimmte Eingaben (oder auch für eine Verteilung auf den Eingaben) betrachten, ist es oft leicht untere Schranken für die Kommunika-

tionskomplexität zu beweisen. Im Gegensatz dazu ist der Beweis von unteren Schranken für randomisierte Protokolle sehr schwierig. In [Yao77] hat Yao eine sehr wichtige Methode zum Beweis von unteren Schranken für die Laufzeit von randomisierten Algorithmen vorgestellt, die den Entwurf randomisierter Algorithmen als spieltheoretisches Problem modelliert. Diese Idee werden wir in diesem Abschnitt dazu verwenden, um randomisierte Kommunikationskomplexität mit Hilfe von verteilungsbezogener Kommunikationskomplexität zu charakterisieren.

Betrachten wir dazu für eine Funktion $f : X \times Y \to \{0, 1\}$ und ein $c \in \mathbb{N}$ folgendes Spiel zwischen den zwei Spielern Eva und Thomas:

- Eva wählt ein deterministisches Kommunikationsprotokoll P für f, das genau c Bits austauscht.
- Thomas wählt eine Eingabe $(x, y) \in X \times Y$.
- Auszahlung: Eva bekommt 1 von Thomas, wenn die Ausgabe von P gleich dem Funktionswert $f(x, y)$ ist. Sonst geht Eva leer aus.

Ziel von Eva ist es, die Auszahlung zu maximieren, während Thomas probiert, die Auszahlung zu minimieren. In der Spieltheorie ist hier von einem *Zweipersonen-Nullsummen-Spiel* die Rede, da der Gewinn von Eva gleich dem Verlust von Thomas ist. Auf den ersten Blick ist nicht klar, wo die randomisierten Protokolle ins Spiel kommen. Die Spieler haben die Möglichkeit ihre Auswahl entweder deterministisch (sog. *reine Strategien*) oder randomisiert (*gemischte Strategien*) vorzunehmen. Im Falle einer gemischten Strategie von Eva liegt eine Verteilung auf deterministischen Protokollen vor. Wenn wir ein randomisiertes Protokoll so verändern, dass die Zufallsbits am Anfang gezogen werden, und dann anhand dieser festen Zufallsbits deterministisch weiter gearbeitet wird, sehen wir, dass eine gemischte Strategie von Eva äquivalent zu einem randomisierten Protokoll ist. Da der Zufallsbitstring am Anfang des Protokolls festgelegt wird, hat jeder Spieler Kenntnis dieses Zufallsbitstrings und somit liegen hier randomisierte Protokolle mit öffentlichem Zufall vor. Es existieren nur endlich viele Eingaben aus $X \times Y$ und da wir uns auf eine feste Anzahl an ausgetauschten Bits festgelegt haben, existieren auch nur endlich viele deterministische Protokolle. Daher existieren auch nur endlich viele Strategien für Eva und Thomas.

Das Minimax-Theorem von Von Neumann (siehe [Neu28]) besagt folgendes.

Theorem 3.2.2 (Minimax-Theorem von Von Neumann)**.** *Für jedes Zweipersonen-Nullsummen-Spiel, bei dem gemischte Strategien erlaubt sind und nur endlich viele Strategien existieren, existiert ein Wert K und für Eva und Thomas jeweils eine gemischte Strategie, so dass folgendes gilt:*

- *Befolgt Eva ihre Strategie, so ist K die beste (= minimale) Auszahlung, die Thomas erreichen kann.*
- *Befolgt Thomas seine Strategie, so ist K die beste (= maximale) Auszahlung, die Eva erreichen kann.*

Mit Hilfe dieser Modellierung bekommen wir eine Charakterisierung der randomisierten Kommunikationskomplexität mit öffentlichem Zufall (siehe [KN97]).

Theorem 3.2.3. *Sei $f : X \times Y \to \{0,1\}$ eine Funktion. Es gilt*

$$R_\varepsilon^{pub}(f) = \max\{D_{\mu,\varepsilon}(f) \mid \mu \text{ Verteilung auf } X \times Y\}.$$

Beweis. Angenommen, es existiert ein randomisiertes Kommunikationsprotokoll P mit öffentlichen Zufallsbits r und Fehler ε, das c Bits zur Kommunikation benötigt. D.h., für alle Eingaben (x,y) ist die Fehlerwahrscheinlichkeit höchstens ε, wobei die Wahrscheinlichkeit über die Wahl der Zufallsbits berechnet wird. Das bedeutet, dass für jede Eingabeverteilung μ die Fehlerwahrscheinlichkeit auch höchstens ε ist, wobei hier die Wahrscheinlichkeit sowohl über die Zufallsbits als auch über die Eingabeverteilung berechnet wird. Denn für jede Eingabeverteilung μ gilt

$$\begin{aligned} &\Pr_{(x,y),r}[\text{Fehler mit Zufallbitstring } r \text{ und Eingabe } (x,y)] &&= \\ &\sum_{(x,y)} \mu(x,y) \cdot \Pr_r[\text{Fehler mit Zufallbitstring } r \text{ und Eingabe } (x,y)] &&\leq \sum_{(x,y)} \mu(x,y) \cdot \varepsilon = \varepsilon. \end{aligned}$$

Dies bedeutet aber auch, dass für einen festen Zufallsbitstring r_μ^*, der von der Eingabeverteilung abhängt, die Fehlerwahrscheinlichkeit höchstens ε ist, wobei hier die Wahrscheinlichkeit nur über die Eingabeverteilung berechnet wird. Angenommen für alle Zufallsbitstrings ist die Fehlerwahrscheinlichkeit größer als ε, dann gilt

$$\begin{aligned} &\Pr_{(x,y),r}[\text{Fehler mit Zufallsbitstring } r \text{ und Eingabe } (x,y)] \\ = \;& \sum_r \Pr[\text{Zufallsbitstring} = r] \cdot \Pr_{(x,y)}[\text{Fehler mit Zufallsbitstring } r \text{ und Eingabe } (x,y)] \\ > \;& \sum_r \Pr[\text{Zufallsbitstring} = r] \cdot \varepsilon = \varepsilon. \end{aligned}$$

Dies wäre im Widerspruch zur Fehlerschranke von höchstens ε. Damit haben wir für jede Eingabeverteilung μ ein deterministisches Protokoll mit (μ, ε)-verteilungsbezogener Komplexität von höchstens c, indem wir den Zufallsbitstring auf r^*_μ in P fixieren. Damit gilt $\max\{D_{\mu,\varepsilon}(f) \mid \mu \text{ Verteilung auf } X \times Y\} \leq R^{pub}_\varepsilon(f)$.

Nun kommen wir zum schwierigeren Beweisteil. Sei dazu

$$c = \max\{D_{\mu,\varepsilon}(f) \mid \mu \text{ Verteilung auf } X \times Y\},$$

die Anzahl an benötigten Bits für ein deterministisches Protokoll mit worst-case Verteilung. Wir wollen jetzt die Existenz eines randomisierten Protokolls mit Fehler höchstens ε nachweisen, das c Bits zur Kommunikation benötigt, um so $R^{pub}_\varepsilon(f) \leq \max\{D_{\mu,\varepsilon}(f) \mid \mu \text{ Verteilung auf } X \times Y\}$ zu zeigen. Hier kommt jetzt die spieltheoretische Modellierung ins Spiel. Eva wählt ein deterministisches Protokoll, das genau c Bits für die Kommunikation benötigt. Thomas wählt eine Eingabe (x, y). Eva erhält 1 von Thomas, wenn ihr Protokoll keinen Fehler auf (x, y) macht, sonst erhält sie nichts. Da c die verteilungsbezogene Komplexität für eine worst-case Verteilung ist, kann Eva für jede gemischte Strategie von Thomas (dies entspricht einer Verteilung auf den Eingaben) eine erwartete Auszahlung von mindestens $1 - \varepsilon$ erreichen. Damit ist die Auszahlung in einer Lösung des Spiels auch mindestens $1-\varepsilon$. Da es sich hier um ein endliches Zweipersonen-Nullsummen-Spiel handelt, existiert daher nach Theorem 3.2.2 eine gemischte Strategie von Eva (dies entspricht dem randomisierten Protokoll mit öffentlichem Zufall), so dass Eva für *jede* Strategie von Thomas eine Auszahlung von mindestens $1 - \varepsilon$ bekommt. Damit haben wir ein randomisiertes Protokoll, dass für jede Eingabe ein Fehler von höchstens ε hat. Daher gilt $R^{pub}_\varepsilon(f) \leq c$. □

Wir werden dieses Theorem im REL in zwei Richtungen anwenden: Im ersten Fall haben wir ein randomisiertes Protokoll mit Fehler ε und Kommunikationskomplexität l. Aus Theorem 3.2.3 folgt dann, dass für jede Eingabeverteilung ein deterministisches Protokoll mit Komplexität höchstens l und Fehlerwahrscheinlichkeit ε existiert. Die zweite Anwendung des Theorems benutzt die $\leq$-Beziehung der Aussage. Wir zeigen, dass für alle Eingabeverteilungen μ die Komplexität $D_{\mu,\varepsilon}$ höchstens l ist. Nach Theorem 3.2.3 folgt dann die Existenz eines randomisierten Protokolls mit Kommunikationskomplexität höchstens l und Fehler ε. Da dies kein konstruktiver Beweis ist, wissen wir nicht genau wie die Protokolle aussehen. Dies ist für unsere Beweise jedoch nicht weiter von Bedeutung.

3.2.2 Wahrscheinlichkeitstheorie

In den Beweisen zum REL kommen keine komplizierten Rechnungen mit Wahrscheinlichkeiten vor. Die Hauptaufgaben erledigen die informationstheoretischen Aussagen aus dem nächsten Abschnitt. Trotzdem benötigen wir eine bekannte Ungleichung aus der Wahrscheinlichkeitstheorie, die wir in diesem Abschnitt erarbeiten.

Zuerst brauchen wir folgende aus der Analysis bekannte Definition.

Definition 3.2.4. *Sei I ein reelles Intervall und $f : I \to \mathbb{R}$ eine Funktion. Die Funktion f heißt* konvex*, wenn für alle $x, y \in I$ mit $x < y$ und $t \in (0,1)$ gilt*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).$$

f heißt konkav*, wenn die Funktion $-f$ konvex ist, d.h., es gilt*

$$f(tx + (1-t)y) \geq tf(x) + (1-t)f(y).$$

Anschaulich bedeutet die Definition einer konvexen Funktion, dass alle Funktionswerte der Funktion zwischen zwei beliebigen Stellen x und y unterhalb oder auf der Geraden liegen, die die beiden Funktionswerte $f(x)$ und $f(y)$ miteinander verbindet. Analog bedeutet konkav, dass alle Funktionswerte oberhalb oder auf der Verbindungsgeraden liegen.

Die sogenannte *Jensen* Ungleichung besagt, dass die Ungleichung in der Definition von konvexen Funktion auch für mehr als nur zwei Stützstellen gilt. Seien dazu $x_1, \ldots, x_n$ verschiedene Stellen aus dem Intervall I. Eine Konvexkombination von $x_1, \ldots, x_n$ ist eine gewichtete Summe $\sum_{i=1}^{n} \lambda_i x_i$, wobei $\sum_{i=1}^{n} \lambda_i = 1$ und $\lambda_i \geq 0$ gilt. Die Jensen-Ungleichung besagt dann:

$$f(\sum_{i=1}^{n} \lambda_i x_i) \leq \sum_{i=1}^{n} \lambda_i f(x_i).$$

Wir brauchen die Ungleichung im Zusammenhang mit dem Erwartungswert einer Zufallsvariable. Da der Erwartungswert eine Konvexkombination von den Werten der Zufallsvariable ist, ist die obige Formulierung äquivalent zu $f(E(X)) \leq E(f(X))$, die wir nun beweisen werden (für den Beweis siehe auch [CT06]).

Theorem 3.2.5. *Sei f eine konvexe Funktion und X eine Zufallsvariable. Dann gilt*

$$f(E(X)) \leq E(f(X)).$$

Beweis. Wir beweisen die Aussage per vollständiger Induktion über die Anzahl an Stützstellen, d. h. in unserem Fall über die Anzahl an Werten $x_1, \ldots, x_n$, die die Zufallsvariable annehmen kann. Sei $p(x)$ die Wahrscheinlichkeit für das Ereignis $X = x$. Für $n = 2$ folgt die Behauptung aus der Definition von konvex. Sei die Aussage wahr für $n = k$. Seien $x_1, \ldots, x_{k+1}$ die möglichen Werte von X. Wir setzen

$$p'(x_i) = p(x_i)/(1 - p(x_{k+1})) \qquad \text{für } i = 1, \ldots, k.$$

Dann gilt

$$\begin{aligned}
f(E(X)) &= f\left(\sum_{i=1}^{k+1} p(x_i)x_i\right) \\
&= f\left(p(x_{k+1})x_{k+1} + (1 - p(x_{k+1}))\sum_{i=1}^{k} p'(x_i)x_i\right) \\
&\overset{f \text{ konvex}}{\leq} p(x_{k+1})f(x_{k+1}) + (1 - p(x_{k+1}))f\left(\sum_{i=1}^{k} p'(x_i)x_i\right) \\
&\overset{\text{I.V.}}{\leq} p(x_{k+1})f(x_{k+1}) + (1 - p(x_{k+1}))\sum_{i=1}^{k} p'(x_i)f(x_i) \\
&= \sum_{i=1}^{k+1} p(x_i)f(x_i)
\end{aligned}$$

□

Für konkave Funktionen lässt sich leicht aus der Jensen Ungleichung folgendes Korollar herleiten.

Korollar 3.2.6. *Sei f eine konkave Funktion und X eine Zufallsvariable. Dann gilt*

$$f(E(X)) \geq E(f(X)).$$

Beweis. Nach der Definition von konkav ist die Funktion $-f$ konvex. Nach der Jensen Ungleichung gilt also

$$-f(E(X)) \leq E(-f(X)) = -E(f(X)),$$

denn es gilt $E(cX) = cE(X)$ für alle konstanten $c \in \mathbb{R}$. Somit folgt die Behauptung. □

3.2.3 Informationstheorie

In diesem Abschnitt werde wir einige Begriffe und Aussagen aus der Informationstheorie behandeln. Einen guten Überblick über dieses Themengebiet liefert das Buch [CT06] von Cover und Thomas, an dem wir uns in diesem Kapitel größtenteils orientieren werden, d.h., falls es nicht anders angegeben ist, stammen die Definitionen und Sätze in diesem Abschnitt aus diesem Buch.

In der Informationstheorie ist man an dem Informationsgehalt einer Zufallsvariable interessiert. Die ersten Definitionen behandeln daher unterschiedliche Maße, um den Informationsgehalt einer Zufallsvariable, den Informationsgehalt einer Zufallsvariable über eine andere Zufallsvariable und den Abstand zweier Zufallsvariablen zu messen.

Definition 3.2.7. *Sei $X : \Omega \to \mathbb{R}$ eine diskrete Zufallsvariable (d. h., die Ergebnismenge Ω ist endlich bzw. abzählbar unendlich) mit Wertebereich $\mathcal{X} = \{X(\omega) \mid \omega \in \Omega\}$ und Dichtefunktion $p_X(x) = Pr[X = x]$ für $x \in \mathcal{X}$. Die* Entropie $H(X)$ *von X ist definiert durch*

$$H(X) := -\sum_{x \in \mathcal{X}} p_X(x) \log p_X(x).$$

Wir setzen dabei $p_X(x) \log p_X(x) = 0$ für $p_X(x) = 0$, da die Funktion $x \log x$ gegen Null konvergiert für $x \to 0$.

Die Entropie einer Zufallsvariable misst die Unsicherheit über den Wert einer Zufallsvariable bzw. gibt an, wie viel Information wir gewinnen, wenn wir den Wert der Zufallsvariable erfahren. Da wir den Logarithmus zur Basis 2 benutzen, wird diese Information/Unsicherheit in Bits gemessen. Die Entropie $H(X)$ gibt damit die durchschnittliche Anzahl an benötigten Bits an, die wir benötigen, um den Wert der Zufallsvariable zu kodieren. Betrachten wir zwei Beispiele für die Entropie. Seien X, Y zwei Zufallsvariablen, die $n \in \mathbb{N}$ verschiedene Werte annehmen können. Die Dichtefunktion von X sei hier die Gleichverteilung, d.h. $p_X(x) = 1/n$ für alle $x \in \mathcal{X}$, und die Zufallsvariable Y nimmt mit Wahrscheinlichkeit 1 genau einen Wert an. Dann gilt

$$H(X) = -\sum_{x \in \mathcal{X}} \frac{1}{n} \log \frac{1}{n} = n \frac{1}{n} \log n = \log n$$

und

$$H(Y) = -\sum_{y \in \mathcal{Y}} p_Y(y) \log p_Y(y) = 1 \log 1 = 0.$$

Dies entspricht der oben beschriebenen Charakterisierung der Entropie als Anzahl unsicherer Bits. Im Falle der Gleichverteilung sind alle Bits nicht vorhersagbar, während im Extremfall von Y alle Bits vorher feststehen.

Definition 3.2.8. *Seien X, Y zwei diskrete Zufallsvariablen mit gemeinsamer Verteilung $p_{(X,Y)}(x,y) = Pr[X = x, Y = y]$. Dann definieren wir*

1. *die* gemeinsame Entropie *von X und Y als*

$$H(X,Y) := -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{(X,Y)}(x,y)\log p_{(X,Y)}(x,y).$$

2. *die* bedingte Entropie *von Y gegeben X als*

$$H(Y \mid X) := \sum_{x\in\mathcal{X}} p_X(x)H(Y \mid X = x),$$

wobei $H(Y \mid X = x)$ die Entropie der Zufallsvariable Y unter der Bedingung $X = x$ bezeichnet. D.h. für $p(y \mid x) := Pr[Y \mid X = x]$ ist

$$H(Y \mid X = x) = -\sum_{y\in\mathcal{Y}} p(y \mid x)\log p(y \mid x).$$

Wenn klar ist, zu welcher Zufallsvariable eine Dichtefunktion gehört, lassen wir im weiteren Verlauf den Index der Dichtefunktionen weg und deuten durch den Variablennamen an, welche Zufallsvariable gemeint ist (z.B. $p(y) = p_Y(y)$).

Wie man es erwartet und der nächste Satz auch zeigen wird, ist die gemeinsame Entropie zweier Zufallsvariablen gleich der Summe der Entropie einer Zufallsvariablen und der bedingten Entropie von der anderen Zufallsvariable.

Theorem 3.2.9 (Kettenregel für die Entropie)**.**

$$H(X,Y) = H(X) + H(Y \mid X)$$

Beweis. Nach dem Satz der totalen Wahrscheinlichkeit und der Definition der bedingten Wahrscheinlichkeit $p(x)p(y \mid x) = p(y)p(x \mid y) = p(x,y)$ gilt

$$p(x) = \sum_y p(y)p(x \mid y) = \sum_y p(x,y).$$

Durch Einsetzen der Definition der Entropie und kleine Umformungen erhalten wir damit

$$\begin{aligned} H(X,Y) &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x)p(y\mid x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x) - \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(y\mid x) \\ &= -\sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x)p(y\mid x)\log p(y\mid x) \\ &= H(X) + H(Y\mid X). \end{aligned}$$

□

Die Gleichung gilt auch dann, wenn auf beiden Seiten der Kettenregel die gleiche Bedingung hinzugefügt wird

$$H(X,Y\mid Z) = H(X\mid Z) + H(Y\mid X,Z).$$

Der Beweis verläuft analog zu dem unbedingten Fall. Damit kann durch induktives Anwenden die Kettenregel auch auf eine größere Menge von Zufallsvariablen $X_1,\ldots,X_n$ erweitert werden:

$$\begin{aligned} H(X_1,X_2) &= H(X_1) + H(X_1\mid X_2) \\ H(X_1,X_2,X_3) &= H(X_1) + H(X_2,X_3\mid X_1) \\ &= H(X_1) + H(X_2\mid X_1) + H(X_3\mid X_2,X_1) \\ &\vdots \\ H(X_1,\ldots,X_n) &= \sum_{i=1}^{n} H(X_i\mid X_{i-1},\ldots,X_1) \end{aligned}$$

Als nächstes führen wir eine Art Abstandsmaß zwischen zwei Wahrscheinlichkeitsverteilungen ein.

Definition 3.2.10. *Die* relative Entropie *oder* Kullback Leibler Divergenz *zweier Dichtefunktionen $p(x)$ und $q(x)$ ist definiert durch*

$$D(p\parallel q) \coloneqq \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}.$$

Wie schon bei der Definition zur Entropie setzen wir begründet durch die Grenzwerte $0\log\frac{0}{q} = 0$ und $p\log\frac{p}{0} = \infty$ für $p \neq 0$.

Die relative Entropie ist keine Metrik, da sie weder symmetrisch ist noch die Dreiecksungleichung erfüllt. Betrachte z.B. $p(0) = p(1) = 1/2$ und $q(0) = 1/3$ und $q(1) = 2/3$. Dann gilt

$$D(p \parallel q) = p(0) \log \frac{p(0)}{q(0)} + p(1) \log \frac{p(1)}{q(1)} = 1/2 \cdot \log 3/2 + 1/2 \cdot \log 3/4 \approx 0.085$$

$$D(q \parallel p) = q(0) \log \frac{q(0)}{p(0)} + q(1) \log \frac{q(1)}{p(1)} = 1/3 \cdot \log 2/3 + 2/3 \cdot \log 4/3 \approx 0.082$$

Im weiteren hilft uns aber die Ansicht, dass die relative Entropie in gewisser Weise die „Distanz“ zwischen zwei Verteilungen misst.

Es gilt $D(p \parallel q) \geq 0$ mit Gleichheit genau dann, wenn $p = q$ ist (hier ohne Beweis). Diese Aussage bringt uns zu einer oberen Schranke für die Entropie.

Theorem 3.2.11. *Für eine diskrete Zufallsvariable X mit Wertebereich $\mathcal{X}$ gilt*

$$0 \leq H(X) \leq \log |\mathcal{X}|.$$

Beweis. Die untere Schranke sehen wir leicht, da aus $0 \leq p(x) \leq 1$ folgt $\log p(x) \leq 0$ und damit $H(X) \geq 0$ gilt.

Für die obere Schranke sei $u(x) = \frac{1}{|\mathcal{X}|}$ für alle $x \in \mathcal{X}$ die Gleichverteilung über $\mathcal{X}$ und p die Dichtefunktion von der Zufallsvariablen X. Dann gilt

$$0 \leq D(p \parallel u) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{u(x)} = \log |\mathcal{X}| \sum_{x \in \mathcal{X}} p(x) + \sum_{x \in \mathcal{X}} p(x) \log p(x) = \log |\mathcal{X}| - H(X)$$

und damit die Behauptung. □

Kommen wir nun zur wesentlichen Definition der *wechselseitigen Information*. Wir sind später daran interessiert, wie viel Information eine Zufallsvariable (z.B. eine Nachricht von Alice im Kommunikationsprotokoll) über eine andere Zufallsvariable (z.B. die Eingabe von Alice) enthalten kann.

Definition 3.2.12. *Für zwei diskrete Zufallsvariablen X, Y ist die* wechselseitige Information *$I(X : Y)$ definiert durch*

$$I(X : Y) = H(X) + H(Y) - H(X, Y).$$

Durch Anwenden der Kettenregel für die gemeinsame Entropie (Theorem 3.2.9) ist dies äquivalent zu

$$I(X:Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X).$$

Die bedingte wechselseitige Information *von X und Y gegeben eine weitere Zufallsvariable Z ist definiert durch*

$$I(X:Y \mid Z) := H(X \mid Z) - H(X \mid Y,Z) = H(Y \mid Z) - H(Y \mid X,Z).$$

Das bedeutet, die wechselseitige Information misst die Unsicherheit über X bzw. Y, die übrig bleibt, wenn wir den Ausgang von Y bzw. X kennen. Aus Sicht der relativen Entropie ist die wechselseitige Information gleich dem „Abstand“ der gemeinsamen Verteilung von X und Y und der Produktverteilung der Randverteilungen von X und Y.

Theorem 3.2.13. *Für zwei diskrete Zufallsvariablen X und Y mit Dichtefunktionen $p_X(x)$ und $p_Y(y)$ sei $p(x,y) = Pr[X = x, Y = y]$ die gemeinsame Verteilung. Weiter sei $(p_X \otimes p_Y)(x,y) = Pr[X = x] \cdot Pr[Y = y] = p_X(x)p_Y(y)$ die Produktverteilung der Randverteilungen p_X und p_Y. Dann gilt*

$$I(X:Y) = D(p \parallel p_X \otimes p_Y).$$

Beweis. Durch einfaches Einsetzen der Definitionen erhalten wir

$$\begin{aligned}
D(p \parallel p_X \otimes p_Y) &= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p_X(x)p_Y(y)} \\
&= -\sum_{x,y} p(x,y) \log p_X(x) - \sum_{x,y} p(x,y) \log p_Y(y) + \sum_{x,y} p(x,y) \log p(x,y) \\
&= -\sum_{x} p_X(x) \log p_X(x) - \sum_{y} p_Y(y) \log p_Y(y) + \sum_{x,y} p(x,y) \log p(x,y) \\
&= H(X) + H(Y) - H(X,Y) \\
&= I(X:Y).
\end{aligned}$$

□

Aus Theorem 3.2.13 folgt damit auch, dass $I(X:Y) \geq 0$ und $I(X:Y) = 0$ genau dann, wenn X und Y unabhängig sind. Das sind Eigenschaften, die man intuitiv auch von wechselseitiger Information erwartet hat.

Analog zur Entropie können wir auch eine Kettenregel für die wechselseitige Information angeben.

Theorem 3.2.14 (Kettenregel für die wechselseitige Information)**.**

$$I(X_1,\dots,X_n:Y)=\sum_{i=1}^{n} I(X_i:Y\mid X_{i-1},\dots X_1)$$

Beweis. Durch Anwenden der Definition $I(X:Y)=H(X)-H(X\mid Y)$ bzw. $I(X:Y\mid Z)=H(X\mid Z)-H(X\mid Y,Z)$ und der Kettenregel für die Entropie erhalten wir

$$\begin{aligned} I(X_1,\dots,X_n:Y) &= H(X_1,\dots,X_n)-H(X_1,\dots,X_n\mid Y)\\ &= \sum_{i=1}^{n} H(X_i\mid X_{i-1},\dots,X_1)-\sum_{i=1}^{n} H(X_i\mid X_{i-1},\dots,X_1,Y)\\ &= \sum_{i=1}^{n} I(X_i:Y\mid X_{i-1},\dots,X_1) \end{aligned}$$

□

Für den Beweis des REL benötigen wir noch drei Beobachtungen (ohne Beweis aus [SV08]), die leicht aus den Definitionen und bekannten Resultaten folgen.

Lemma 3.2.15. *Seien X,Y,Z diskrete Zufallsvariablen. Dann gilt*

1. *$I(X:Y)\le\log|\mathcal{X}|$, wobei $\mathcal{X}$ der Wertebereich von X ist.*
2. *$I(X,Y:Z)=I(X:Z)+I(Y:Z,X)-I(X:Y)$. Insbesondere ist $I(X,Y:Z)=I(X:Z)+I(Y:Z,X)$, wenn X und Y unabhängig sind.*
3. *Ein Spezialfall der Kettenregel für die wechselseitige Information ist $I(Y:Z,X)=I(X:Y)+I((Y:Z)\mid X)$.*

Beweis. Es gilt $I(X:Y)=H(X)-H(X\mid Y)$ und damit folgt aus Theorem 3.2.11 die erste Aussage $I(X:Y)\le\log|\mathcal{X}|$.

Für die zweite Aussage benutzen wir die Definition $I(X:Y)=H(X)+H(Y)-H(X,Y)$ und die einfache Beobachtung, dass $H(X,Y)=H(Y,X)$ gilt:

$$\begin{aligned} I(X,Y:Z) &= H(X,Y)+H(Z)-H(X,Y,Z)\\ &= (\mathbf{H(X)}+H(Z)-\mathbf{H(X,Z)})+(\mathbf{H(Y)}+\mathbf{H(Z,X)}-H(Z,Y,X))\\ &\quad -(\mathbf{H(X)}+\mathbf{H(Y)}-H(X,Y))\\ &= I(X:Z)+I(Y:Z,X)-I(X:Y) \end{aligned}$$

Wenn X und Y unabhängig sind, ist die wechselseitige Information von X und Y gleich Null und daher gilt auch $I(X,Y:Z) = I(X:Z) + I(Y:Z,X)$.

Da die wechselseitige Information symmetrisch ist, gilt $I(Y:ZX) = I(ZX:Y)$. Anwenden der Kettenregel aus Theorem 3.2.14 liefert uns die dritte Aussage. □

Wie wir bereits gesehen haben, misst die relative Entropie intuitiv den Abstand zweier Verteilungen, aber sie ist keine Metrik im mathematischen Sinne. Wir definieren als nächstes die sogenannte $\mathcal{L}_1$ Distanz zweier Verteilungen, die eine Metrik auf dem Raum der Wahrscheinlichkeitsverteilungen mit Ergebnismenge Ω definiert.

Definition 3.2.16. *Die* $\mathcal{L}_1$ Distanz (Totalvariationsabstand) *zwischen zwei Wahrscheinlichkeitsverteilungen* $P : \Omega \to [0,1]$ *und* $Q : \Omega \to [0,1]$ *auf der diskreten Ergebnismenge* Ω *ist definiert durch*

$$\| P - Q \|_1 := \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)|.$$

In dieser Definition erlauben wir auch Dichtefunktionen von Zufallsvariablen anstelle von Verteilungen, indem wir den Wertebereich der Zufallsvariable einfach als Ergebnismenge Ω interpretieren. Zuerst zeigen wir eine alternative Definition der $\mathcal{L}_1$ Distanz, die auch oft in der Literatur verwendet wird und die uns eine Abschätzung der Summe $\sum_{\omega \in \Omega} P(\omega) - Q(\omega)$ liefert.

Beobachtung 3.2.17. *Es gilt*

$$\frac{1}{2} \| P - Q \|_1 = \max_{A \subseteq \Omega} \sum_{\omega \in A} (P(\omega) - Q(\omega)).$$

Beweis. Sei $B \subseteq \Omega$ die Menge aller Elemente $\omega \in \Omega$ mit $P(\omega) > Q(\omega)$. Da P eine Wahrscheinlichkeitsverteilung ist, gilt

$$\sum_{\omega \in B} P(\omega) = 1 - \sum_{\omega \in \Omega \setminus B} P(\omega) \qquad \text{(analog auch für } Q\text{)}.$$

Damit haben wir

$$\begin{aligned}
\| P - Q \|_1 &= \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)| \\
&= \sum_{\omega \in B} (P(\omega) - Q(\omega)) + \sum_{\omega \in \Omega \setminus B} (Q(\omega) - P(\omega)) \\
&= \sum_{\omega \in B} (P(\omega) - Q(\omega)) + 1 - \sum_{\omega \in B} Q(\omega) - (1 - \sum_{\omega \in B} P(\omega)) \\
&= 2 \sum_{\omega \in B} (P(\omega) - Q(\omega)).
\end{aligned}$$

Da alle Elemente aus B auch in einer Menge A^* enthalten sein müssen, die $\max_{A\subseteq\Omega} \sum_{\omega\in A} (P(\omega) - Q(\omega)) = \sum_{\omega\in A^*} (P(\omega) - Q(\omega))$ erfüllt, und für alle Elemente $\omega \in A^*\backslash B$ die Wahrscheinlichkeiten $P(\omega)$ und $Q(\omega)$ gleich sein müssen, folgt die Behauptung. □

Die folgende (nicht triviale) Beziehung zwischen der $\mathcal{L}_1$ Distanz und der relativen Entropie wollen wir hier ohne Beweis festhalten (siehe z.B. Lemma 12.6.1 in [CT06] für einen Beweis).

Theorem 3.2.18. *Seien P und Q Wahrscheinlichkeitsverteilungen. Dann gilt*

$$D(P \parallel Q) \geq \frac{1}{2\ln 2} \parallel P - Q \parallel_1^2 .$$

Wie wir in Theorem 3.2.13 gesehen haben, ist die wechselseitige Information zweier Zufallsvariablen gleich der relativen Entropie von ihrer gemeinsamen Verteilung und der Produktverteilung. In Kombination mit Theorem 3.2.18 bekommen wir das sogenannte *Average Encoding* Theorem in der klassischen Form. Das Average Encoding Theorem wurde zuerst von Klauck, Nayack, Ta-Shma und Zuckerman in [KNTSZ01] für die Quanteninformationstheorie bewiesen. Aus diesem Theorem folgt auch die klassische Variante, die wir hier betrachten (siehe [SV08]).

Theorem 3.2.19. *Seien X und M Zufallsvariablen und p_X die Dichtefunktion von X. Sei Π^x die bedingte Verteilung von M gegeben $X = x$ und $\Pi = \sum_x p_X(x)\Pi^x$ die Verteilung von M. Dann gilt*

$$\sum_x p_X(x) \parallel \Pi^x - \Pi \parallel_1 \leq \sqrt{(2\ln 2) I(X:M)}.$$

Beweis. Sei $p(x,m)$ die gemeinsame Verteilung von X und M, d.h. $p(x,m) = p_X(x)\Pi^x(m)$, und $(p_X \otimes \Pi)(x,m) = p_X(x)\Pi(m)$ die Produktverteilung von X und M. Nach Theorem 3.2.13 gilt dann $I(X:M) = D(p \parallel p_X \otimes \Pi)$. Nach der Definition der Totalvariationsdistanz gilt

$$\begin{aligned} \parallel p - (p_X \otimes \Pi) \parallel_1 &= \sum_{x,m} |p_X(x)\Pi^x(m) - p_X(x)\Pi(m)| \\ &= \sum_x p_X(x) \sum_m |\Pi^x(m) - \Pi(m)| \\ &= \sum_x p_X(x) \parallel \Pi^x - \Pi \parallel_1 . \end{aligned}$$

Und mit Theorem 3.2.18 folgt dann

$$\sqrt{(2\ln 2)I(X:M)} = \sqrt{(2\ln 2)D(p \parallel p_X \otimes \Pi)} \geq \parallel p - (p_X \otimes \Pi) \parallel_1 = \sum_x p_X(x) \parallel \Pi^x - \Pi \parallel_1 .$$

□

Betrachten wir ein konkretes Beispiel für ein besseres Verständnis des Average Encoding Theorems. Sei X die Zufallsvariable für die Eingabe von Alice (bzgl. einer Eingabeverteilung) und M die Zufallsvariable für die erste Nachricht von Alice. Falls die Information, die die erste Nachricht über Alices Eingabe hergibt, klein ist, d.h. $I(X:M)$ klein ist, dann besagt das Average Encoding Theorem, dass die durchschnittliche Nachricht fast so gut wie die tatsächliche Nachricht ist. Das würde bedeuten, dass Bob sich diese durchschnittliche Nachricht selber erzeugen kann, und wir so eine Runde einsparen können.

3.3 Das Round Elimination Lemma

Nun haben wir alle nötigen Grundlagen zusammen, um das REL zu beweisen. Die in diesem Abschnitt enthaltenen Definitionen und Sätze sind wie anfangs erwähnt aus der Arbeit von P. Sen und S. Venkatesh ([SV08]).

Betrachten wir folgendes Kommunikationsproblem:

- Alice bekommt $x_1, \ldots, x_m$ als Eingabe.
- Bob bekommt y und $i \in \{1, \ldots, m\}$ als Eingabe.
- Ziel: Berechnung von $GT(x_i, y)$.

Alice beginnt mit der ersten Nachricht. Wenn diese Nachricht nur wenige Bits benutzt, kann diese Nachricht wenig nützliche Informationen über Alices Eingabe beinhalten, da Alice i noch nicht kennt. D.h., wir können die erste Nachricht im Prinzip überspringen und erst bei der zweiten anfangen. Dies motiviert die folgende Definition.

Definition 3.3.1. *Sei $m \geq 1$. Im Kommunikationsproblem $GT_n^{(m),A}$ erhält Alice als Eingabe $x_1, \ldots x_m \in \{0,1\}^n$ und Bob $y \in \{0,1\}^n$, $i \in \{1, \ldots, m\}$ sowie Kopien von $x_1, \ldots, x_{i-1}$. Alice beginnt mit der ersten Nachricht. Ziel ist es $GT_n(x_i, y)$ zu berechnen.*

Die Intuition des REL ist folgende: Wir betrachten ein deterministisches Protokoll mit einer Eingabeverteilung, bei der i gleichverteilt gezogen wird (was der worst-case für Alice wäre). Wenn Alice nun l_1 Bits für ihre erste Nachricht verwendet, dann sind im Erwartungswert l_1/m Bits nützlich für Bob. Wenn Bob diese Bits einfach raten würde, würde er mit einer Wahrscheinlichkeit von $1 - 2^{-l_1/m}$ diese Bits nicht korrekt raten. Das bedeutet, die Fehlerwahrscheinlichkeit sollte sich um $1 - 2^{-l_1/m}$ erhöhen, wenn wir die erste Nachricht weglassen und Bob die Nachricht rät. Aus der bekannten Abschätzung für die Exponentialfunktion $e^x \leq \frac{1}{1-x}$ für $x < 1$ folgt

$$e^{1-\frac{1}{2^x}} \leq \frac{1}{1-1+2^{-x}} = 2^x$$

und damit $1 - 2^{-l_1/m} \leq l_1/m$. Durch das REL erreichen wir einen etwas schlechteren additiven Fehler von $O(\sqrt{l_1/m})$, da die Fehlerabschätzung hier etwas vereinfacht dargestellt ist.

Im Folgenden benutzen wir die Notation ε_P^μ, um den Fehler eines Protokolls P über die Eingabeverteilung μ zu beschreiben (oder im Falle eines randomisierten Protokolls noch zusätzlich über die Zufallsbits). Um die Länge der einzelnen Nachrichten und den Spieler mit der ersten Nachricht für ein Protokoll explizit angeben zu können, machen wir folgende Definition.

Definition 3.3.2. *Ein randomisiertes $[k, l_1, \ldots, l_k]^S$ Protokoll ist ein k-Runden Protokoll, bei dem Alice ($S = A$) oder Bob ($S = B$) anfängt, und die i-te Nachricht l_i Bits benötigt für $i = 1, \ldots, k$.*

Für das REL brauchen wir eine Hilfsaussage, so dass wir aus einem randomisierten $[k, l_1, \ldots, l_k]^A$ Protokoll P mit Eingabeverteilung μ und Fehler ε_P^μ die Existenz eines deterministischen Protokolls P' mit Fehler $\varepsilon_{P'}^\mu \leq \varepsilon_P^\mu + \sqrt{(2 \ln 2) I(X : M)}$ nachweisen können, wobei X die Zufallsvariable von Alices Eingabe und M die Zufallsvariable der ersten Nachricht bezeichnet. Der grobe Ablauf des Beweises dieser Hilfsaussage sieht folgendermaßen aus:

1. Wir definieren ein neues randomisiertes Protokoll Q, das die erste Nachricht erzeugt, ohne die Eingabe von Alice zu betrachten.

2. Wir zeigen mit Hilfe des Average Encoding Theorems, dass die Fehlerwahrscheinlichkeit von Q durch $\varepsilon_P^\mu + \sqrt{(2 \ln 2) I(X : M)}$ nach oben beschränkt ist.

3. Da die erste Nachricht in Q unabhängig ist von der Eingabe, kann Bob sich diese Nachricht selbst generieren. D.h., wir haben ein randomisiertes $[k-1, l_2, \ldots, l_k]^B$ Protokoll mit Fehler von ε_Q^μ.

4. Festlegen der Zufallsbits liefert uns das gewünschte deterministische $[k-1, l_2, \ldots, l_k]^B$ Protokoll.

Mit Hilfe dieser Aussage können wir das REL beweisen. Das REL besagt, dass wir aus der Existenz eines randomisierten $[k, l_1, \ldots, l_k]^A$ Protokolls für $GT_n^{(m),A}$ mit öffentlichem Zufall und Fehler kleiner als δ die Existenz eines $[k-1, l_2, \ldots, l_k]^B$ Protokolls für GT_n mit öffentlichem Zufall und Fehler kleiner als $\delta + \sqrt{(2l_1 \ln 2)/m}$ nachweisen können. Auch hier geben wir kurz den groben Ablauf des Beweises an:

1. Angenommen es existiert ein randomisiertes $[k, l_1, \ldots, l_k]^A$ Protokoll P für $GT_n^{(m),A}$ mit öffentlichem Zufall und Fehler kleiner als δ.

2. Idee zur Existenz des Protokolls für GT_n:
   Wir zeigen, dass für jede Eingabeverteilung μ auf $\{0,1\}^n \times \{0,1\}^n$ ein deterministisches $[k-1, l_2, \ldots, l_k]^B$ Protokoll für GT_n mit Fehler kleiner als $\delta + \sqrt{(2l_1 \ln 2)/n}$ existiert. Aus dem Minimax-Prinzip (Theorem 3.2.3) folgt die Existenz eines randomisierten $[k-1, l_2, \ldots, l_k]^B$ Protokolls für GT_n mit gleicher Fehlerwahrscheinlichkeit.

3. Idee zur Konstruktion eines deterministischen Protokolls für GT_n mit einer Eingabeverteilung μ:

   a) Aus dem Minimax-Prinzip folgt, dass für jede Eingabeverteilung auf Eingaben für $GT_n^{(m),A}$ ein deterministisches Protokoll P' existiert, dass die gleiche Anzahl an Bits wie P versendet und die gleiche Fehlerwahrscheinlichkeit (bzgl. der Eingabeverteilung) hat.

   b) Wir wählen die Verteilung μ' so, dass Paare (x_j, y_j) zufällig nach μ gezogen werden und i gleichverteilt zufällig gewählt wird. Dann erhalten Alice $x_1, \ldots, x_m$ und Bob i und y_i als Eingaben.

   c) Die Verteilung ist so gewählt, dass die erwartete relevante Information (in P'), die Bob von der ersten Nachricht erhält, durch l_1/m nach oben beschränkt ist.

d) Wir konstruieren für alle $i, x_1, \ldots, x_{i-1}$ ein randomisiertes Protokoll $P_{i;x_1,\ldots,x_{i-1}}$ für GT_n aus Q, so dass der erwartete Fehler (bzgl. μ) dieser Protokolle gleich dem Fehler von P' ist (bzgl. μ' und der Bedingung, dass $i, x_1, \ldots, x_{i-1}$ gegeben sind).

e) Mit der Hilfsaussage erhalten wir aus diesen randomisierten Protokollen deterministische Protokolle und können zeigen, dass der durchschnittliche Fehler der deterministischen Protokolle (bzgl. μ') kleiner als $\delta + \sqrt{(2l_1 \ln 2)/m}$ ist. Damit existiert ein deterministisches Protokoll mit Fehler (bzgl. μ) kleiner als $\delta + \sqrt{(2l_1 \ln 2)/m}$.

Das Hilfslemma ist also für die Reduzierung der Runden verantwortlich, während das REL das Minimax-Prinzip vollständig ausnutzt, um mit Hilfe dieses Tricks der Runden-Elimination von einem Protokoll für $GT_n^{(m),A}$ auf ein Protokoll für GT_n zu kommen.

Beginnen wir mit dem detaillierten Beweis für das Hilfslemma.

Lemma 3.3.3 (Verteilungsbezogene Runden-Elimination)**.** *Sei μ eine Wahrscheinlichkeitsverteilung auf $\{0,1\}^n \times \{0,1\}^n$ und P ein randomisiertes $[k, l_1, \ldots, l_k]$ private-coin Protokoll für GT_n. Sei X die Zufallsvariable, die die Eingabe $x \in \{0,1\}^n$ von Alice bezeichnet, und M die Zufallsvariable der ersten Nachricht von Alice in P. Dann existiert ein deterministisches $[k-1, l_2, \ldots, l_k]^B$ Protokoll P' für GT_n mit Fehlerwahrscheinlichkeit $\varepsilon_{P'}^{\mu} \leq \varepsilon_P + (1/2)\sqrt{(2 \ln 2) I(X:M)}$, wobei $I(X:M)$ die wechselseitige Information von X und M bezeichnet, wenn die Eingabe für P bzgl. μ gezogen wird.*

Beweis. Zuerst definieren wir uns einige Verteilungen, die wir benötigen. Mit $\mu(x)$ bezeichnen wir die Wahrscheinlichkeit, dass $X = x$ ist, d.h. $\mu(x) = \sum_y \mu(x,y)$. Mit Π^x bezeichnen wir die bedingte Verteilung der ersten Nachricht von Alice, wenn Alices Eingabe x ist, und mit Π die Verteilung der durchschnittlichen ersten Nachricht von Alice, d.h. $\Pi = \sum_x \mu(x)\Pi^x$. Bzgl. der privaten Zufallsbits von Alice (= alle Zufallsbits von Alice; nicht nur die, die für die erste Nachricht relevant sind) definieren wir

$$p_r^{xm} := \Pr[\text{Zufallsbits} = r \text{ in P} \mid X = x, M = m]$$

als die bedingte Wahrscheinlichkeit, dass Alices Zufallsbits gleich r sind, wenn Alices Eingabe x und ihre erste Nachricht m ist. Wenn diese Konstellation von Eingabe x und erste Nachricht m nicht möglich ist, setzen wir $p_r^{xm} = 0$ für alle r.

Nun konstruieren wir ein Protokoll Q für GT_n mit Eingabe (x, y), bei dem die erste Nachricht von Alice unabhängig von ihrer Eingabe x ist:

1. Alice wählt ihre erste Nachricht m mit Wahrscheinlichkeit $\Pi(m)$ und setzt ihre privaten Zufallsbits auf r mit Wahrscheinlichkeit p_r^{xm}.
2. Alice und Bob verfahren wie in Protokoll P fort.

Falls Alice im ersten Schritt eine Nachricht gewählt hat, zu der sie keine passenden Zufallsbits wählen kann (d.h. alle $p_r^{xm} = 0$), sagen wir, dass das Protokoll einen Fehler macht. Betrachten wir nun zuerst die Situation in P bei Eingabe (x, y). Die Wahrscheinlichkeit, dass Alice die Zufallsbits r wählt, ist gleich $\sum_m \Pi^x(m) p_r^{xm}$. In Q ist diese Wahrscheinlichkeit dahingegen $\sum_m \Pi(m) p_r^{xm}$. Die Totalvariationsdistanz dieser beiden Verteilungen (bei Eingabe (x, y)) ist somit

$$\begin{aligned}
\sum_r |\sum_m p_r^{xm}(\Pi^x(m) - \Pi(m))| &\leq \sum_r \sum_m p_r^{xm} |\Pi^x(m) - \Pi(m)| \\
&= \sum_m \left(|\Pi^x(m) - \Pi(m)| \sum_r p_r^{xm} \right) \\
&= \sum_m |\Pi^x(m) - \Pi(m)| = \| \Pi^x - \Pi \|_1 .
\end{aligned}$$

Wenn wir einen parallelen Durchlauf von Q und P durchführen und die Zufallsbits von Alice identisch sind, dann ist auch die erste Nachricht identisch und da Q danach völlig analog wie P vorgeht, ist auch die Fehlerwahrscheinlichkeit identisch. Nun können wir die Fehlerwahrscheinlichkeit von Q auf einer Eingabe (x, y) abschätzen.

$$\begin{aligned}
\varepsilon_Q(x, y) &= \sum_{r,m} \Pi(m) p_r^{xm} \cdot \Pr[\text{Fehler in } Q \text{ auf } (x, y) \mid r, m] \\
&= \sum_{r,m} (\Pi(m) p_r^{xm} - \Pi^x(m) p_r^{xm} + \Pi^x(m) p_r^{xm}) \\
&\quad \cdot \Pr[\text{Fehler in } Q \text{ auf } (x, y) \mid r, m] \\
&= \sum_{r,m} \Pi^x(m) p_r^{xm} \cdot \Pr[\text{Fehler in } Q \text{ auf } (x, y) \mid r, m] \\
&\quad + \sum_{r,m} (\Pi(m) p_r^{xm} - \Pi^x(m) p_r^{xm}) \cdot \Pr[\text{Fehler in } Q \text{ auf } (x, y) \mid r, m] \\
&\leq \varepsilon_P(x, y) + \sum_{r,m} (\Pi(m) p_r^{xm} - \Pi^x(m) p_r^{xm}) \\
&\overset{\text{Beob. 3.2.17}}{\leq} \varepsilon_P(x, y) + (1/2) \sum_r |\sum_m p_r^{xm} (\Pi^x(m) - \Pi(m))| \\
&\leq \varepsilon_P(x, y) + (1/2) \| \Pi^x - \Pi \|_1
\end{aligned}$$

Das bedeutet, der durchschnittliche Fehler von Q ist

$$\begin{aligned}
\varepsilon_Q^\mu &= \sum_{x,y} \mu(x,y)\varepsilon_Q(x,y) \\
&\leq \sum_{x,y} \mu(x,y)\varepsilon_P(x,y) + \sum_{x,y} \mu(x,y)(1/2) \parallel \Pi^x - \Pi \parallel_1 \\
&= \varepsilon_P + (1/2) \sum_x \mu(x) \parallel \Pi^x - \Pi \parallel_1 \\
&\overset{\text{Theorem 3.2.19}}{\leq} \varepsilon_P + (1/2)\sqrt{(2\ln 2) I(X:M)}
\end{aligned}$$

Wenn wir Q als randomisiertes Protokoll mit öffentlichem Zufall betrachten, kann Bob sich die erste Nachricht von Alice aus den Zufallsbits von Alice selber konstruieren, da die Nachricht unabhängig von der Eingabe von Alice ist. Das liefert uns ein randomisiertes $[k-1, l_2, \ldots, l_k]$ public coin Protokoll Q' mit $\varepsilon_{Q'}^\mu = \varepsilon_Q^\mu$. Analog zum Beweis der $\max\{D_{\mu,\varepsilon}(f) \mid \mu \text{ Verteilung auf } X \times Y\} \leq R_\varepsilon^{pub}(f)$ Aussage des Minimax-Prinzips (Theorem 3.2.3) gibt es dann eine Belegung der Zufallsbits, so dass das daraus resultierende deterministische $[k-1, l_2, \ldots, l_k]$ Protokoll einen Fehler (nur noch bzgl. μ) von höchstens $\varepsilon_{Q'}^\mu \leq \varepsilon_P + (1/2)\sqrt{(2\ln 2) I(X:M)}$ hat. Damit gilt die Behauptung. □

Mit dieser Methode zur Runden-Elimination von Protokollen mit einer Eingabeverteilung können wir nun das REL beweisen.

Lemma 3.3.4 (Round Elimination Lemma)**.** *Angenommen, die Funktion $GT_n^{(m),A}$ hat ein randomisiertes $[k, l_1, ..., l_k]^A$ public coin Protokoll mit Fehler kleiner als δ. Dann existiert ein randomisiertes $[k-1, l_2, ..., l_k]^B$ public coin Protokoll für GT_n mit Fehler kleiner als $\delta + (1/2)\sqrt{(2l_1 \ln 2)/m}$.*

Beweis. Sei P das randomisierte $[k, l_1, ..., l_k]^A$ public coin Protokoll mit Fehler ε_P kleiner als δ für $GT_n^{(m),A}$. Wir konstruieren für jede Eingabeverteilung μ auf $\{0,1\}^n \times \{0,1\}^n$ ein deterministisches $[k-1, l_2, ..., l_k]^B$ Protokoll für GT_n mit Fehler kleiner als $\delta + (1/2)\sqrt{(2l_1 \ln 2)/m}$. Mit dem Minimax-Prinzip (Theorem 3.2.3) folgt dann auch die Existenz des gewünschten randomisierten $[k-1, l_2, ..., l_k]^B$ public coin Protokolls für GT_n.

Zuerst konstruieren wir uns folgendermaßen eine Eingabeverteilung μ' für $GT_n^{(m),A}$:

- Wähle $i \in \{1, \ldots, m\}$ zufällig gleichverteilt.
- Wähle Paare (x_j, y_j) für $j = 1, \ldots, m$ nach der Verteilung μ. Setze $y = y_i$.

- $x_1, \ldots, x_m$ ist die Eingabe für Alice und i, y die Eingabe für Bob.

Nach dem Minimax-Prinzip folgt aus P die Existenz eines deterministischen $[k, l_1, ..., l_k]^A$ Protokolls P' mit Fehler $\varepsilon_{P'}^{\mu'} = \varepsilon_P$.

Sei $X = X_1 \ldots X_m$ die Zufallsvariable für die Eingabe von Alice, wobei X_i die Zufallsvariable für die Eingabe x_i bezeichnet, M die Zufallsvariable der ersten Nachricht von Alice in P' und Y, I die Zufallsvariablen für die Eingabe von Bob, wenn die Eingaben nach μ' gezogen werden. Wir definieren folgende bedingte Verteilungen

$$\begin{aligned}
\mu'_{i;x_1,\ldots,x_{i-1}} &= \text{Verteilung } \mu' \text{ unter den Bedingungen } I = i \text{ und } X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, \\
\mu'_{i,y;x_1,\ldots,x_i} &= \text{Verteilung } \mu' \text{ unter den Bedingungen } I = i, Y = y \text{ und } X_1 = x_1, \ldots, X_i = x_i.
\end{aligned}$$

Nun nutzen wir die informationstheoretischen Grundlagen aus, um die erwartete wechselseitige Information von X_i und M für Bob nach oben abzuschätzen. Bob kennt nach Definition des Kommunikationsproblems $GT_n^{(m),A}$ die Eingaben $x_1, \ldots, x_{i-1}$. Das bedeutet, die erwartete Information (bzgl. μ'), die er über x_i durch die erste Nachricht erhält, ist $E_{i,X}[I(X_i : M \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]$ (der Index des Erwartungswerts gibt die Variable(n) an, über die der Durchschnitt gebildet wird). Nach Lemma 3.2.15 (3) gilt

$$I(X_i : M \mid X_1, \ldots, X_{i-1}) = I(X_i : M, X_1, \ldots, X_{i-1}) - I(X_i : X_1, \ldots, X_{i-1}).$$

Da die Zufallsvariablen $X_1, \ldots, X_m$ unabhängig sind, gilt $I(X_i : X_1, \ldots, X_{i-1}) = 0$ und somit

$$E_{i,X}[I(X_i : M \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] = E_i[I(X_i : M, X_1, \ldots, X_{i-1})].$$

Wegen der Unabhängigkeit von $X_1, \ldots, X_n$ gilt nach Lemma 3.2.15 (2) weiter

$$I(X_i : M, X_1, \ldots, X_{i-1}) = I(X_1, \ldots, X_i : M) - I(X_1, \ldots, X_{i-1} : M)$$

und damit

$$\begin{aligned}
E_i[I(X_i : M, X_1, \ldots, X_{i-1})] &= \frac{1}{m} \sum_i I(X_1, \ldots, X_i : M) - I(X_1, \ldots, X_{i-1} : M) \\
&= \frac{I(X : M)}{m} \leq \frac{l_1}{m}, \qquad (3.1)
\end{aligned}$$

wobei die letzte Ungleichung aus Theorem 3.2.11 folgt. Nach dem Satz der totalen Wahrscheinlichkeit gilt natürlich

$$E_{i,X}[\varepsilon_{P'}^{\mu'_{i;x_1,\ldots,x_{i-1}}}] = \varepsilon_{P'}^{\mu'} = \varepsilon_P < \delta. \tag{3.2}$$

Für alle $i \in \{1,\ldots,m\}$ und $x_1,\ldots,x_{i-1} \in \{0,1\}^n$ konstruieren wir nun ein randomisiertes $[k, l_1, \ldots, l_k]^A$ private coin Protokoll $P'_{i;x_1,\ldots,x_{i-1}}$ für GT_n:

- Eingabe: $(x,y) \in \{0,1\}^n \times \{0,1\}^n$
- Alice setzt $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_i = x$ und wählt $x_{i+1}, \ldots, x_m$ zufällig nach der durch μ induzierten Verteilung für die Eingaben von Alice.
- Bob setzt $I = i$ und $Y = y$.
- Nun lassen sie das Protokoll P' auf ihren Eingaben laufen.

Für eine feste Eingabe (x,y) ist der Fehler von $P'_{i;x_1,\ldots,x_{i-1}}$ gleich dem durchschnittlichem Fehler von P' unter der Verteilung $\mu'_{i,y;x_1,\ldots,x_i}$. Damit gilt insgesamt

$$\varepsilon^{\mu}_{P'_{i;x_1,\ldots,x_{i-1}}} = \varepsilon_{P'}^{\mu'_{i;x_1,\ldots,x_{i-1}}}. \tag{3.3}$$

Mit M' bezeichnen wir die Zufallsvariable der ersten Nachricht von Alice und X' die Eingabe von Alice in $P'_{i;x_1,\ldots,x_{i-1}}$. Die Information, die Bob von der ersten Nachricht in P' über die Eingabe X_i erfährt, wenn er weiß, dass $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$ ist, ist die gleiche Information, die Bob von der ersten Nachricht in $P'_{i;x_1,\ldots,x_{i-1}}$ über die Eingabe von Alice erhält. D.h., es gilt

$$I(X_i : M \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) = I(X' : M'),$$

wobei die linke Seite die wechselseitige Information in P' bzgl. der Verteilung μ' und die rechte Seite die wechselseitige Information in $P'_{i;x_1,\ldots,x_{i-1}}$ bzgl. der Verteilung μ ist. Mit dem Hilfslemma 3.3.3 erhalten wir aus dem Protokoll $P'_{i;x_1,\ldots,x_{i-1}}$ ein deterministisches $[k-1, l_2, \ldots, l_k]$ Protokoll $Q_{i;x_1,\ldots,x_{i-1}}$ für GT_n mit

$$\begin{aligned}
\varepsilon^{\mu}_{Q_{i;x_1,\ldots,x_{i-1}}} &\leq \varepsilon^{\mu}_{P'_{i;x_1,\ldots,x_{i-1}}} + \frac{1}{2}\sqrt{(2\ln 2) I(X' : M')} \\
&\overset{(3.3)}{=} \varepsilon_{P'}^{\mu'_{i;x_1,\ldots,x_{i-1}}} + \frac{1}{2}\sqrt{(2\ln 2) I(X_i : M \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})}.
\end{aligned}$$

Damit haben wir (die Erwartungswerte sind bzgl. der Verteilung μ')

$$\begin{aligned}
E_{i,X}[\varepsilon^{\mu}_{Q_{i;x_1,\ldots,x_{i-1}}}] &\leq E_{i,X}[\varepsilon^{\mu'_{i;x_1,\ldots,x_{i-1}}}_{P'}] \\
&\quad + E_{i,X}[\frac{1}{2}\sqrt{(2\ln 2)I(X_i : M \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})}] \\
&\overset{\text{Theorem 3.2.6}}{\leq} E_{i,X}[\varepsilon^{\mu'_{i;x_1,\ldots,x_{i-1}}}_{P'}] \\
&\quad + \frac{1}{2}\sqrt{(2\ln 2)E_{i,X}[I(X_i : M \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})]} \\
&\overset{(3.2)}{<} \delta + \frac{1}{2}\sqrt{(2\ln 2)E_{i,X}[I(X_i : M \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})]} \\
&\overset{(3.1)}{\leq} \delta + \frac{1}{2}\sqrt{(2\ln 2)\frac{l_1}{m}}.
\end{aligned}$$

D.h., es existieren Werte für $i, x_1, \ldots, x_{i-1}$ mit $\varepsilon^{\mu}_{Q_{i;x_1,\ldots,x_{i-1}}} < \delta + \frac{1}{2}\sqrt{(2\ln 2)\frac{l_1}{m}}$. Dies ist unser gewünschtes deterministisches $[k-1, l_2, \ldots, l_k]$ Protokoll für GT_n und Eingabeverteilung μ. □

3.4 Untere Schranke für die randomisierte Kommunikationskomplexität von GT

Das REL liefert nicht direkt eine untere Schranke für die Kommunikationskomplexität von einer Funktion. Die Idee ist es, eine Reduktion von $GT^{(m),A}$ auf GT anzugeben, um iterativ die Reduktion und das REL anzuwenden, bis wir unter bestimmten Annahmen ein 0-Runden Protokoll mit Fehlerwahrscheinlichkeit kleiner als 1/2 für ein nicht triviales Problem haben. Dies ist ein Widerspruch (siehe 2.2.1), d.h., unsere Annahmen sind nicht korrekt und erreichen so eine untere Schranke (wenn die Annahmen einen Schluss bzgl. der Kommunikationskomplexität zulassen). Diese Methode wenden wir nun auf GT an und zeigen damit eine lineare untere Schranke für die randomisierte Kommunikationskomplexität mit öffentlichem Zufall (siehe [SV08]).

Theorem 3.4.1. *Die Komplexität eines randomisierten k-Runden Protokolls mit beschränktem Fehler $\varepsilon < 1/2$ für GT_n ist nach unten beschränkt durch $\Omega(n^{1/k}k^{-2})$.*

Beweis. Reduktion von $GT^{(t),A}_{n/t}$ auf GT_n (siehe [MNSW98]):

- Eingabe: $x_1, \ldots, x_t \in \{0,1\}^{n/t}$, $i \in \{1, \ldots, t\}$ und $y \in \{0,1\}^{n/t}$

- Alice konkateniert ihre Eingabe zu $\hat{x} = x_1x_2\cdots x_t \in \{0,1\}^n$.
- Bob erzeugt $\hat{y} = x_1x_2\cdots x_{i-1}y1^{n-in/t}$ (Erinnerung: Bob kennt $x_1,\ldots,x_{i-1}$ nach Definition von $GT^{(t),A}_{n/t}$).

Nach Konstruktion stimmen also die ersten $(i-1)n/t$ Bits von $\hat{x}$ und $\hat{y}$ überein. Die $n-in/t$ niedrigstwertigen Bits von $\hat{y}$ sind mindestens so groß wie die in $\hat{x}$. Damit gilt $\hat{x} > \hat{y}$ genau dann, wenn $x > y$.

Angenommen, wir haben ein randomisiertes public coin Protokoll für GT_n mit beschränktem zweiseitigen Fehler und Komplexität c. Wir können dies als randomisiertes $[k,c,\ldots,c]$ public coin Protokoll für GT_n und Fehler kleiner als 1/3 betrachten (probability amplification benötigt eventuell konstanten Zusatzfaktor für die Komplexität). Wir definieren $t := (2\ln 2)(9k^2)c$. Angenommen es gilt $n \geq t^k$. Mit der Reduktion erhalten wir ein $[k,c,\ldots,c]$ Protokoll für $GT^{(t),A}_{n/t}$ mit Fehler kleiner als 1/3. Wenden wir das REL auf dieses Protokoll an, erhalten wir ein randomisiertes $[k-1,c,\ldots,c]^B$ Protokoll für $GT_{n/t}$ mit Fehler kleiner als $1/3 + (1/2)\sqrt{(2\ln 2)c/t}$. Daraus konstruieren wir mit der Reduktion wieder ein $GT^{(t),B}_{n/t^2}$ Protokoll und wenden erneut das REL an, um ein $[k-2,c,\ldots,c]$ Protokoll für GT_{n/t^2} zu erhalten, usw. Wenn wir Reduktion und REL abwechselnd j-mal anwenden, erhalten wir ein $[k-j,c,\ldots,c]^Z$ Protokoll für GT_{n/t^j} mit Fehler kleiner als

$$(1/3) + \sum_{i=1}^{j}(1/2)\sqrt{(2\ln 2)c/t} = (1/3) + \sum_{i=1}^{j}(1/2)\sqrt{\frac{1}{9k^2}} = (1/3) + \sum_{i=1}^{j}\frac{1}{6k}$$

und $Z = A$, falls $j \bmod 2 = 1$, und $Z = B$ sonst. Das bedeutet, wenn wir dies k-mal anwenden, erhalten wir ein 0-Runden Protokoll für GT_{n/t^k} mit $n/t^k \geq 1$ und Fehlerwahrscheinlichkeit kleiner als

$$(1/3) + \sum_{i=1}^{k}\frac{1}{6k} = 1/2.$$

D.h., wir haben ein 0-Runden Protokoll für ein nicht triviales Problem und Fehler kleiner als 1/2. Dies ist ein Widerspruch (siehe Beobachtung 2.2.1). Also gilt $n < t^k$ und damit

$$(2\ln 2)(9k^2)c = t > \sqrt[k]{n} \Rightarrow c = \Omega(n^{1/k}k^{-2}).$$

□

Da eine untere Schranke für die randomisierte Kommunikationskomplexität für public coin Protokolle auch eine untere Schranke für private coin Protokolle ist, liefert uns dieses

Theorem eine lineare untere Schranke für alle randomisierten Protokolle mit zweiseitigem Fehler und konstant vielen Runden. Insbesondere ist die Ein-Runden-Komplexität von GT_n linear, so dass wir dieses Resultat zum Beweis von exponentiellen unteren Schranken für die Größe randomisierter OBDDs verwenden können.

4 Untere Schranken für die Größe von randomisierten OBDDs für die Multiplikation

In diesem Kapitel untersuchen wir die Größe von randomisierten OBDDs, die einzelne Ergebnisbits von der Multiplikation zweier Binärzahlen berechnen. Wenn wir zwei Binärzahlen x und y der Länge n miteinander multiplizieren, erhalten wir als Ergebnis eine Binärzahl z der Länge höchstens $2n$. Wir werden zeigen, dass zwei Ergebnisbits besondere Bedeutungen haben: Einmal das sogenannte mittlere Bit der Multiplikation, d. h. das Bit z_{n-1} mit der Wertigkeit 2^{n-1}, und das höchstwertige Bit der Multiplikation, d. h. das Bit z_{2n-1} mit der Wertigkeit 2^{2n-1}. Im ersten Abschnitt definieren wir einige Notationen, die wir im Laufe dieses Kapitels benötigen. Da wir uns mit dem Berechnen und der Manipulation von Bitstrings beschäftigen, hilft uns das, die Lesbarkeit der Beweise zu erhöhen. Im zweiten Abschnitt werden wir zeigen, dass wir mit Hilfe von einer Read-Once-Projektion eine untere Schranke für die Größe von randomisierten OBDDs für die Berechnung des höchstwertigen Bits auf die Größe von randomisierten OBDDs für die Berechnung der restlichen Bits übertragen können. Dahingegen ist eine obere Schranke für die Größe von randomisierten OBDDs für die Berechnung des mittleren Bits auch eine obere Schranke für die Größe von randomisierten OBDDs für alle anderen Ergebnisbits. Im dritten Abschnitt zeigen wir eine untere Schranke für die Größe von randomisierten OBDDs für das höchstwertige Bit der Multiplikation, indem wir ein Resultat für die Größe von deterministischen OBDDs von B. Bollig aus [Bol10] modifizieren, um eine Rechteckreduktion von GT auf die Berechnung des höchstwertigen Bits zu konstruieren. Somit erhalten wir aus den Ergebnissen aus Kapitel 3 eine exponentiell große untere Schranke für die Größe randomisierter OBDDs für das höchstwertige Bit der Multiplikation. Im letzten Abschnitt

folgern wir aus den Vorüberlegungen aus dem ersten Abschnitt und der unteren Schranke für das höchstwertige Bit auch eine exponentiell große untere Schranke für die Größe von randomisierten OBDDs für das mittlere Bit der Multiplikation.

4.1 Notation

Wir legen uns auf folgende Notationen fest:

- Sei $x, y \in \{0,1\}^n$. Ein Teilstring von x bezeichnen wir mit $x_j^i := x_i \cdots x_j$ für $0 \leq j \leq i \leq n-1$ und die Konkatenation der Bitsstrings x und y bezeichnen wir mit xy.
- Mit $0^j = \underbrace{0 \cdots 0}_{j \text{ mal}}$ bezeichnen wir den Bitstring, der aus genau $j \in \mathbb{N}$ Nullen besteht. Analog definieren wir $1^j = \underbrace{1 \cdots 1}_{j \text{ mal}}$.
- Sei $l \in \{0, \ldots, 2^m - 1\}$ und $x \in \{0,1\}^n$. Wir definieren $\overline{l} := (2^m - 1) - l$ und analog $\overline{x} = (\overline{x}_{n-1}, \ldots, \overline{x}_0)$.
- Für $l \in \mathbb{N}$ bezeichnen wir mit $[l] \in \{0,1\}^{\lfloor \log l \rfloor + 1}$ die Binärdarstellung der Länge $\lfloor \log l \rfloor + 1$ von l. Umgekehrt bezeichnen wir mit $|x|$ die ganze Zahl, die der Bitstring $x \in \{0,1\}^n$ darstellt, d.h. $|x| = \sum_{i=0}^{n-1} 2^i \cdot x_i$.
- Sei $l \in \{0, \ldots, 2^m - 1\}$. Für $m \in \mathbb{N}$ gibt es $q, r \in \mathbb{N}_0$ mit $q \geq 0$ und $0 \leq r < m$, so dass $l = q \cdot m + r$ gilt. Mit $l \operatorname{div} m := q$ bezeichnen wir das Ergebnis der ganzzahligen Division von l und m und mit $l \bmod m := r$ den Rest der Division. Falls $m = 2^i$ mit $1 \leq i \leq \lfloor \log l \rfloor$ ist, können die Zahlen q und r sehr leicht aus der Binärdarstellung von l bestimmt werden. Es gilt

$$l \operatorname{div} 2^i = \left| [l]_i^{\lfloor \log l \rfloor} \right| \quad \text{und} \quad l \bmod 2^i = \left| [l]_0^{i-1} \right|.$$

4.2 Einleitung

In diesem Kapitel wollen wir uns die Größe randomisierter OBDDs für die Multiplikation anschauen. Genauer gesagt betrachten wir randomisierte OBDDs für die Funktion

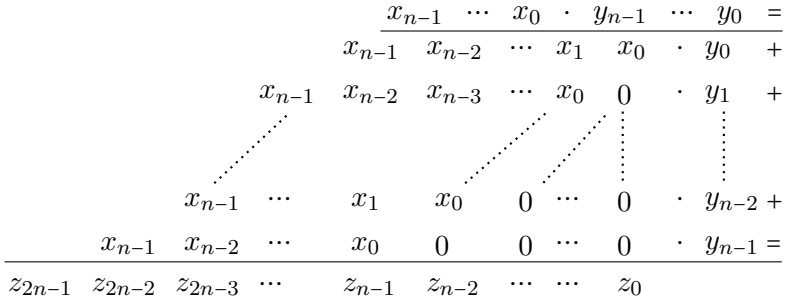

Abbildung 4.1: Schriftliche Multiplikation zweier Bitstrings $x, y \in \{0,1\}^n$

$MUL_{i,n} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ mit $MUL_{i,n}(x,y) = z_i$ und $i = 0,\ldots,2n-1$, wobei $z_{2n-1}\ldots z_0 = x \cdot y$ ist, d.h., $MUL_{i,n}$ berechnet das $i+1$-te Bit des Produkts von x und y. Hierbei steht z_0 für das niedrigstwertige Bit und z_{2n-1} für das höchstwertige Bit des Produkts $x \cdot y$. Analog zur Berechnung des Produkts zweier natürlicher Zahlen kann die Berechnung von $MUL_{i,n}(x,y)$ anschaulich mit Hilfe der schriftlichen Multiplikation durchgeführt werden (siehe Abb. 4.1).

Wir machen uns die Beobachtung 2.3.6 zu Nutze, um die Komplexität der Funktionen $MUL_{i,n}$ mit Hilfe von Read-Once-Projektionen untereinander zu vergleichen. Es ist offensichtlich, dass nicht jede Position gleich schwierig oder gleich einfach ist. In dem ersten Lemma in diesem Kapitel halten wir folgende Besonderheit des mittleren ($i = n-1$) und des höchstwertigen ($i = 2n-1$) Bits der Multiplikation fest:

Falls es ein OBDD der Größe $s(n)$ für die Funktion $MUL_{n-1,n}$ gibt, dann können alle Funktionen $MUL_{i,n}$ mit $i = 0,\ldots,2n-1$ mit einem OBDD der Größe $s(2n)$ berechnet werden. D.h., das mittlere Bit ist in diesem Sinne am schwierigsten zu berechnen, denn wir können eine untere Schranke für die OBDD Größe von $MUL_{i,n}$ mit $0 \le i \le 2n-1$ auf $MUL_{n-1,n}$ übertragen. Im Gegensatz dazu können wir zeigen, wenn kein OBDD der Größe $s(n)$ für das höchstwertige Bit der Multiplikation existiert, dann existiert auch kein OBDD der Größe $s(n/2)$ für die Funktionen $MUL_{i,n}$ mit $n-1 \le i < 2n-1$ bzw. der Größe $s(i/2)$ für $MUL_{i,n}$ mit $0 < i < n-1$. Also ist das höchstwertige Bit sozusagen am einfachsten zu berechnen. Im Fall $i = 0$ kann der Funktionswert von den jeweiligen niedrigstwertigen Bits von x und y abgelesen werden und ist daher uninteressant.

Lemma 4.2.1. *Sei $n \in \mathbb{N}$. Dann gilt*

$$\begin{array}{lll}
(a) & MUL_{i,n} \leq_{rop} MUL_{2n-1,2n} & \text{für} \quad 0 \leq i \leq 2n-1 \\
(b) & MUL_{2n-1,n} \leq_{rop} MUL_{i,2n} & \text{für} \quad 2n-1 \leq i < 4n-1 \\
(c) & MUL_{2i-1,i} \leq_{rop} MUL_{2i-1,2n} & \text{für} \quad 0 < i \leq n-1 \\
 & MUL_{2i-1,i} \leq_{rop} MUL_{2i,2n} & \text{für} \quad 0 < i \leq n-1
\end{array}$$

Beweis. (a) Sei $(x,y) \in \{0,1\}^n \times \{0,1\}^n$ die Eingabe für $MUL_{i,n}$ mit $0 \leq i \leq 2n-1$. Wir verändern die Eingabe (x,y) in die Eingabe (x',y') für $MUL_{2n-1,2n}$ und verschieben dabei die Berechnung des i-ten Bits von $x \cdot y$ an die Stelle $2n-1$ des Produkts $x' \cdot y'$. Um nun ein beliebiges Ergebnisbit an der Stelle $0 \leq i \leq n-1$ an die Position $2n-1$ zu verschieben, verwerfen wir die $n-(i+1)$ führenden Bits von x und fügen vor den i übrigen Bits $2n-1-i$ Nullen ein. Falls $n-1 < i \leq 2n-1$ ist, brauchen wir keine Bits zu verwerfen, sondern fügen vor der x-Eingabe $2n-1-i$ Nullen und nach der x-Eingabe $i+1-n$ Nullen ein. Die y-Eingabe füllen wir mit n führenden Nullen auf. Formal definieren wir

$$x' := \begin{cases} x_0^i 0^{2n-1-i} & \text{für} \quad 0 \leq i \leq n-1 \\ 0^{i+1-n} x 0^{2n-1-i} & \text{für} \quad n-1 < i \leq 2n-1 \end{cases}$$

und

$$y' := 0^n y.$$

Durch das Einfügen von Nullen vor der x-Eingabe (oder einem Teil der x-Eingabe) verschieben wir die Bits des Produkts $x \cdot y$. Eingefügte führende Nullen verändern natürlich das Ergebnis nicht (siehe Abb. 4.2). Es gilt also $x' \cdot y' = x' \cdot y$. Durch das Einfügen der $2n-1-i$ Nullen in x' verschieben wir die Berechnung des i-ten Ergebnisbits von $x \cdot y$ zur Position $2n-1$ in $x' \cdot y$. Da wir im Fall $0 \leq i \leq n-1$ nur irrelevante Bits von x verwerfen und sonst alle x-Bits benutzen, ist die Konstruktion insgesamt korrekt und es folgt $MUL_{i,n} \leq_{rop} MUL_{2n-1,2n}$.

(b) Sei nun (x,y) eine Eingabe für die Funktion $MUL_{2n-1,n}$ und $2n-1 \leq i < 4n-1$ beliebig aber fest. Das Einfügen von $i-(2n-1)$ Nullen vor der x-Eingabe verschiebt das höchstwertige Bit von $x \cdot y$ an die Stelle i von $x0^{i-(2n-1)} \cdot y$. Für $i \geq 3n-2$ würden wir aber mehr als $2n$ Bits für die Eingabe für $MUL_{i,2n}$ benötigen. Daher müssen wir in diesem Fall

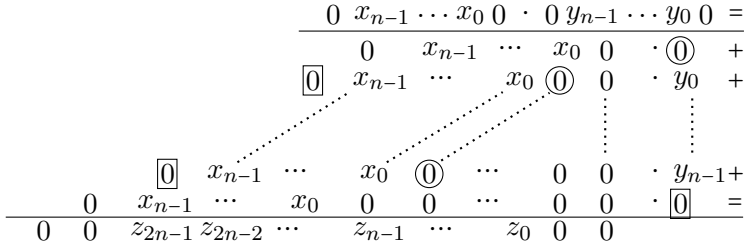

Abbildung 4.2: Einfügen von Nullen vor x und y (Kreis) und am Ende von x und y (Rechteck) mit $z = x \cdot y$

auch vor der y-Eingabe Nullen einfügen.

$$x' := \begin{cases} 0^{3n-1-i}x0^{i-(2n-1)} & \text{für} \quad 2n-1 \le i \le 3n-2 \\ x0^n & \text{für} \quad 3n-1 \le i \le 4n-2 \end{cases}$$

und

$$y' := \begin{cases} 0^n y & \text{für} \quad 2n-1 \le i \le 3n-2 \\ 0^{4n-1-i}y0^{i-(3n-1)} & \text{für} \quad 3n-1 \le i \le 4n-2 \end{cases}.$$

Insgesamt gilt somit $MUL_{2n-1,n}(x,y) = MUL_{i,2n}(x',y')$.

(c) Angenommen, es gilt $MUL_{2i-1,2n} \le_{rop} MUL_{2i,2n}$ für $0 < i \le n-1$. Dann reicht es $MUL_{2i-1,i} \le_{rop} MUL_{2i-1,2n}$ zu zeigen, denn aus der Transitivität von $\le_{rop}$ folgt dann auch $MUL_{2i-1,i} \le_{rop} MUL_{2i,2n}$.

Für die Aussage $MUL_{2i-1,i} \le_{rop} MUL_{2i-1,2n}$ sei $(x,y) \in \{0,1\}^i$ eine Eingabe von $MUL_{2i-1,i}$ mit $0 < i \le n-1$, d.h. es gilt $2n > i$. Da die zu berechnende Position $2i-1$ bei beiden Funktionen gleich ist, unterscheiden sich $MUL_{2i-1,i}$ und $MUL_{2i-1,2n}$ nur in der Länge der Eingabe. Wir setzen $x' = 0^{2n-i}x$ und $y' = 0^{2n-i}y$. Durch das Auffüllen mit Nullen ist das Produkt $x' \cdot y'$ gleich dem Produkt von $x \cdot y$. Somit ist unsere Konstruktion korrekt und es gilt $MUL_{2i-1,i} \le_{rop} MUL_{2i-1,2n}$.

Bleibt noch $MUL_{2i-1,2n} \le_{rop} MUL_{2i,2n}$ für $0 < i \le n-1$ zu zeigen. Dazu sei $u,v \in \{0,1\}^{2n}$ eine Eingabe für $MUL_{2i-1,2n}$. Wir setzen $u' = u_{2n-1}\dots u_0 0$ und $v' = v$ und verschieben so die Berechnung des Bits an Position $2i-1$ von $x \cdot y$ an die Position $2i$ von $u' \cdot v'$. Da für die Berechnung der Positionen $0,\dots,2n-2$ von $u \cdot v$ das höchstwertige Bit u_{2n-1} von u nicht benötigt wird (siehe Abb. 4.1), gilt $MUL_{2i-1,2n}(u,v) = MUL_{2i,2n}(u',v')$ und damit $MUL_{2i-1,2n} \le_{rop} MUL_{2i,2n}$. □

Diese Reduktionen motivieren uns, untere Schranken für die Größe randomisierter OBDDs für das höchstwertige Bit der Multiplikation anzuschauen. Wir werden im nächsten Kapitel eine exponentielle untere Schranke für die Größe randomisierter OBDDs zeigen, die $MUL_{2n-1,n}$ berechnen.

4.3 Eine exponentielle untere Schranke für die Größe randomisierter OBDDs für das höchstwertige Bit der Multiplikation

Nun haben wir alle nötigen Werkzeuge zusammen, um eine untere Schranke für das höchstwertige Bit zu beweisen. Die Idee ist, die Berechnung des höchstwertigen Bits auf GT bzw. $\overline{GT}$ zurückzuführen, indem gewisse Teile der Eingabe konstant gesetzt werden. D.h., wir wollen für jede Variablenordnung π eine Rechteckreduktion von GT_m auf die partitionierte Version $MUL^{\pi,p}_{2n-1,n}$ mit p entsprechend gewählt und $m = \Theta(n)$ konstruieren, so dass aus der Kommunikationskomplexität von GT aus Kapitel 3 die exponentiell große untere Schranke für $MUL^{\pi,p}_{2n-1,n}$ folgt. Die Reduktion basiert im wesentlichen auf der Reduktion aus [Bol10], die für deterministische OBDDs benutzt wurde. Wir werden die Reduktion so modifizieren, dass wir am Ende eine Rechteckreduktion erhalten.

Wenn es den Beweis nicht wesentlich beeinflusst, werden wir zur besseren Lesbarkeit Gaußklammern weglassen, auch wenn die Zahl ganzzahlig sein muss.

Im ersten Unterabschnitt betrachten wir die Berechnung des höchstwertigen Bits der Multiplikation für x-Eingaben von der Form $2^{n-1} + l \cdot 2^{n/3}$. Es ist klar, dass $MUL_{2n-1,n}(x,y) = 1 \Leftrightarrow |x| \cdot |y| \geq 2^{2n-1}$. Wir werden zeigen, dass die kleinste Binärzahl y' mit $|x| \cdot |y'| \geq 2^{2n-1}$ folgende Eigenschaften hat (siehe Abb. 4.3 für die Struktur von x und y'):

- Die $(2/3)n$ höchstwertigen Bits sind im Großen und Ganzen identisch mit den Bits der Binärdarstellung von $2^{(2/3)n} - 2l$. Wenn in der Binärdarstellung von l eine 1 an einer niedrigwertigen Stelle vorkommt, ist ein großer Teil der Binärdarstellung von $2^{(2/3)n} - 2l$ identisch mit der Binärdarstellung von $\bar{l}$.

- Die $n/3$ niedrigstwertigen Bits sind im wesentlichen identisch mit der Binärdarstel-

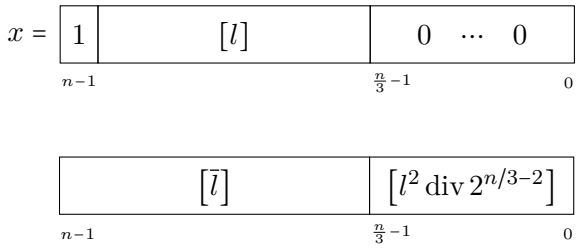

Abbildung 4.3: Binärdarstellung von der x-Eingabe und eine vereinfachte Darstellung der kleinsten Binärzahl y' mit $MUL_{2n-1,n}(x, y') = 1$

lung von $l^2 \operatorname{div} 2^{n/3-2}$.

Wir können die Binärdarstellung von l in der x-Eingabe ab Position $n/3$ wiederfinden. Die Charakterisierung der kleinsten Eingabe für y' mit $MUL_{2n-1,n}(x, y') = 1$ erlaubt es uns, die Berechnung des höchstwertigen Bits für eine Eingabe (x, y) in zwei Fälle zu unterteilen.

1. Falls ein Teil der $(2/3)n$ höchstwertigen Bits von $\overline{y}$ echt größer bzw. echt kleiner als ein Teil der $(2/3)n$ höchstwertigen Bits von x ist, dann ist $MUL_{2n-1,n}(x, y) = 0$ bzw. $MUL_{2n-1,n}(x, y) = 1$.

2. Falls die $(2/3)n$ höchswertigen Bits von y identisch mit den Bits von y' sind, dann ist $MUL_{2n-1,n}(x, y) = 1$ genau dann, wenn die $n/3$ niedrigstwertigen Bits von y in gewisser Weise größer oder gleich der Binärdarstellung von $l^2 \operatorname{div} 2^{n/3-2}$ ist.

Der erste Fall liefert uns eine Rechteckreduktion von GT auf $MUL_{2n-1,n}$, wenn wir die $n/3$ niedrigstwertigen Bits von y kleiner als die Bits der kleinsten 1-Eingaben setzen (wir zeigen, dass es reicht die Bits auf 0 zu setzen). Diese Reduktion liefert nur dann eine exponentielle untere Schranke, wenn die x- und y-Bits getrennt werden können, die für die „größer als“-Relation verglichen werden müssen. D.h., für jede Variablenordnung müssen wir eine Partition der Variablen finden, so dass die Variablen der einen Menge vor den Variablen der anderen Menge getestet werden, und die Bits mit der gleichen Wertigkeit nicht in derselben Menge sind. Wir werden sehen, dass es nicht für alle Variablenordnungen möglich ist, solch eine Partition zu finden. Daher brauchen wir auch den zweiten Fall für unsere Reduktion.

Um den zweiten Fall für eine Rechteckreduktion zu benutzen, müssen die x- und y-Bits ab der Stelle $n/3$ komplementär zueinander gesetzt werden können. Um zwei Variablen in

einer Rechteckreduktion komplementär zueinander zu setzen (wobei beide Variablen nicht konstant sind), müssen die beiden Variablen in derselben Partitionsmenge liegen. Da es sich hierbei um die gleichen Paare von Variablen wie im ersten Fall handelt, können wir für jede Variablenordnung eine Fallunterscheidung machen, ob viele dieser Paare getrennt werden können (Reduktion 1) oder viele Paare in derselben Partitionsmenge liegen (Reduktion 2).

Im zweiten Fall ist nicht direkt ersichtlich wie die Reduktion von GT auf $MUL_{2n-1,n}$ konstruiert werden kann. Das Ziel ist es, einen Teil der x-Eingabe in $l^2 \operatorname{div} 2^{n/3-2}$ wiederzufinden. Dazu betrachten wir l von der Form $2^d \cdot 2^m + w$ mit d und m geeignet gewählt. Dann gilt

$$l^2 = 2^{2m+2d} + w2^{m+d+1} + w^2.$$

Wir werden zeigen, dass wir w so wählen können, dass in der Binärdarstellung von l^2 ab einer Position j mit $j > n/3 - 2$ ein Teil der Binärdarstellung von w stehen muss. Dies erreichen wir, indem wir folgende Aussagen beweisen:

- In der Binärdarstellung von $w2^{m+d+1} + w^2$ können Überträge nur bis zu der Position $2m+1$ erzeugt oder weitergeleitet werden. Wenn die Länge der Binärdarstellung von w genau m ist, werden die $d-1$ höchstwertigen Bits der Binärdarstellung von w ab der Position $2m+2$ in der Binärdarstellung von $w2^{m+d+1} + w^2$ stehen müssen.
- Der Term 2^{2m+2d} ist zu groß, um einen Übertrag an den relevanten Stellen zu erzeugen.

Wir wählen m so, dass $2m+1 > n/3-2$ ist. Damit finden wir den Teil der Bits der Binärdarstellung von w auch in $l^2 \operatorname{div} 2^{n/3-2}$ wieder. Das bedeutet, wenn wir die niedrigstwertigen Bits von der y-Eingabe echt größer als die niedrigstwertigen Bits von der Binärdarstellung von $l^2 \operatorname{div} 2^{n/3-2}$ konstant setzen und die höchstwertigen Bits von der y-Eingabe gleich den höchstwertigen Bits von der Binärdarstellung von $l^2 \operatorname{div} 2^{n/3-2}$ setzen, bleiben die Bits der y-Eingabe noch „frei“, die größer oder gleich dem Teil der Binärdarstellung von w sein müssen, damit $MUL_{2n-1,n}(x,y) = 1$ gilt. Mit Hilfe des Parameters d und einem einfachen Abzählargument können wir zeigen, dass für jede Variablenordnung viele Paare der x- und y-Bits getrennt werden können, die für diese „größer oder gleich“-Relation benötigt werden.

Letztlich zeigen wir im ersten Unterabschnitt, dass eine leicht modifizierte Variante von GT, bei der kein Spieler ein x- und ein y-Bit der gleichen Wertigkeit kennt, nicht einfacher ist als GT. Dies erlaubt es uns, bei der Reduktion keine strikte Trennung der relevanten Bits zu fordern, sondern nur eine Trennung der Bits mit der gleichen Wertigkeit.

Im letzten Unterabschnitt fügen wir die Resultate zu einer Reduktion von $GT_{\Omega(n)}$ auf $MUL_{2n-1,n}$ zusammen und zeigen somit die exponentielle untere Schranke für die Größe von randomisierten OBDDs, die das höchstwertige Bit der Multiplikation berechnen.

4.3.1 Berechnung des höchstwertigen Bits der Multiplikation für x-Eingaben von der Form $2^{n-1} + l \cdot 2^{n/3}$

Es ist $MUL_{2n-1,n}(x,y) = 1$ für $x, y \in \{0,1\}^n$ genau dann, wenn $|x| \cdot |y| \geq 2^{2n-1}$ ist. Wir betrachten nun Eingaben für x von der Form $|x| = 2^{n-1} + l \cdot 2^{n/3}$ mit $1 \leq l < 2^{n/3-1}$ und schauen uns für diese Eingaben die Bedingung $|x| \cdot |y| \geq 2^{2n-1}$ genauer an, um so etwas über die Eingaben für y aussagen zu können, für die $MUL_{2n-1,n}(x,y) = 1$ gilt.

$$|x| \cdot |y| \geq 2^{2n-1} \Leftrightarrow |y| \geq \left\lceil \frac{2^{2n-1}}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil$$

Wenn wir die Division nun für ein paar Schritte ausführen, erhalten wir einen Ausdruck, der uns später noch nützlich sein wird.

$$\begin{aligned} \left\lceil \frac{2^{2n-1}}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil &= \left\lceil 2^n - \frac{l \cdot 2^{(4/3)n}}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil = \left\lceil 2^n - l \cdot 2^{n/3+1} + \frac{l^2 \cdot 2^{(2/3)n+1}}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil \\ &= 2^n - l \cdot 2^{n/3+1} + \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil \end{aligned}$$

Damit erhalten wir unsere erste wichtige Beobachtung (siehe [Bol10]).

Beobachtung 4.3.1. *Sei $n \in \mathbb{N}$, $x \in \{0,1\}^n$ mit $|x| = 2^{n-1} + l \cdot 2^{n/3}$ und $1 \leq l < 2^{n/3-1}$. Weiter sei $y \in \{0,1\}^n$. Es gilt $MUL_{2n-1,n}(x,y) = 1$ genau dann, wenn $|y|$ mindestens $2^n - l \cdot 2^{n/3+1} + \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil$ ist.*

Um die Lesbarkeit von späteren Beweisen zu verbessern, machen wir folgende Definition

$$LB(l) := 2^n - l \cdot 2^{n/3+1} + \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil.$$

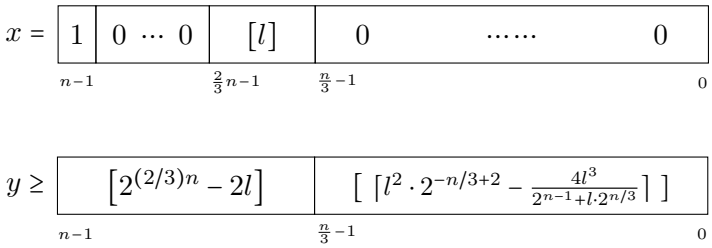

Abbildung 4.4: Bitstruktur von $|x| = 2^{n-1} + l \cdot 2^{n/3}$ und von der kleinsten y-Eingabe, so dass $MUL_{2n-1,n}(x,y) = 1$ gilt.

Wir können den Bruch $\frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}}$ nach oben abschätzen, wenn wir unsere Grenzen für $1 \le l < 2^{n/3-1}$ einsetzen

$$\frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \le \frac{4l^3}{2^{n-1}+2^{n/3}} < \frac{4\cdot 2^{n-3}}{2^{n-1}+2^{n/3}} = \frac{2^{n-1}}{2^{n-1}+2^{n/3}} < 1.$$

D.h., der Einfluss dieses Bruches auf das y ist gering und wir können ihn später gut kontrollieren, indem wir das l geeignet wählen. Für $l < 2^{n/3-1}$ gilt weiter

$$\begin{aligned} l^2 \cdot 2^{-n/3+2} &\le (2^{n/3-1}-1)^2 \cdot 2^{-n/3+2} \\ &= (2^{2/3n-2} - 2^{n/3} + 1)\cdot 2^{-n/3+2} \\ &= 2^{n/3} - 2^2 + 2^{-n/3+2} \le 2^{n/3} - 1 \quad \text{für } n \ge 2 \end{aligned}$$

und somit

$$\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil < 2^{n/3}.$$

Für $l > 0$ gilt weiter

$$l^2 \cdot 2^{-n/3+2} = \frac{4l^3}{l\cdot 2^{n/3}} > \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}}.$$

Somit ist $\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \rceil \ge 1$. Dies halten wir in folgender Beobachtung fest.

Beobachtung 4.3.2. *Sei $n \in \mathbb{N}$. Für $1 \le l < 2^{n/3-1}$ gilt $\frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} < 1$ und $1 \le \lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \rceil < 2^{n/3}$.*

Wenn wir uns die Beobachtung 4.3.1 noch etwas genauer anschauen, können wir wie in [Bol10] die Berechnung von $MUL_{2n-1,n}(x,y)$ unterteilen, indem wir die Bitstruktur

von $[LB(l)]$ ausnutzen. Nach Beobachtung 4.3.2 gilt $\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \rceil < 2^{n/3}$. Das bedeutet, in der Binärdarstellung von $LB(l)$ steht an den $n/3$ niedrigstwertigen Bits $[\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \rceil]$ und an den $(2/3)n$ höchstwertigen Bits $[2^{(2/3)n} - 2l]$ (siehe Abb. 4.4). Daher können wir zur Berechnung von $MUL_{2n-1,n}$ die y-Eingabe unterteilen in $y_0^{n/3-1}$ und $y_{n/3}^{n-1}$ und diese Teilbitstrings mit $[LB(l)]_0^{n/3-1}$ bzw. mit $[LB(l)]_{n/3}^{n-1}$ vergleichen. Wie wir im nächsten Lemma zeigen werden, ist die Binärdarstellung von $|x| = 2^{n-1} + l \cdot 2^{n/3}$ und $2^n - l \cdot 2^{n/3+1}$ sehr ähnlich und wir können daher $x_{n/3}^{n-1}$ und $y_{n/3}^{n-1}$ vergleichen, um $MUL_{2n-1,n}(x,y)$ zu berechnen.

Lemma 4.3.3. *Sei $n \in \mathbb{N}$, $x \in \{0,1\}^n$ mit $|x| = 2^{n-1} + l \cdot 2^{n/3}$ und $1 \le l < 2^{n/3-1}$. Weiter sei $0 \le j < n/3 - 1$ der kleinste Offset mit $x_{n/3+j} = 1$ und $y \in \{0,1\}^n$ mit $y_{n/3}^{n/3+j} = 0^{j+1}$ und $y_{n/3+j+1} = 1$. Dann gelten folgende Aussagen.*

(a) $\overline{x_{n/3+j+1}^{n-2}} = [2^{(2/3)n} - 2l]_{j+2}^{(2/3)n-1}$

(b) Falls $\overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2}$ *ist, dann gilt* $MUL_{2n-1,n}(x,y) = 1$.

(c) Falls $\overline{y_{n/3+j+2}^{n-1}} > x_{n/3+j+1}^{n-2}$ *ist, dann gilt* $MUL_{2n-1,n}(x,y) = 0$.

(d) Falls $\overline{y_{n/3+j+2}^{n-1}} = x_{n/3+j+1}^{n-2}$ *ist, dann gilt* $MUL_{2n-1,n}(x,y) = 1$ *genau dann, wenn* $|y_0^{n/3-1}| \ge \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil$.

Beweis. Zur Erinnerung: Für $|x| = 2^{n-1} + l \cdot 2^{n/3}$ mit $1 \le l < 2^{n/3-1}$ gilt

$$MUL_{2n-1,n}(x,y) = 1 \Leftrightarrow |y| \ge 2^n - l \cdot 2^{n/3+1} + \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil = LB(l) \quad (4.1)$$

und nach Beobachtung 4.3.2 gilt außerdem

$$\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil < 2^{n/3}.$$

Damit erhalten wir die Aussage

$$|y| \ge 2^n - l \cdot 2^{n/3+1} + 2^{n/3} \Rightarrow |y| \ge LB(l). \quad (4.2)$$

Falls $|y_{n/3}^{n-1}| = 2^{(2/3)n} - 2l$ ist, d.h., die $(2/3)n$ höchstwertigen Bits von y stimmen mit den Bits von $[LB(l)]_{n/3}^{n-1}$ überein (siehe Abb. 4.4), dann gilt nach (4.1)

$$MUL_{2n-1,n}(x,y) = 1 \Leftrightarrow |y_0^{n/3-1}| \ge \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil. \quad (4.3)$$

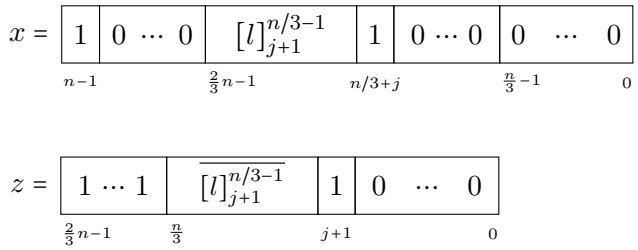

Abbildung 4.5: Bitstruktur von x und $z = [2^{(2/3)n} - 2l]$

Wir betrachten nun folgende Fallunterscheidung.

- 1. Fall: $|y_{n/3}^{n-1}| > 2^{(2/3)n} - 2l$
  In diesem Fall ist $|y| \geq 2^{n/3} \cdot (2^{(2/3)n} - 2l + 1) = 2^n - l \cdot 2^{n/3+1} + 2^{n/3}$. Nach (4.2) und (4.1) gilt dann $MUL_{2n-1,n}(x, y) = 1$.

- 2. Fall: $|y_{n/3}^{n-1}| < 2^{(2/3)n} - 2l$
  Hier gilt

  $$|y| < 2^{n/3} \cdot (2^{(2/3)n} - 2l) = 2^n - l \cdot 2^{n/3+1} \leq 2^n - l \cdot 2^{n/3+1} + \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil.$$

  Nach (4.1) gilt somit $MUL_{2n-1,n}(x, y) = 0$.

Um die Aussagen des Lemmas zu beweisen, ist es also wichtig die Aussage (a)

$$\overline{x_{n/3+j+1}^{n-2}} = [2^{(2/3)n} - 2l]_{j+2}^{(2/3)n-1}$$

zu zeigen. Dazu schauen wir uns nun die Struktur von x und $[2^{(2/3)n} - 2l]$ etwas genauer an. Sei z die Binärdarstellung der Länge $(2/3)n$ von $2^{(2/3)n} - 2l$, d.h. $|z| = 2^{(2/3)n} - 2l$. An den ersten $j+1$ Stellen von z stehen Nullen und an der Stelle $j+1$ steht eine Eins. Anschaulich kann man sich $2^{(2/3)n} - 2l$ als $(2^{(2/3)n} - 1) - 2l + 1$ vorstellen, d.h., in dem Bitstring $1^{(2/3)n}$ wird an der Stelle $j+1$ die Eins zu einer Null (durch $(2^{(2/3)n} - 1) - 2l$) und durch das Addieren von Eins werden die ersten $j+1$ Einsen zur Null und der Übertrag bleibt an der Stelle $j+1$ stehen. Das bedeutet auch, dass ab der Stelle $j+2$ das Komplement von $[l]_{j+1}^{n/3-1}$ steht. Aus $|x| = 2^{n-1} + l \cdot 2^{n/3}$ folgt dann Aussage (a)

$$[2^{(2/3)n} - 2l]_{j+2}^{(2/3)n-1} = z_{j+2}^{(2/3)n-1} = \overline{x_{n/3+j+1}^{n-2}} \qquad \text{(vgl. Abb. 4.5).}$$

Nach unseren Voraussetzungen gilt

$$y_{n/3}^{n/3+j} = 0^{j+1} \text{ und } y_{n/3+j+1} = 1$$

und somit $z_0^{j+1} = y_{n/3}^{n/3+j+1}$ (vgl. Abb. 4.5). Damit können wir nun $y_{n/3}^{n-1}$ und z vergleichen, indem wir uns $y_{n/3+j+2}^{n-1}$ und die entsprechenden Bits von x anschauen. Denn nach Aussage (a) gilt

$$\overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2} \Leftrightarrow \overline{x_{n/3+j+1}^{n-2}} = z_{j+2}^{(2/3)n-1} < y_{n/3+j+2}^{n-1}$$

und damit ist $|y_{n/3}^{n-1}| > 2^{(2/3)n} - 2l$. Aus den Vorüberlegungen (Fall 1) gilt dann $MUL_{2n-1,n}(x,y) = 1$, womit wir Aussage (b) gezeigt haben. Analog zeigen wir Aussage (c), denn aus $\overline{y_{n/3+j+2}^{n-1}} > x_{n/3+j+1}^{n-2}$ folgt, dass $|y_{n/3}^{n-1}| < 2^{(2/3)n} - 2l$ ist und somit $MUL_{2n-1,n}(x,y) = 0$ (Fall 2). Aus der zusätzlichen Voraussetzung

$$\overline{y_{n/3+j+2}^{n-1}} = x_{n/3+j+1}^{n-2}$$

der Aussage (d) folgt $z = y_{n/3}^{n-1}$ (vgl. Abb. 4.5). Also ist $|y_{n/3}^{n-1}| = 2^{(2/3)n} - 2l$ und damit gilt nach (4.3)

$$MUL_{2n-1,n}(x,y) = 1 \Leftrightarrow |y_0^{n/3-1}| \geq \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil.$$

□

Mit diesem Lemma können wir einen Spezialfall betrachten, bei dem die Berechnung von $MUL_{2n-1,n}$ nur von $x_{n/3}^{n-1}$ und $y_{n/3}^{n-1}$ abhängt.

Korollar 4.3.4. *Sei $n \in \mathbb{N}$, $x,y \in \{0,1\}^n$ mit $|x| = 2^{n-1} + l \cdot 2^{n/3}$, $1 \leq l < 2^{n/3-1}$ und $0 \leq j < (n/3) - 1$ der kleinste Offset mit $x_{n/3+j} = 1$. Weiter sei $y_0^{n/3-1} = 0^{n/3}$, $y_{n/3}^{n/3+j} = 0^{j+1}$ und $y_{n/3+j+1} = 1$. Dann gilt $MUL_{2n-1,n}(x,y) = 1$ genau dann, wenn $\overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2}$.*

Beweis. Nach Lemma 4.3.3 gilt $\overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2} \Rightarrow MUL_{2n-1,n}(x,y) = 1$. D.h., es bleibt die „⇐“ Richtung zu zeigen.

Angenommen, es gilt $MUL_{2n-1,n}(x,y) = 1$. Nach unseren Voraussetzungen $y_{n/3}^{n/3+j} = 0^{j+1}$ und $y_{n/3+j+1} = 1$ gilt wieder $y_{n/3}^{n/3+j+1} = [2^{(2/3)n} - 2l]_0^{j+1}$. Wir müssen zwei mögliche Fälle betrachten:

1. Fall: $y_{n/3+j+2}^{n-1} > [2^{(2/3)n} - 2l]_{j+2}^{(2/3)n-1} \overset{\text{Lemma 4.3.3 (a)}}{\Leftrightarrow} \overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2}$

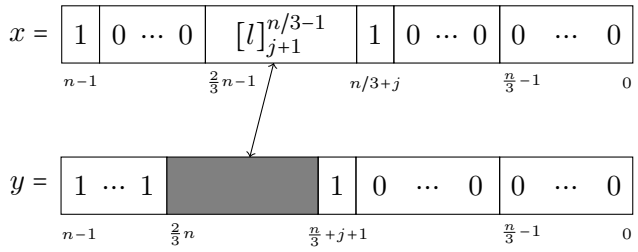

Abbildung 4.6: Entscheidende Bits zur Berechnung von $MUL_{2n-1,n}(x,y)$

2. Fall: $y_{n/3+j+2}^{n-1} = [2^{(2/3)n} - 2l]_{j+2}^{(2/3)n-1}$ und $|y_0^{n/3-1}| \geq \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil$

$\overset{\text{Lemma 4.3.3}(a)}{\Leftrightarrow}$ $\overline{y_{n/3+j+2}^{n-1}} = x_{n/3+j+1}^{n-2}$ und $|y_0^{n/3-1}| \geq \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil$

Angenommen, es würde weder Fall 1 noch Fall 2 eintreten, dann wäre entweder $\overline{y_{n/3+j+2}^{n-1}} > x_{n/3+j+1}^{n-2}$ oder $\overline{y_{n/3+j+2}^{n-1}} = x_{n/3+j+1}^{n-2}$ und $|y_0^{n/3-1}| < \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil$. Nach Lemma 4.3.3 (c) und (d) würde in beiden Fällen $MUL_{2n-1,n}(x,y) = 0$ folgen und damit unserer Annahme $MUL_{2n-1,n}(x,y) = 1$ widersprechen.

Angenommen es gilt $\overline{y_{n/3+j+2}^{n-1}} = x_{n/3+j+1}^{n-2}$. Nach Beobachtung 4.3.2 gilt $\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \rceil \geq 1$. Nach unseren Voraussetzungen gilt aber $y_0^{n/3-1} = 0^{n/3}$. Damit kann der Fall $\overline{y_{n/3+j+2}^{n-1}} = x_{n/3+j+1}^{n-2}$ nicht eintreten, denn nach Lemma 4.3.3 (d) müsste $MUL_{2n-1,n}(x,y) = 0$ gelten. Somit kann nur der Fall $\overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2}$ eintreten. Also gilt

$$MUL_{2n-1,n}(x,y) = 1 \Rightarrow \overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2}.$$

□

Wir kennen die Bitstruktur von x schon etwas genauer. Unter anderem wissen wir, dass die Bits an den Stellen $(2/3)n$ bis $n-2$ konstant auf Null gesetzt sind. Mit den Voraussetzungen aus dem letzten Korollar gilt

$$MUL_{2n-1,n}(x,y) = 1 \Leftrightarrow \overline{y_{n/3+j+2}^{n-1}} < x_{n/3+j+1}^{n-2}.$$

Das bedeutet, wenn eine Stelle $j' \in \{(2/3)n+1, \ldots, n-1\}$ mit $y_{j'} = 0$ existiert, dann gilt $\overline{y_{n/3+j+2}^{n-1}} > x_{n/3+j+1}^{n-2}$ und damit $MUL_{2n-1,n}(x,y) = 0$. Diesen einfachen Fall wollen wir

ausschließen, indem wir die Bits an den Stellen $(2/3)n+1,\ldots,n-1$ in y konstant auf Eins setzen und somit folgende Aussage bekommen:

$$MUL_{2n-1,n}(x,y) = 1 \Leftrightarrow \overline{y_{n/3+j+2}^{(2/3)n}} > x_{n/3+j+1}^{(2/3)n-1}$$

Nun haben wir eine Beziehung zwischen den nicht konstant gesetzten Bits in x und einem Teil der y-Eingabe (siehe Abb. 4.6). Im Hinblick auf unsere Reduktion haben wir nun aber noch folgendes Problem: Für die Komplexität des Kommunikationsproblems GT_n ist es entscheidend, dass kein Spieler zwei Bits mit der gleichen Wertigkeit als Eingabe zur Verfügung hat. In der Definition aus Kapitel 3 hat sogar jeder Spieler nur Bits derselben Eingabe, also ein Spieler alle Bits von der Eingabe $a \in \{0,1\}^n$ und der andere Spieler alle Bits der Eingabe $b \in \{0,1\}^n$. Wir werden später zeigen, dass das Kommunikationsproblem, bei dem die Aufteilung der Bits etwas offener ist, so dass kein Spieler die beiden Bits der gleichen Wertigkeit von a und b bekommt, mindestens so schwierig ist wie das Kommunikationsproblem mit der strikten Aufteilung. Wenn wir nun unsere Reduktion mit Hilfe von Korollar 4.3.4 konstruieren wollen, bedeutet das, dass wir für jede Variablenordnung π eine Partition L und R der Variablen finden müssen, so dass alle Variablen in L bzgl. π vor den Variablen aus R getestet werden, und möglichst viele Paare (x_i, y_{i+1}) für $n/3+j+1 \leq i < (2/3)n-1$ „getrennt“ sind. Genauer gesagt, brauchen wir $\Omega(n)$ Paare (x_i, y_{i+1}), für die entweder $x_i \in L$ und $y_{i+1} \in R$ oder $x_i \in R$ und $y_{i+1} \in L$ gilt. Diese Paare sind gerade die Bits mit der gleichen Wertigkeit in den Bitstrings $y_{n/3+j+2}^{n-1}$ und $x_{n/3+j+1}^{n-2}$. Wir können aber sehr leicht eine Variablenordnung angeben, bei der diese Trennung nicht möglich ist. Betrachten wir z.B. die Variablenordnung $(x_0, y_0, \ldots, x_{n-1}, y_{n-1})$. Jede gültige Partition trennt maximal ein Paar (x_i, y_{i+1}). Daher kommen wir in diesem Fall mit Korollar 4.3.4 nicht weiter. Falls für eine Variablenordnung π aber solch eine Trennung von $\Omega(n)$ Paaren möglich ist, liefert uns das Korollar eine geeignete Reduktion von $GT_{\Omega(n)}$ auf $MUL_{2n-1,n}^{\pi,p}$ mit p geeignet gewählt.

Das bedeutet, wir brauchen noch eine weitere Reduktionsmöglichkeit für den Fall, dass viele Paare (x_i, y_{i+1}) nicht getrennt werden können. Wenn dies jedoch der Fall ist, können wir in einer Rechteckreduktion diese Paare komplementär zueinander belegen. Um in einer Rechteckreduktion eine Variable komplementär zu einer nicht konstanten Variablen belegen zu können, müssen beide Variablen in der selben Partition liegen. Dies motiviert uns, die Aussage aus Lemma 4.3.3 (d) nochmal genauer anzuschauen.

$$\text{Falls } \overline{y_{n/3+j+2}^{n-1}} = x_{n/3+j+1}^{n-2} \text{ ist, dann gilt } MUL_{2n-1,n}(x,y) = 1$$
$$\text{genau dann, wenn } |y_0^{n/3-1}| \geq \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil.$$

Das bedeutet, wenn wir in diesem Fall auch eine geeignete Rechteckreduktion auf GT konstruieren können, können wir für jede Variablenordnung mit Hilfe einer Fallunterscheidung bzgl. der Anzahl der „getrennten“ (x_i, y_{i+1}) Paare eine Rechteckreduktion von GT auf $MUL_{2n-1,n}$ angeben.

Wie wir anfangs erwähnt haben, wollen wir zeigen, dass wir bestimmte Bits von $y_0^{n/3-1}$ so konstant setzen können, dass die „freien“ Bits genau dann größer oder gleich einem Teil der x-Eingabe sind, wenn $MUL_{2n-1,n}(x,y) = 1$ ist. Dazu müssen wir uns die Bedingung $|y_0^{n/3-1}| \geq \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil$ genauer anschauen.

Wie wir in Beobachtung 4.3.2 festgestellt haben, ist der Einfluss von $\frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}}$ auf den gesamten Term gering. D.h., der ausschlaggebende Term ist $l^2 \cdot 2^{-n/3+2}$. Wir wollen l geschickt wählen, um die Struktur von l^2 ausnutzen zu können. Wie in [Bol10] wählen wir dazu l von der Form $u \cdot 2^m + w$ mit $w < 2^m$, $u < 2^{(7/8)m+2}$ und $m = (8/45)n - 24/15$. Dies führt dazu, dass l^2 nun gleich

$$u^2 \cdot 2^{2m} + uw \cdot 2^{m+1} + w^2$$

ist.

Angenommen, wir setzen $u = 2^d$ konstant. Wenn wir uns nun die Bitstruktur von $[l^2]$ bzw. die Summe von den Bitstrings $[w^2]$, $[uw \cdot 2^{m+1}]$ und $[u^2 \cdot 2^{2m}]$ näher anschauen, sehen wir, dass $[u^2 \cdot 2^{2m}]$ nur eine Eins an der Stelle $2m + 2d$ in $[l^2]$ erzeugt und nur die Stellen $m+d+1, \ldots, 2m+d$ von möglichen Überträgen durch die Summe $[w^2] + [uw \cdot 2^{m+1}]$ betroffen sein können (siehe Abb. 4.7). Wenn wir w so wählen können, dass bei der Summe von $[w^2]$ und $[uw \cdot 2^{m+1}]$ ab Position $2m+1$ kein Übertrag erzeugt oder von vorherigen Überträgen weitergeleitet werden kann, dann gilt $[l^2]_{2m+2}^{2m+d} = [w]_{m-d+1}^{m-1}$.

Das nächste Lemma dient als Grundlage, damit wir zeigen können, dass $[w]_{m-d+1}^{m-1}$ auch in $[l^2 \cdot 2^{-n/3+2}]$ wiedergefunden werden kann, und wie wir die niedrigstwertigen Bits von y setzen müssen, damit dieser Teil größer oder gleich den niedrigstwertigen Bits von $\left[\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil\right]$ ist.

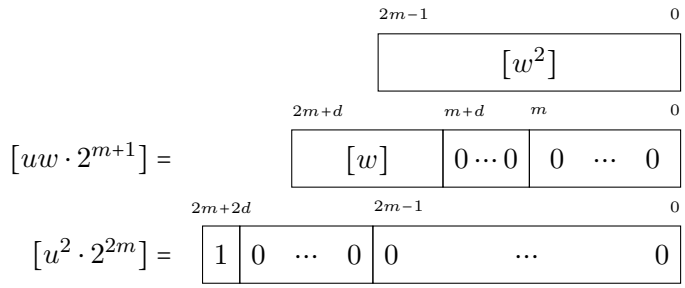

Abbildung 4.7: Zusammensetzung von $l^2 = u^2 \cdot 2^{2m} + uw \cdot 2^{m+1} + w^2$ mit $u = 2^d$

Lemma 4.3.5. *Sei $n \in \mathbb{N}$, $m = (8/45)n - 24/15$ und $1 \leq d < (7/8)m + 2$. Weiter sei $u = 2^d$, $W' := \{w \in \{0,1\}^m \mid w_0^{m-d} = 0^{m-d}1\}$ und $W = \{|w| \mid w \in W'\}$. Für alle $w, w', w'' \in W$ mit $w' < w''$ gilt*

(a) $(u^2 \cdot 2^{2m} + uw' \cdot 2^{m+1}) \operatorname{div} 2^{2m+2} < (u^2 \cdot 2^{2m} + uw'' \cdot 2^{m+1}) \operatorname{div} 2^{2m+2}$ *und*

(b) $(u^2 \cdot 2^{2m} + uw \cdot 2^{m+1} + w^2) \bmod 2^{2m+2} < 2^{2m+1}$.

Beweis. Für die Aussage (a) beobachten wir, dass die $m-d+1$ niedrigstwertigen Bits für alle Elemente aus W' konstant sind. D.h., das erste Bit, das sich im Produkt $[uw' \cdot 2^{m+1}] = [w' \cdot 2^{m+1+d}]$ vom Produkt $[w'' \cdot 2^{m+1+d}]$ unterscheiden kann, ist an der Stelle $2m+2$. Durch $uw' \operatorname{div} 2^{2m+2}$ werden nur die Bits an den Stellen $0, \ldots, 2m+1$ abgeschnitten. Daher folgt aus $w' < w''$ auch die Aussage (a).

Für die Aussage (b) sehen wir zuerst, dass $u^2 \cdot 2^{2m} = 2^{2m+2d} = 2^{2m+2} \cdot 2^{2d-2}$ ist und somit $(u^2 \cdot 2^{2m}) \bmod 2^{2m+2} = 0$ gilt. D.h., dieser Term spielt $\bmod 2^{2m+2}$ keine Rolle. Wir haben w so gewählt, dass in $[w]$, insbesondere an den Stellen $m-d$ und $m-d-1$, eine Null steht, d.h., in $[uw \cdot 2^{m+1}]$ steht an der Position $2m+1$ und $2m$ auch jeweils eine Null. Damit die Addition von $[w^2]$ nun einen Übertrag bis zu der Stelle $2m+1$ erzeugen kann, muss unter anderem an Position $2m$ in $[w^2]$ eine Eins stehen, d.h. $w^2 \geq 2^{2m}$. Dies ist aber ein Widerspruch zu unserer Wahl $w < 2^m$. Also kann in der Summe $[uw \cdot 2^{m+1}] + [w^2]$ keine Eins an Position $2m+1$ stehen und somit gilt auch Aussage (b).

□

Die Idee ist es nun, Bits von $y_0^{n/3-1}$ konstant zu setzen, so dass $|y_0^{2m-n/3+3}| > uw \cdot 2^{m+1} + w^2 \bmod 2^{n/3+2}$ und $|y_{2m-n/3+3+d}^{n/3-1}| = 2^{2d+2m} \operatorname{div} 2^{2m+d+1} = 2^{d-1}$ gilt. Aus dem letzten Lemma folgt dann, dass die $2m - n/3 + 4$ niedrigstwertigen Bits von y größer oder gleich

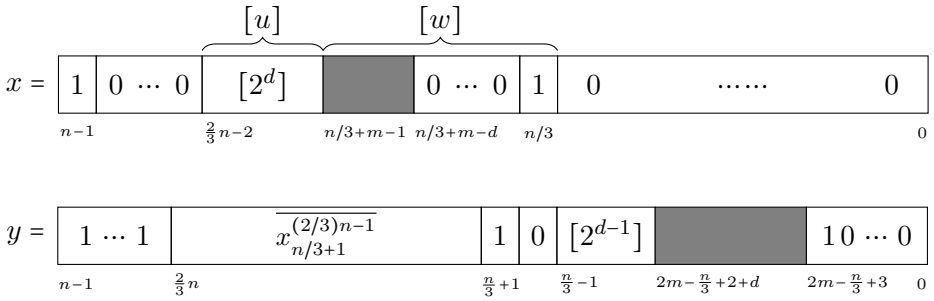

Abbildung 4.8: Zusammensetzung von x und y für die Reduktion von $\overline{GT}$ auf $MUL_{2n-1,n}$ (Bemerkung: Es gilt $n/3 - 1 + m + (7/8)m + 2 = (2/3)n - 2$)

$\left[\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil\right]_0^{2m-n/3+3}$ sind und die höchstwertigen Bits von $y_0^{n/3-1}$ mit den Bits von $\left[\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil\right]$ übereinstimmen. Dies liefert uns die Reduktion von $\overline{GT}$ auf $MUL_{2n-1,n}$ (siehe Abb. 4.8 für die Struktur von x und y).

Korollar 4.3.6. *Seien $n \in \mathbb{N}$, $m = (8/45)n - 24/15$ und $1 \le d < (7/8)m + 2$. Weiter seien $u = 2^d$, $W' := \{w \in \{0,1\}^m \mid w_0^{m-d} = 0^{m-d}1\}$, $W = \{|w| \mid w \in W'\}$ und $w \in W$ beliebig. Seien $x = 2^{n-1} + l \cdot 2^{n/3}$ mit $l = u \cdot 2^m + w$ und $y \in \{0,1\}^n$ mit $y_{n/3} = 0$, $y_{n/3+1} = 1$, $\overline{y_{n/3+2}^{n-1}} = x_{n/3+1}^{n-2}$, $|y_0^{2m-n/3+3}| = 2^{2m-n/3+3}$ und $|y_{2m-n/3+3+d}^{n/3-1}| = 2^{d-1}$. Dann gilt $MUL_{2n-1,n}(x,y) = 1$ genau dann, wenn $y_{2m-n/3+4}^{2m-n/3+2+d} \ge [w]_{m-d+1}^{m-1}$ gilt.*

Beweis. Mit unseren Voraussetzungen gilt nach Lemma 4.3.3, dass $MUL_{2n-1,n}(x,y) = 1$ ist genau dann, wenn $|y_0^{n/3-1}| \ge \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil$. Es gilt

$$l = u \cdot 2^m + w < (2^{(7/8)m+2} - 1) \cdot 2^m + 2^m = 2^{(15/8)m+2} = 2^{n/3-3+2} = 2^{n/3-1}. \tag{4.4}$$

Nach Beobachtung 4.3.2 folgt damit $\frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} < 1$. Damit haben wir

$$\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1}+l\cdot 2^{n/3}} \right\rceil = \begin{cases} l^2 \operatorname{div} 2^{n/3-2} & \text{oder} \\ l^2 \operatorname{div} 2^{n/3-2} + 1. \end{cases} \tag{4.5}$$

Nach Lemma 4.3.5 (b) wissen wir, dass $[l^2]_{2m+1} = 0$ ist. Es gilt

$$2m + 1 = (16/45)n - 33/15 \overset{n>9}{>} (15/45)n - 30/15 = n/3 - 2$$

und somit gilt auch $[l^2 \operatorname{div} 2^{n/3-2}]_{2m-n/3+3} = 0$, wenn n groß genug ist (Dies ist kein Problem, da n schon sehr groß sein muss, damit die Indizes ganze Zahlen sein können). Aus (4.5) folgt damit

$$y_0^{2m-n/3+3} \geq \left[\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil\right]_0^{2m-n/3+3}. \tag{4.6}$$

Aus $[l^2 \operatorname{div} 2^{n/3-2}]_{2m-n/3+3} = 0$ und (4.5) folgt weiter, dass

$$\left[\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil\right]_{2m-n/3+4}^{n/3-1} = [l^2 \operatorname{div} 2^{n/3-2}]_{2m-n/3+4}^{n/3-1} \tag{4.7}$$

gilt. Insbesondere haben wir

$$\begin{aligned} \left[\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil\right]_{2m-n/3+3+d}^{n/3-1} &= [l^2 \operatorname{div} 2^{n/3-2}]_{2m-n/3+3+d}^{n/3-1} \\ &= [l^2]_{2m+d+1}^{(2/3)n+1} = [u^2 \cdot 2^m]_{2m+d+1}^{(2/3)n+1} \\ &= [2^{d-1}] = y_{2m-n/3+3+d}^{n/3-1}. \end{aligned} \tag{4.8}$$

Angenommen, es gilt $y_{2m-n/3+4}^{2m-n/3+2+d} \geq [w]_{m-d+1}^{m-1}$. Nach (4.7) gilt

$$\left[\left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil\right]_{2m-n/3-1}^{2m-n/3+2+d} = [l^2 \operatorname{div} 2^{n/3-2}]_{2m-n/3-1}^{2m-n/3+2+d} = [w]_{m-d+1}^{m-1}.$$

Zusammen mit (4.6) und (4.8) erhalten wir

$$|y_0^{n/3-1}| \geq \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil$$

und damit gilt dann $MUL_{2n-1,n}(x, y) = 1$.

Angenommen, wir haben $y_{2m-n/3+4}^{2m-n/3+2+d} < [w]_{m-d+1}^{m-1}$. Sei $w' \in W$ mit $[w']_{m-d+1}^{m-1} = y_{2m-n/3+4}^{2m-n/3+2+d}$. Da $[w]$ und $[w']$ an den Stellen $0, \ldots, m-d$ identisch sind, gilt $w' < w$. Da $2m+1 > n/3-2$ ist, gilt die Aussage aus Lemma 4.3.5 (a) auch für $\operatorname{div} 2^{n/3-2}$, d.h.

$$(u^2 \cdot 2^m + uw' \cdot 2^{m+1}) \operatorname{div} 2^{n/3-2} < (u^2 \cdot 2^m + uw \cdot 2^{m+1}) \operatorname{div} 2^{n/3-2}.$$

Daraus folgt, dass $y_{2m-n/3+4}^{n/3-1} < [l^2 \operatorname{div} 2^{n/3-2}]_{2m-n/3+4}^{n/3-1}$ und damit auch

$$|y_0^{n/3-1}| < \left\lceil l^2 \cdot 2^{-n/3+2} - \frac{4l^3}{2^{n-1} + l \cdot 2^{n/3}} \right\rceil.$$

Somit gilt $MUL_{2n-1,n}(x,y) = 0$. Insgesamt gilt also

$$MUL_{2n-1,n}(x,y) = 1 \Leftrightarrow y_{2m-n/3+4}^{2m-n/3+2+d} \geq [w]_{m-d+1}^{m-1}.$$

□

Wir können mit Hilfe des Parameters d die $[w]$-Bits innerhalb von $y_0^{n/3-1}$ verschieben. Daher haben wir die Hoffnung, dass wir für jede Variablenordnung ein geeignetes d finden, so dass viele entsprechende Paare (d.h. Bits der gleichen Wertigkeit in $y_{2m-n/3+4}^{2m-n/3+2+d}$ und $[w]_{m-d+1}^{m-1}$) getrennt werden können. Das folgende Lemma liefert uns diese Eigenschaft (siehe [Bol10] und einen ähnlichen Beweis in [Bry91]).

Lemma 4.3.7. *Seien $n \in \mathbb{N}$, π eine Variablenordnung auf $X = \{x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}\}$ und $m = (8/45)n - 24/15$. Definiere $w_0 := x_{n/3}, \ldots, w_{m-1} = x_{n/3+m-1}$. Für die Variablen $S := \{w_{m/2}, \ldots, w_{m-1}, y_{2m+4-n/3}, \ldots, y_{(5/2)m+3-n/3}\}$ sei L die Menge der ersten $|L|$ Variablen bzgl. π, die $m/2$ Variablen aus S enthält. Dann existiert ein Parameter $2 \leq d \leq m$, so dass es mindestens $m/8$ Paare $(w_i, y_{m+3+i+d-n/3})$ gibt, für die genau einer der jeweiligen Variablen in L liegt.*

Beweis. Sei k die Anzahl an y-Variablen in der Menge L. D.h., L enthält $m/2 - k$ w-Variablen und k y-Variablen. Wir definieren die Mengen

$$S_d := \{(w_i, y_{m+3+i+d-n/3}) \mid \quad m/2 \leq i \leq m-1 \text{ und } (w_i, y_{m+3+i+d-n/3}) \in (L \times (X \backslash L)) \cup ((X \backslash L) \times L)\}$$

mit $2 \leq d \leq m$. Jedes mögliche Paar $(w_i, y_{2m+4-n/3+j})$ mit $m/2 \leq i \leq m-1$ und $0 \leq j \leq m/2 - 1$ aus $\{w_{m/2}, \ldots, w_{m-1}\} \times \{y_{2m+4-n/3}, \ldots, y_{(5/2)m+3-n/3}\}$ ist darstellbar als $(w_i, y_{m+3+i+d-n/3})$ mit $d = m + 1 + j - i$. Außerdem gilt

$$2 = m + 1 + 0 - m + 1 \leq d \leq m + 1 + m/2 - 1 - m/2 = m.$$

D.h., jedes durch L getrennte Paar muss in mindestens einer S_d Menge auftauchen. Es gilt also

$$\sum_{d=2}^{m} |S_d| \geq (m/2 - k)^2 + k^2.$$

Wir definieren die Funktion $f : [0, \frac{m}{2}] \to \mathbb{N}$ mit $f(k) := (m/2 - k)^2 + k^2$. Die Ableitung lautet

$$f'(k) = -2(m/2 - k) + 2k = 4k - m.$$

Also haben wir eine einzige Nullstelle bei $m/4$ und wegen $f''(k) = 4 > 0$ liegt also das Minimum der Funktion bei $m/4$. D.h., es gilt

$$\sum_{d=2}^{m} |S_d| \geq (m/2 - m/4)^2 + (m/4)^2 = m^2/8.$$

Nach dem Schubfachprinzip muss es also ein d geben mit

$$|S_d| \geq \frac{m^2}{8(m-1)} \geq m/8.$$

□

Wie wir bereits angedeutet haben und es auch in Lemma 4.3.7 zu sehen ist, brauchen wir eine modifizierte Variante von dem Kommunikationsproblem GT, um unsere untere Schranke zu beweisen. Wir trennen die Eingabe nicht strikt, so dass Alice die Bits zu der Eingabe a und Bob die Bits zu der Eingabe b bekommt, sondern lockern diese Verteilung etwas und erlauben für jede Position $0 \leq i \leq n-1$, dass entweder Alice a_i und Bob b_i erhält oder umgekehrt. D.h., keiner von beiden hat für ein $0 \leq i \leq n-1$ beide Bits a_i und b_i als Eingabe.

Definition 4.3.8. *Sei $n \in \mathbb{N}$. $GT'_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ ist das Kommunikationsproblem für $GT_n(a,b)$ mit der Modifizierung, dass für jedes $0 \leq i \leq n-1$ weder Alice noch Bob die Bits a_i und b_i gemeinsam kennen.*

Die Kommunikationskomplexität von GT kann auf GT' übertragen werden.

Lemma 4.3.9. *Sei $n \in \mathbb{N}$. Es gilt $GT_n \leq_{rec} GT'_n$.*

Beweis. Vorab definieren wir eine Indexmenge, die die Positionen zu den „richtigen“ Parteien in der Eingabe (a', b') für GT' zuordnet. Sei dazu

$$I := \{i \in \mathbb{N} \mid \text{ Alice kennt } a'_i \text{ und Bob } b'_i \text{ in } GT'\}$$

die Menge der Indizes, die „richtig“ verteilt sind (d.h. so wie in dem Kommunikationsproblem GT). Sei nun (a,b) eine Eingabe für GT_n. Wir konstruieren eine Eingabe (a', b') für GT'_n folgendermaßen:

- Für die Positionen $i \in I$ setze $a'_i = a_i$ bzw. $b'_i = b_i$.

- Für die Positionen $i \notin I$ setze $b'_i = 1 - a_i$ bzw. $a'_i = 1 - b_i$.

Sei j der Index des ersten höchstwertigen Bits mit $a_j \neq b_j$. Nach unserer Konstruktion gilt $a'_i = b'_i$ für $i > j$ (falls es den Index nicht gibt, gilt $a = b$ und $a' = b'$). Falls die Position j „richtig“ verteilt ist, d.h. $j \in I$, gilt $a > b$ genau dann, wenn $a' > b'$. Falls die Position j „falsch“ verteilt ist (d.h. $j \notin I$), gilt

$$GT'_n(a', b') = 1 \Leftrightarrow 1 - b_j = a'_j > b'_j = 1 - a_j \Leftrightarrow a_j > b_j \Leftrightarrow GT_n(a, b) = 1.$$

Und damit gilt $GT(a, b) = GT'(a', b')$. □

Analog zu dieser Aussage können wir beweisen, dass diese Reduktion auch für $\overline{GT}$ und $\overline{GT'}$ funktioniert.

Korollar 4.3.10. *Sei $n \in \mathbb{N}$. Es gilt $\overline{GT_n} \leq_{rec} \overline{GT'_n}$.*

Beweis. Sei (a, b) die Eingabe für $\overline{GT}_n$. Wir setzen die Eingabe (a', b') für $\overline{GT'_n}$ analog zu Lemma 4.3.9. Falls $a = b$ ist, gilt auch $a' = b'$. Falls $a < b$ ist, sei j der höchstwertige Index mit $a_j < b_j$. Falls die Position richtig verteilt ist, haben wir auch $a'_j = a_j < b_j = b'_j$ und sonst $a'_j = 1 - b_j < 1 - a_j = b'_j$. Analog gilt auch $a' \leq b' \Rightarrow a \leq b$. Also gilt $a \leq b$ genau dann, wenn $a' \leq b'$. □

Um die Aussagen aus Kapitel 3 zu verwenden, müssen wir noch zeigen, dass die untere Schranke für die Kommunikationskomplexität von GT auch für $\overline{GT}$ gilt.

Lemma 4.3.11. *Sei $n \in \mathbb{N}$. Es gilt $GT_n \leq_{rec} \overline{GT_n}$.*

Beweis. Sei $a, b \in \{0,1\}^n$ eine Eingabe für GT_n. Für $c, d \in \{0,1\}^n$ setze $c = \overline{a}$ und $d = \overline{b} - 1$. Dann gilt

$$a > b \Leftrightarrow |a| > |b| \Leftrightarrow |a| \geq |b| + 1 \Leftrightarrow (2^n - 1) - |b| - 1 \geq (2^n - 1) - |a| \Leftrightarrow c = \overline{a} \leq \overline{b} - 1 = d.$$

Also folgt $GT(a, b) = \overline{GT}(c, d)$. □

4.3.2 Reduktion von GT auf $MUL_{2n-1,n}$

Nun müssen wir noch alles richtig zusammensetzen, um unsere Reduktion zu konstruieren. Die Variablen aus Lemma 4.3.7, die getrennt werden können, haben in unserer Reduktion eine entscheidende Bedeutung. Daher machen wir folgende Definition.

Definition 4.3.12. *Sei* $X = \{x_0, \dots, x_{n-1}, y_0, \dots, y_{n-1}\}$ *die Menge von Variablen und* $S := \{w_{m/2}, \dots, w_{m-1}, y_{2m+4-n/3}, \dots, y_{(5/2)m+3-n/3}\}$ *mit* $w_{m/2} := x_{n/3+m/2}, \dots, w_{m-1} := x_{n/3+m-1}$. *Für eine Variablenordnung* π *auf* X *sei* $L := L(\pi)$ *die Menge der ersten* $|L|$ *Variablen bzgl.* π, *die genau* $m/2$ *Variablen aus* S *enthält. Wir definieren* $d := d(\pi)$ *als den Distanzparameter aus Lemma 4.3.7 und die Indexmenge* $I := I(\pi)$ *mit*

$$i \in I \Leftrightarrow (w_i, y_{m+3+i+d-n/3}) \in (L \times (X \backslash L)) \cup ((X \backslash L) \times L).$$

Für $i \in I$ *bezeichnen wir* $x_{n/3+i}$ *als* freie x*-Variable und* $y_{n/3+i+1}$ *und* $y_{m+3+i+d-n/3}$ *als* freie y*-Variablen.*

Die Idee der Reduktion ist es, anhand der Paare $(x_{n/3+i}, y_{n/3+i+1})$ mit $i \in I$ eine Fallunterscheidung zu machen.

1. Fall: Falls es viele Paare $(x_{n/3+i}, y_{n/3+i+1})$ gibt, so dass beide Variablen in L oder beide nicht in L sind, dann können wir diese Variablen komplementär zueinander setzen und so $\overline{GT'}$ auf $MUL_{2n-1,n}$ (Korollar 4.3.6) reduzieren. Lemma 4.3.7 sichert uns, dass die entscheidenden Variablen gut verteilt sind.

2. Fall: Sonst gibt es viele Paare dessen Variablen getrennt sind und somit haben wir mit Korollar 4.3.4 eine Reduktion von GT' auf $MUL_{2n-1,n}$. Hier liefert uns schon die Voraussetzung die gute Verteilung der Variablen.

Im Folgenden seien $w := w_0^{m-1} := x_{n/3}^{n/3+m-1}$ und $u := u_0^{(7/8)m+1} = x_{n/3+m}^{n/3+m+(7/8)m+1}$ (Anmerkung: $n/3 + m + (7/8)m + 1 = (2/3)n - 2$ für $m = (8/45)n - 24/15$). Nun können wir unsere Reduktion mit Hilfe der Fallunterscheidung beweisen.

Theorem 4.3.13. *Seien* $n \in \mathbb{N}$, $m = (8/45)n - 24/15$, π *eine Variablenordnung auf* $X = \{x_0, \dots, x_{n-1}, y_0, \dots, y_{n-1}\}$ *und* L *die Menge der ersten* $|L|$ *Variablen bzgl.* π, *die* $m/2$ *Variablen aus* $S := \{w_{m/2}, \dots, w_{m-1}, y_{2m+4-n/3}, \dots, y_{(5/2)m+3-n/3}\}$ *enthält.*

1. *Falls es mindestens* $m/16$ *Paare* $(x_{n/3+i}, y_{n/3+i+1})$ *mit* $i \in I$ *gibt, so dass entweder beide Variablen in* L *oder beide nicht in* L *liegen, dann gilt*

$$\overline{GT'_{m_1}} \leq_{rec} MUL_{2n-1,n}^{\pi,|L|} \text{ mit } m_1 \geq m/16.$$

2. Falls es weniger als $m/16$ solcher Paare gibt, dann gilt

$$GT'_{m_2} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n} \text{ mit } m_2 \geq m/16.$$

Beweis. Angenommen, wir haben mindestens $m/16$ Paare $(x_{n/3+i}, y_{n/3+i+1})$ mit $i \in I$, so dass entweder beide Variablen in L oder beide nicht in L liegen. Wir verkleinern I auf die Indizes, die zu solch einem Paar gehören. Das bedeutet auch, dass wir die Menge der freien Variablen entsprechend verkleinern.

Zuerst zeigen wir, dass unser Distanzparameter d durch $(7/8)m+1$ nach oben beschränkt ist. Diese Eigenschaft brauchen wir als Voraussetzung, um unsere vorherigen Aussagen anwenden zu können. Für $d = m$ kann nur ein Paar $(w_i, y_{m+3+i+d-n/3}) = (w_i, y_{2m+3+i-n/3})$ mit $i = m/2, \ldots, m-1$ erzeugt werden, so dass der Index der y-Variablen zwischen $2m + 4 - n/3$ und $(5/2)m + 3 - n/3$ liegt. D.h. für $d = m - (j-1)$ für $1 \leq j \leq m/2$ sind höchstens j Paare möglich. Laut Lemma 4.3.7 haben wir mindestens $m/8$ Paare, d.h.

$$d \leq m - m/8 + 1 = (7/8)m + 1.$$

Eine untere Schranke für den kleinsten Index in I ist $m - d + 1$, denn der kleinste Index für die y-Variable ist $2m + 4 - n/3$ und es gilt

$$m + 3 + i + d - n/3 = 2m + 4 - n/3 \Leftrightarrow i = m - d + 1.$$

Sei $m_1 = |I| \geq m/16$ und (a', b') eine Eingabe für $\overline{GT'_{m_1}}$. Wir setzen unsere Eingabe (x, y) für $MUL_{2n-1,n}$ folgendermaßen:

- Wir setzen $x_{n-1} = 1$ und $y_{n-1} = 1$.
  Auswirkung: Beide Eingaben sind groß genug und wir haben nicht die konstante Nullfunktion.

- Die Variablen $w_0 = x_{n/3}$ und $u_d = x_{n/3+m+d}$ werden auf 1 gesetzt. Abgesehen von den freien x-Variablen werden alle weiteren Bits auf 0 gesetzt. Die freien x-Variablen werden entsprechend den a' Bits gesetzt.
  Auswirkung: Die x Eingabe hat nun die Form $2^{n-1} + l \cdot 2^{n/3}$ mit $l = |u| \cdot 2^m + |w|$ und $u = 2^d$. Wie wir gezeigt haben, gilt $i \geq m - d + 1$ für alle $i \in I$. Daraus folgt, dass auf jeden Fall $w_0^{m-d} = 0^{m-d}1$ gilt.

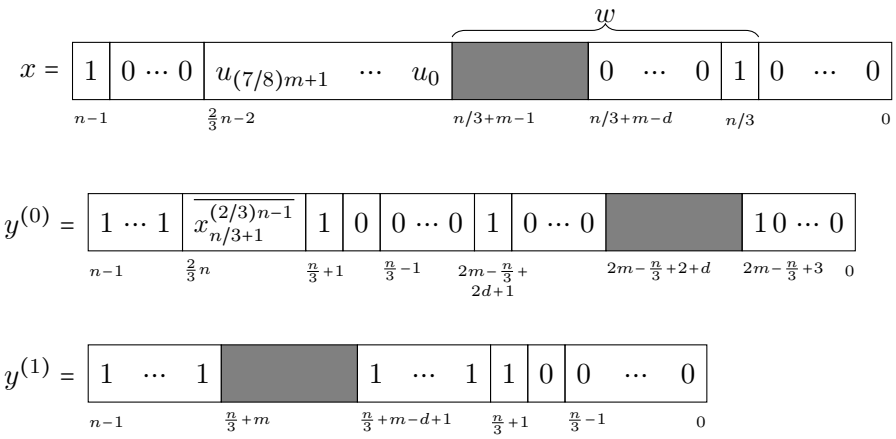

Abbildung 4.9: Veranschaulichung der Eingabe für $MUL_{2n-1,n}$ für die Reduktion von $\overline{GT'}$ ($y^{(0)}$ und $u = [2^d]$) bzw. GT' ($y^{(1)}$ und $u = 0^{(7/8)m+2}$). Die grauen Bereiche beinhalten u.a. die entscheidenden freien Variablen.

- Wir setzen $y_{n/3+1}$ auf 1 und $y_{n/3}$ und $y_{n/3+m+d+1}$ auf 0. Abgesehen von den freien $y_{n/3+i+1}$-Variablen werden alle weiteren Variablen aus $\{y_{n/3}, \ldots, y_{n-1}\}$ auf 1 gesetzt. Die freien $y_{n/3+i+1}$-Variablen werden entgegengesetzt ihrer korrespondierenden freien $x_{n/3+i}$-Variable belegt.
  Auswirkung: Es gilt nun $y_{n/3} = 0$, $y_{n/3+1} = 1$ und $\overline{y^{n-1}_{n/3+2}} = x^{n-2}_{n/3+1}$.

- Die Variablen $y_{2m-n/3+3}$ und $y_{2m-n/3+2+2d}$ werden auf 1 gesetzt. Abgesehen von den freien $y_{m+3+i+d-n/3}$-Variablen werden die weiteren Bits aus $\{y_0, \ldots, y_{n/3-1}\}$ auf 0 gesetzt. Die freien $y_{m+3+i+d-n/3}$-Variablen werden entsprechend den b'-Bits gesetzt.
  Auswirkung: Es gilt nun $|y_0^{2m-n/3+3}| = 2^{2m-n/3+3}$ und $|y^{n/3-1}_{2m-n/3+3+d}| = 2^{d-1}$.

Durch unsere Konstantsetzungen haben wir die Voraussetzungen für Korollar 4.3.6 erfüllt und damit gilt $MUL_{2n-1,n}(x, y) = 1$ genau dann, wenn $y^{2m-n/3+2+d}_{2m-n/3+4} \geq w^{m-1}_{m-d+1} = x^{n/3+m-1}_{n/3+m-d+1}$. Abgesehen von den freien Variablen in $\{y_{2m-n/3+4}, \ldots, y_{2m-n/3+2+d}\}$ und $\{x_{n/3+m-d+1}, \ldots, x_{n/3+m-1}\}$ sind die Bits auf 0 gesetzt. D.h., es sind nur die freien Variablen entscheidend. Sei $i \in I$ mit $i = m - d + 1 + j$ und $j \geq 0$. Dann ist das entsprechende Paar zu $x_{n/3+m-d+1+j}$ die Variable $y_{m+1+i+d-n/3+2} = y_{2m-n/3+4+j}$. D.h., die Wertigkeit in $y^{2m-n/3+2+d}_{2m-n/3+4}$ und $x^{n/3+m-1}_{n/3+m-d+1}$ dieser Paare stimmen überein. Da wir die freien x-Variablen

entsprechend a' und die freien y-Variablen entsprechend b' gesetzt haben, gilt also

$$y_{2m-n/3+4}^{2m-n/3+2+d} \geq w_{m-d+1}^{m-1} = x_{n/3+m-d+1}^{n/3+m-1} \Leftrightarrow a' \leq b' \Leftrightarrow \overline{GT'_{m_1}}(a', b') = 1.$$

Es bleibt noch zu zeigen, dass dies wirklich eine Rechteckreduktion ist. Dazu nutzen wir die Voraussetzung aus, dass die Variablenpaare $(x_{n/3+i}, y_{n/3+i+1})$ in der gleichen Partitionsmenge liegen. Denn wir müssen diese Variablen gegensätzlich belegen und $x_{n/3+i}$ hängt von der Eingabe a' ab. Das bedeutet, wenn beide Variablen in der Menge L sind, muss Alice das entsprechende Bit von a' als Eingabe von $\overline{GT'}$ bekommen und umgekehrt, wenn beide Variablen nicht in L sind, muss Bob das Bit von a' als Eingabe bekommen. Weiter wissen wir, dass die Paare $(x_{n/3+i}, y_{m+3+i+d-n/3})$ nach Lemma 4.3.7 getrennt liegen. Dies entspricht der Aufteilung in $\overline{GT'}$, dass weder Alice noch Bob zwei Bits der gleichen Wertigkeit kennen. Insgesamt haben wir damit eine korrekte Rechteckreduktion von $\overline{GT'_{m_1}}$ auf $MUL_{2n-1,n}^{\pi,|L|}$ mit $m_1 \geq m/16$.

Angenommen, wir haben nun weniger als $m/16$ solcher Paare, das bedeutet, wir haben mehr als $m/16$ Paare $(x_{n/3+i}, y_{n/3+i+1})$, so dass die Variablen weder beide in L noch beide nicht in L sind. Wir verkleinern die Indexmenge I wieder auf die Indizes dieser Paare (somit auch wieder die Menge der freien Variablen). Sei wieder $m_2 = |I| \geq m/16$. Für eine Eingabe (a', b') für GT'_{m_2} setzen wir die Eingabe (x, y) für $MUL_{2n-1,n}$ folgendermaßen:

- Wir setzen $x_{n-1} = 1$ und $y_{n-1} = 1$.
  Auswirkung: Beide Eingaben sind groß genug und wir haben nicht die konstante Nullfunktion.

- Wir setzen $y_0^{n/3-1} = 0^{n/3}$, $w_0 = x_{n/3} = 1$, $y_{n/3+1} = 1$ und $y_{n/3} = 0$.
  Auswirkung: Wir erfüllen nun die Voraussetzungen von Korollar 4.3.4.

- Abgesehen von den freien x-Variablen werden alle x-Bits auf 0 gesetzt. Die freien x-Variablen werden entsprechend den a'-Bits gesetzt. Abgesehen von den freien $y_{n/3+i+1}$-Variablen werden die noch nicht gesetzten y-Bits auf 1 gesetzt und die freien $y_{n/3+i+1}$-Variablen entsprechend $\overline{b'}$ gesetzt.
  Auswirkung: Nach Korollar 4.3.4 gilt $MUL_{2n-1,n}(x, y) = 1$ genau dann, wenn $\overline{y_{n/3+2}^{n-1}} < x_{n/3+1}^{n-2}$. Die nicht freien Variablen sind gegensätzlich gesetzt worden. D.h., es sind nur noch die freien Variablen entscheidend.

Wir haben nun $MUL_{2n-1,n}(x,y) = 1$ genau dann, wenn $\overline{\overline{b'}} = b' < a' \Leftrightarrow GT'_{m_2}(a',b') = 1$ gilt. Nach Voraussetzung liegen die Paare $(x_{n/3+i}, y_{n/3+i+1})$ getrennt und daher ist dies eine korrekte Rechteckreduktion von GT'_{m_2} auf $MUL^{\pi,|L|}_{2n-1,n}$ mit $m_2 \geq m/16$. □

In Kapitel 3 haben wir eine lineare untere Schranke für die Kommunikationskomplexität für GT bewiesen. Damit können wir nun unser Endresultat festhalten.

Theorem 4.3.14. *Die randomisierte OBDD-Größe für die Funktion $MUL_{2n-1,n}$ ist mindestens $2^{\Omega(n)}$.*

Beweis. Für alle Variablenordnungen können wir nach Lemma 4.3.7 einen geeigneten Parameter d, eine Variablenmenge L und eine Indexmenge I mit $|I| \geq m/8$ finden, so dass die Paare $(x_{n/3+i}, y_{m+1+i+d-n/3+2})$ mit $i \in I$ getrennt liegen, d.h. es liegt genau eine der beiden Variablen in L. Nach dem Schubfachprinzip muss es nun mindestens $m/16$ Paare $(x_{n/3+i}, y_{n/3+i+1})$ mit $i \in I$ und $m = (8/45)n - 24/15$ geben, so dass entweder

1. beide Variablen in L oder beide nicht in L liegen oder
2. in L genau eine der beiden Variablen liegt.

Nach Theorem 4.3.13 können wir für den ersten Fall $\overline{GT'_{m_1}} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n}$ und für den zweiten Fall $GT'_{m_2} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n}$ mit $m_1, m_2 \geq m/16$ zeigen. Damit haben wir:

1. Fall: $\overline{GT'_{m_1}} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n} \overset{\text{Korollar 4.3.10}}{\Rightarrow} \overline{GT_{m_1}} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n} \overset{\text{Lemma 4.3.11}}{\Rightarrow} GT_{m_1} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n}$
2. Fall: $GT'_{m_2} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n} \overset{\text{Lemma 4.3.9}}{\Rightarrow} GT_{m_2} \leq_{rec} MUL^{\pi,|L|}_{2n-1,n}$

Wegen $m_1, m_2 \geq m/16 = \Omega(n)$ und der linearen unteren Schranke der Kommunikationskomplexität von GT aus Theorem 3.4.1 gilt nach Lemma 2.3.4, dass alle randomisierten OBDDs für die Funktion $MUL_{2n-1,n}$ mindestens die Größe $2^{\Omega(n)}$ haben. □

4.4 Eine exponentielle untere Schranke für die Größe randomisierter OBDDs für das mittlere Bit der Multiplikation

Durch unsere untere Schranke für die Größe randomisierter OBDDs für das höchstwertige Bit der Multiplikation und unseren Überlegungen am Anfang des Kapitels ist es nun leicht, auch eine untere Schranke für die Größe randomisierter OBDDs für das mittlere Bit der Multiplikation zu beweisen.

Theorem 4.4.1. *Die randomisierte OBDD-Größe für die Funktion $MUL_{n-1,n}$ ist mindestens $2^{\Omega(n)}$.*

Beweis. Es gilt $MUL_{n-1,n} \leq_{rop} MUL_{n,n+1}$. D. h., es reicht, wenn wir uns den Fall anschauen, wenn n gerade ist. Wir setzen $n' = n/2$. Nach Lemma 4.2.1 (b) gilt $MUL_{2n'-1,n'} \leq_{rop} MUL_{2n'-1,2n'} = MUL_{n-1,n}$. Somit folgt aus Beobachtung 2.3.6 und Theorem 4.3.14, dass die randomisierte OBDD-Größe für $MUL_{n-1,n}$ mindestens $2^{\Omega(n')} = 2^{\Omega(n)}$ ist. □

5 Ausblick

Wir haben uns in der vorliegenden Arbeit die Berechnung des höchstwertigen Bits der Multiplikation für randomisierte OBDDs angeschaut. Wir haben gezeigt, dass Randomisierung bei OBDDs nicht weiterhilft und die randomisierte OBDD-Komplexität exponentiell sowohl für das höchstwertige als auch für das mittlere Bit der Multiplikation ist. Wir können unser Ergebnis leicht auf randomisierte k-OBDDs erweitern: Wir wissen, dass die randomisierte k-Runden Kommunikationskomplexität von GT_n durch $\Omega(n^{1/k}k^{-2})$ nach unten beschränkt ist. Damit können wir mit unserer Reduktion und Lemma 2.3.4 zeigen, dass ein randomisiertes k-OBDD mindestens $2^{\Omega(n^{1/k})}$ Knoten haben muss, wenn k konstant ist. Das bedeutet, wenn wir in einem randomisierten OBDD jede Variable mehr als einmal aber konstant oft abfragen dürfen, haben wir immer noch eine superpolynomielle untere Schranke.

Die Komplexität des höchstwertigen Bits der Multiplikation für allgemeinere BDD-Modelle ist weiterhin offen. Eine Erweiterung von OBDDs, bei der es erlaubt ist, mehrere Variablenordnungen zu benutzen, sind z.B. *baumgesteuerte* BDDs. Allgemeiner bezeichnen wir ein BDD als *graphgesteuertes BDD* (siehe auch [Weg00]), wenn auf jedem Pfad jede Variable höchstens einmal getestet wird (auch *Free Binary Decision Diagram* kurz FBDD genannt) und das BDD zusätzlich eine Graphordnung beachten muss. Dabei ist eine Graphordnung ein vollständiges FBDD, d.h. jede Variable kommt auf jedem Pfad genau einmal vor, und diese Graphordnung gibt bei Eingabe x durch den Pfad in dem vollständigen FBDD an, in welcher Ordnung das BDD die Variablen testen darf. Für ein baumgesteuertes BDD muss die Graphordnung ein Baum sein und darf nur polynomielle Größe haben. Für das mittlere Bit der Multiplikation wurde in [Bol00] bewiesen, dass die Größe von baumgesteuerten BDDs $2^{\Omega(n/\log n)}$ ist. Für diese untere Schranke wurde gezeigt, dass für alle Fixierungen von $O(\log n)$ Bits der Eingabe die OBDD-Komplexität exponentiell ist. Damit kann man dann beweisen, dass auch die baumgesteuerte BDD-

Komplexität exponentiell sein muss. Diese Technik funktioniert bei dem höchstwertigen Bit nicht direkt. Z.B. wenn die höchstwertigen Bits der Eingaben auf 0 gesetzt werden, ist auch die Funktion konstant 0. Die höherwertigen Bits haben einen größeren Einfluss auf das Ergebnis als die niedrigwertigen Bits. D.h., man braucht einen neuen Ansatz, um für baumgesteuerte BDDs eine untere Schranke für das höchstwertige Bit der Multiplikation zu zeigen.

Eine weitere Hürde ist es, ein besseres Verständnis für die Berechnung des höchstwertigen Bits der Multiplikation zu bekommen. Die Methoden, die hier für randomisierte OBDDs und auch für deterministische OBDDs verwendet werden, reichen nicht aus, um für baumgesteuerte BDDs, FBDDs oder noch allgemeinere BDD-Modelle gute untere Schranken zu zeigen.

Eine weitere spannende Frage ist, ob es ein BDD-Modell gibt, bei dem die Komplexität des mittleren Bits exponentiell ist aber die Komplexität des höchstwertigen Bits polynomiell. D.h., gibt es ein Modell, in dem die Berechnung des höchstwertigen Bits effizient möglich ist, während die Berechnung des mittleren Bit weiterhin schwierig bleibt?

Es bieten sich also noch viele offene und spannende Fragen bzgl. der BDD-Komplexität des höchstwertigen Bits der Multiplikation.

Literaturverzeichnis

[Abl03] Ablayev, F: A lower bound for integer multiplication on randomized ordered read-once branching programs. In: *Information and Computation* 186 (2003), Nr. 1, S. 78–89

[AK96] Ablayev, Farid M. ; Karpinski, Marek: On the power of randomized branching programs. In: *Proc. of ICALP*, 1996 (Lecture Notes in Computer Science vol. 1099), S. 348–356

[Bol00] Bollig, Beate: Restricted nondeterministic read-once branching programs and an exponential lower bound for integer multiplication. In: *Proc. of MFCS*, 2000, S. 222–231

[Bol10] Bollig, Beate: A larger lower bound on the OBDD complexity of the most significant bit of multiplication. In: *Proc. of LATIN*, 2010 (Lecture Notes in Computer Science vol. 6034), S. 255–266

[Bry86] Bryant, Randal E.: Graph-based algorithms for boolean function manipulation. In: *IEEE Trans. Computers* 35 (1986), S. 677–691

[Bry91] Bryant, Randal E.: On the complexity of VLSI implementations and graph representations of boolean functions with application to integer multiplication. In: *IEEE Trans. Computers* 40 (1991), Nr. 2, S. 205–213

[BW96] Bollig, Beate ; Wegener, Ingo: Read-once projections and formal circuit verification with binary decision diagrams. In: *Proc. of STACS*, 1996, S. 491–502

[BW01] Bollig, Beate ; Woelfel, Philipp: A read-once branching program lower bound of $\Omega(2^{n/4})$ for integer multiplication using universal hashing. In: *Proc. of STOC*, 2001, S. 419–424

[CT06] Cover, T.M. ; Thomas, J.A.: *Elements of Information Theory*. Wiley-Interscience, 2006. – ISBN 0–471–24195–4

[DKSS08] De, Anindya ; Kurur, Piyush P. ; Saha, Chandan ; Saptharishi, Ramprasad: Fast integer multiplication using modular arithmetic. In: *Proc. of STOC*, 2008, S. 499–506

[Für07] Fürer, Martin: Faster integer multiplication. In: *Proc. of STOC*, 2007, S. 57–66

[HMP+87] Hajnal, Andras ; Maass, Wolfgang ; Pudlak, Pavel ; Szegedy, Mario ; Turan, Gyorgy: Threshold circuits of bounded depth. In: *Proc. of FOCS*, 1987, S. 99–110

[KN97] Kushilevitz, Eyal ; Nisan, Noam: *Communication Complexity*. Cambridge University Press, 1997. – ISBN 0–521–56067–5

[KNTSZ01] Klauck, Hartmut ; Nayak, Ashwin ; Ta-Shma, Amnon ; Zuckerman, David: Interaction in quantum communication and the complexity of set disjointness. In: *Proc. of STOC*, 2001, S. 124–133

[MNSW98] Miltersen, Peter B. ; Nisan, Noam ; Safra, Shmuel ; Wigderson, Avi: On data structures and asymmetric communication complexity. In: *Journal of Computer and System Sciences* 57 (1998), Nr. 1, S. 37–49

[Neu28] Neumann, John V.: Zur Theorie der Gesellschaftsspiele. In: *Mathematische Annalen* 100 (1928), Nr. 1, S. 295–320

[Sau98] Sauerhoff, Martin: *Complexity Theoretical Results for Randomized Branching Programs*, Fachbereich Informatik, Universität Dortmund, Diss., 1998

[Saw06] Sawitzki, Daniel: Exponential lower bounds on the space complexity of OBDD-based graph algorithms. In: *Proc. of LATIN*, 2006, S. 781–792

[SR94] Siu, Kai-Yeung ; Roychowdhury, Vwani P.: On optimal depth threshold circuits for multiplication and related problems. In: *SIAM Journal on Discrete Mathematics* 7 (1994), S. 284–292

[SS71] Schönhage, Arnold ; Strassen, Volker: Schnelle Multiplikation großer Zahlen. In: *Computing* 7 (1971), Nr. 3-4, S. 281–292

[SV08] Sen, Pranab ; Venkatesh, S.: Lower bounds for predecessor searching in the cell probe model. In: *Journal of Computer and System Sciences* 74 (2008), Nr. 3, S. 364 – 385

[Weg87] Wegener, I.: *The complexity of Boolean functions.* B.G. Teubner, 1987 (Wiley-Teubner series in computer science). – ISBN 978–0471915553

[Weg00] Wegener, Ingo: *Branching programs and binary decision diagrams.* SIAM Monographs on Discrete Mathematics and Applications, 2000. – ISBN 0–89871–458–3

[Woe02] Woelfel, Philipp: On the complexity of integer multiplication in branching programs with multiple tests and in read-once branching programs with limited nondeterminism. In: *Proc. of CCC*, 2002, S. 80–89

[Woe05] Woelfel, Philipp: Bounds on the OBDD-size of integer multiplication via universal hashing. In: *Journal of Computer and System Sciences* 71 (2005), Nr. 4, S. 520–534

[WW05] Wegener, Ingo ; Woelfel, Philipp: New results on the complexity of the middle bit of multiplication. In: *Proc. of CCC*, 2005, S. 100–110

[Yao77] Yao, Andrew Chi-Chin: Probabilistic computations: Toward a unified measure of complexity. In: *Proc. of FOCS*, 1977, S. 222–227