Die Komplexitätsklasse BQP und ihre Beziehung zu klassischen Komplexitätsklassen

Carsten Schwarz[1]
Humboldt-Universität zu Berlin
Institut für Informatik

26. Mai 2004

[1] caschwar@informatik.hu-berlin.de

Zusammenfassung

Die bisherige Herangehensweise an Quantencomputer und an die Klasse der durch Quantencomputer effizient zu entscheidenden Sprachen (BQP) ist stark durch die Physik und eine für Außenstehende nicht leicht zugängliche Notation und Sprache geprägt. Ein Beschäftigung mit dem Thema scheint ohne Hintergrundwissen über Quantenmechanik nicht möglich.
Die vorliegende Arbeit stellt eine andere Annäherung an die Klasse BQP vor. Berechnungen sollen in Form von Manipulationen unitärer Matrizen erfolgen und können damit Quantenschaltkreise oder Quanten-Turing-Maschinen als Berechnungsmodelle ersetzen.
Des Weiteren soll die Komplexitätsklasse BQP in die bekannten Komplexitätsklassen eingeordnet werden.
Die Arbeit stützt sich hautpsächlich auf den Artikel [For00] von Lance Fortnow.

Inhaltsverzeichnis

1 Motivation

Zur Motivation geben ich nachfolgend die Definition der Komplexitätsklasse BQP an, wie sie sich beispielsweise in [Tor03] oder in ähnlicher Form in [NC00] findet.

Definition 1 *Eine Familie von Schaltkreisen $\{C_n\}$ heißt uniforme Schaltkreisfamilie, falls es eine polynomiell-zeitbeschränkte Turing-Maschine gibt, die bei Eingabe von n eine Beschreibung von C_n liefert.*

Definition 2 (BQP) *Sei $L \in BQP$ genau dann, wenn es eine uniforme Familie von Quantenschaltkreisen $\{C_n\}$ und ein Polynom p gibt, so dass für alle $x \in \Sigma^*$ gilt:*

- $x \in L \Rightarrow Prob(C_{|x|}(x) = 1) \geq \frac{2}{3}$ *und*
- $x \notin L \Rightarrow Prob(C_{|x|}(x) = 0) \geq \frac{2}{3}$ *und*
- C_n *hat höchstens $p(n)$ Gatter.*

Notation und Begrifflichkeit aus der Quantenmechanik können auf Außenstehende abschreckend wirken. Man gewinnt schnell den Eindruck, dass eine Beschäftigung mit diesem Bereich ohne genauere Kenntnisse der Quantenmechanik nicht möglich ist. Sowohl von Ethan Bernstein und Umesh Vazirani in [BV97] als auch von Lance Fortnow in [For00] ist gezeigt worden, dass eine andere, zu den bisherigen Definitionen äquivalente Definition der Komplexitätsklasse BQP möglich ist, die auf Begriffe aus der Quantenmechanik, auf komplexe Zahlen und die Bra- und Ket-Notation verzichtet. Berechnung wird hierbei als Manipulation von Matrizen angesehen.

2 Vorbemerkungen

Vorbereitend müssen wir noch einige Begriffe definieren, die im nachfolgenden benötigt werden.

Definition 3 (Turing-Maschine) *Eine Turing-Maschine ist gegeben durch ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \Box, E)$.*
Hierbei sind:

- Z *die endliche Zustandsmenge,*
- Σ *das Eingabealphabet,*

- $\Gamma \supset \Sigma$ *das Arbeitsalphabet,*
- $\delta : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$ *die Überführungsfunktion,*
- $z_0 \in Z$ *der Startzustand,*
- $\square \in \Gamma - \Sigma$ *das Blank-Zeichen und*
- $E \subseteq Z$ *die Menge der Endzustände.*

Anschaulich betrachtet besteht eine Turing-Maschine aus einem unendlichen Band, das in Felder eingeteilt ist. Jedes dieser Felder enthält genau ein Zeichen aus dem Arbeitsalphabet. Ein Schreib-Lesekopf bewegt sich schrittweise über das Band. Nur Zeichen, auf denen der Schreib-Lesekopf steht, können von der Maschine gelesen und beschrieben werden.
$\delta(z, a) = (z', b, x)$ bedeutet: Wenn sich M im Zustand z befindet und der Schreib-Lesekopf sich auf dem Zeichen a befindet, ersetzt M dieses Zeichen durch b, geht in den Zustand z' über und bewegt den Schreib-Lesekopf in x-Richtung.

Definition 4 (Konfiguration) *Eine Konfiguration c einer Turing-Maschine ist ein Wort $k \in \Gamma^* Z \Gamma^*$. Hierbei wird $k = \alpha z \beta$ so interpretiert, dass α und β die Bandinschrift und z der Zustand, in dem sich die Maschine gerade befindet, sind und der Schreib-Leekopf auf dem ersten Zeichen von β steht.*

Definition 5 *C sei die Anzahl der Konfigurationen einer gegebenen Turing-Maschine M bei Eingaben der Länge $|x|$. c_I sei die Anfangskonfiguration, c_A die einzige[1] akzeptierende Konfiguration von $M(x)$.*

3 Berechnung als Matrizenmanipulation

3.1 Komplexitätsklasse P

Definition 6 (P) *P ist die Klasse der Entscheidungsprobleme, die in polynomieller Zeit durch eine Turing-Maschine entscheidbar sind.*

Ausgehend von einer deterministischen Turing-Maschine M mit Überführungsfunktion δ kann man auf die nachfolgend beschriebene Art eine Übergangsmatrix angeben:

Definition 7 (Übergangsmatrix P) *T sei eine aus der Beschreibung der Übergangsfunktion δ von M entstandene $C \times C$ Übergangsmatrix. Dabei sei $T(c_a, c_b) = 1$, wenn man in einem Schritt von c_a nach c_b gelangen kann und $T(c_a, c_b) = 0$ sonst.*

Da es sich bei M um eine *deterministische* Turing-Maschine handelt, befindet sich in der Übergangsmatrix T in einer Zeile genau eine 1. Für jede Konfiguration der Maschine gibt es also genau eine Nachfolgekonfiguration.

Beobachtung 1 *Für alle Konfigurationen c_a und c_b gilt: $T^r(c_a, c_b) = 1$ genau dann, wenn sich M ausgehend von c_a bei Eingabe x nach r Schritten in c_b befindet.*

[1] Dies ist keine Beschränkung der Allgemeinheit, da man zu einer Turing-Maschine M mit mehreren akzeptierenden Konfigurationen eine Turing-Maschine M' mit genau einer akzeptierenden Konfiguration angeben kann, indem M bei Erreichen eines Endzustandes den Bandinhalt leert und in einen ausgezeichneten Zustand übergeht, der der einzige akzeptierende Zustand von M' ist. M' akzeptiert dadurch die gleiche Sprache wie M, verfügt jedoch über genau eine akzeptierende Konfiguration.

Begründung:

$$
\begin{gathered}
T^2(c_a, c_b) = 1 \\
\Leftrightarrow T^2(c_a, c_b) = \sum_{i=1}^{C} T(c_a, c_i) T(c_i, c_b) = 1 \\
\Leftrightarrow \exists c_k : T(c_a, c_k) T(c_k, c_b) = 1 \\
\Leftrightarrow T(c_a, c_k) = 1 \wedge T(c_k, c_b) = 1 \\
\Leftrightarrow M \text{ geht bei Eingabe } x \text{ ausgehend vom Zustand } c_a \text{ in den Zustand } c_k \\
\text{und von diesem in den Zustand } c_b \text{ über.}
\end{gathered}
$$

$$
\begin{gathered}
T^r(c_a, c_b) = 1 \\
\Leftrightarrow T^r(c_a, c_b) = \sum_{i=1}^{C} T^{r-1}(c_a, c_i) T(c_i, c_b) = 1 \\
\Leftrightarrow \exists c_k : T^{r-1}(c_a, c_k) T(c_k, c_b) = 1 \\
\Leftrightarrow T^{r-1}(c_a, c_k) = 1 \wedge T(c_k, c_b) = 1 \\
\Leftrightarrow M \text{ geht bei Eingabe } x \text{ ausgehend vom Zustand } c_a \text{ in } r-1 \text{ Schritten in den} \\
\text{Zustand } c_k \text{ und von diesem in den Zustand } c_b \text{ über.}
\end{gathered}
$$

Beobachtung 2 *$M(x)$ akzeptiert $\Longleftrightarrow T^t(c_I, c_A) = 1$*

Begründung:

Eine t-zeitbeschränkte deterministische Turing-Maschine akzeptiert genau dann, wenn sie in t Schritten ausgehend von der Anfangskonfiguration c_I die Endkonfiguration c_A erreichen kann.

3.2 Komplexitätsklasse NP

Definition 8 (NP) *NP ist die Klasse der Entscheidungsprobleme, die in polynomieller Zeit durch eine nichtdeterministische Turing-Maschine entscheidbar sind, so dass*

- *wenigstens ein Berechnungspfade akzeptieren, wenn die Antwort „ja“ ist und*
- *kein Berechnungspfad akzeptieren, wenn die Antwort „nein“ ist.*

Nun sei M eine nichtdeterministische Turing-Maschine mit Überführungsfunktion

$$\delta : Z \times \Gamma \times Z \times \Gamma \times \{L, R, N\} \rightarrow \{0, 1\}.$$

Definition 9 (Übergangsmatrix NP) *Mit dieser Überführungsfunktion erhält man wiederum eine Übergangsmatrix T mit $T(c_a, c_b) = 1$, wenn man in einem Schritt von c_a nach c_b gelangen kann und $T(c_a, c_b) = 0$ sonst.*

Im Gegensatz zu einer Übergangsmatrix, die aus der Überführungsfunktion einer deterministischen Turing-Maschine erstellt wird, kann es bei einer Übergangsmatrix, die aus der Überführungsfunktion einer *nicht*deterministischen Turing-Maschine erstellt wird, mehrere 1en in einer Zeile geben. Zu einer Konfiguration kann es also mehrere Nachfolgekonfigurationen geben.

Beobachtung 3 *Für alle Konfigurationen c_a und c_b gilt: $T^r(c_a, c_b)$ ist die Anzahl an Berechnungspfade von c_a nach c_b der Länge r.*

Begründung:

$$
\begin{gathered}
T^2(c_a, c_b) = l \\
\Leftrightarrow T^2(c_a, c_b) = \textstyle\sum_{i=1}^{C} T(c_a, c_i)T(c_i, c_b) = l \\
\Leftrightarrow \exists c_{k_1}, \ldots, c_{k_l} : T(c_a, c_{k_j})T(c_{k_j}, c_b) = 1 \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow \forall c_{k_1}, \ldots, c_{k_l} : T(c_a, c_{k_j}) = 1 \wedge T(c_{k_j}, c_b) = 1 \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow M \text{ kann bei Eingabe } x \text{ ausgehend vom Zustand } c_a \text{ in einen der Zustände} \\
c_{k_1} \text{ bis } c_{k_l} \text{ und von diesem in den Zustand } c_b \text{ übergehen.}
\end{gathered}
$$

$$
\begin{gathered}
T^r(c_a, c_b) = l'l \\
\Leftrightarrow T^r(c_a, c_b) = \textstyle\sum_{i=1}^{C} T^{r-1}(c_a, c_i)T(c_i, c_b) = l'l \\
\Leftrightarrow \exists c_{k_1}, \ldots, c_{k_l} : T^{r-1}(c_a, c_{k_j})T(c_{k_j}, c_b) = l' \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow \forall c_{k_1}, \ldots, c_{k_l} : T^{r-1}(c_a, c_{k_l}) = l' \wedge T(c_{k_j}, c_b) = 1 \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow M \text{ kann bei Eingabe } x \text{ ausgehend vom Zustand } c_a \text{ in } r-1 \text{ Schritten} \\
\text{(über } l' \text{ verschiedene Berechnungspfade) in einen der Zustände } c_{k_1} \text{ bis } c_{k_l} \\
\text{und von diesem in den Zustand } c_b \text{ übergehen.}
\end{gathered}
$$

Beobachtung 4 *$M(x)$ akzeptiert $\Longleftrightarrow T^t(c_I, c_A) \geq 1$*

Begründung:

Eine t-zeitbeschränkte nichtdeterministische Turing-Maschine akzeptiert genau dann, wenn sie über wenigstens einen Pfad verfügt, der sie in t Schritten von der Anfangskonfiguration c_I zur Endkonfiguration c_A bringt.

3.3 Komplexitätsklasse #P

Definition 10 (#P) *Die Klasse #P besteht aus Funktionen f, so dass für eine polynomiell-zeitbeschränkte nichtdeterministische Turing-Maschine M, $f(x)$ ist die Anzahl akzeptierender Berechnungen von $M(x)$.*

Wir definieren $\#M(x) := T^t(c_I, c_A)$ als die Anzahl der akzeptierenden Berechnungspfade der Maschine M bei Eingabe x. Für polynomiell-zeitbeschränkte Maschinen entspricht die Menge solcher Matrizen genau der Klasse #P.

Nun nehmen wir an, dass die Überführungsfunktion δ alle natürlichen Zahlen annehmen kann:

$$\delta : Z \times \Gamma \times Z \times \Gamma \times \{L, R, N\} \to \mathcal{N} \cup \{0\}.$$

Die Übergangsmatrix wird wie in den beiden vorherigen Fällen erstellt, mit dem Unterschied, dass die Einträge der Matrix nicht nur 0 oder 1, sondern alle natürlichen Zahlen und die 0 sein können.

Beobachtung 5 *Der Wert $T^r(c_a, c_b)$ ist die Summe über alle möglichen Berechnungspfade von c_a nach c_b, wobei in jedem möglichen Pfad das Produkt der Werte von δ jedes einzelnen Schrittes gebildet wird.*

Begründung:

$$
\begin{gathered}
T^2(c_a, c_b) = l'll'' \\
\Leftrightarrow T^2(c_a, c_b) = \sum_{i=1}^{C} T(c_a, c_i)T(c_i, c_b) = l'll'' \\
\Leftrightarrow \exists c_{k_1}, \ldots, c_{k_l} : T(c_a, c_{k_j})T(c_{k_j}, c_b) = l'l'' \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow \forall c_{k_1}, \ldots, c_{k_l} : T(c_a, c_{k_j}) = l' \wedge T(c_{k_j}, c_b) = l'' \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow M \text{ kann bei Eingabe } x \text{ ausgehend vom Zustand } c_a \text{ über } l' \text{ Pfade} \\
\text{in einen der Zustände } c_{k_1} \text{ bis } c_{k_l} \text{ und von diesem über } l'' \text{ Pfade} \\
\text{in den Zustand } c_b \text{ übergehen.}
\end{gathered}
$$

$$
\begin{gathered}
T^r(c_a, c_b) = l'll'' \\
\Leftrightarrow T^r(c_a, c_b) = \sum_{i=1}^{C} T^{r-1}(c_a, c_i)T(c_i, c_b) = l'll'' \\
\Leftrightarrow \exists c_{k_1}, \ldots, c_{k_l} : T^{r-1}(c_a, c_{k_j})T(c_{k_j}, c_b) = l'l'' \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow \forall c_{k_1}, \ldots, c_{k_l} : T^{r-1}(c_a, c_{k_l}) = l' \wedge T(c_{k_j}, c_b) = l'' \text{ für alle } j = 1, \ldots, l \\
\Leftrightarrow M \text{ kann bei Eingabe } x \text{ ausgehend vom Zustand } c_a \text{ in } r-1 \text{ Schritten über} \\
l' \text{ verschiedene Pfade in einen der Zustände } c_{k_1} \text{ bis } c_{k_l} \text{ und von diesem} \\
\text{über } l'' \text{ Pfade in den Zustand } c_b \text{ übergehen.}
\end{gathered}
$$

Beobachtung 6 *Dies entspricht wieder der Klasse #P.*

Begründung:

Angenommen $T(c_a, c_b) = k$ für beliebige c_a und c_b. Wir können eine neue nichtdeterministische Turing-Maschine erzeugen, die k Berechnungspfade der Länge $\log(k)$ von c_a und c_b hat. Dies erhöht die Laufzeit nur um einen konstanten Faktor.

Nun nehmen wir weiter an, dass die Überführungsfunktion δ alle nichtnegativen rationalen Zahlen annehmen kann:

$$\delta : Z \times \Gamma \times Z \times \Gamma \times \{L, R, N\} \to \mathcal{Q}^+ \cup \{0\}.$$

Auch zu dieser Überführungsfunktion kann wieder eine Übergangsmatrix erstellt werden, bei deren Einträgen es sich um nichtnegative rationale Zahlen handelt.

Beobachtung 7 *Dies führt uns erneut auf die Klasse #P.*

Begründung:

Sei v der kleinste gemeinsame Vielfache der Divisoren aller möglichen Werte von δ. Wir definieren $\delta_* := v\delta$. Seien T und T_* die korrespondierenden Matrizen zu δ und δ_*. $T_* = vT$.

3.4 Komplexitätsklasse BPP

Definition 11 (BPP) *BPP ist die Klasse der Entscheidungsprobleme, die in polynomieller Zeit durch eine nichtdeterministische Turing-Maschine entscheidbar sind, so dass*

- *wenigstens $\frac{2}{3}$ der Berechnungspfade akzeptieren, wenn die Antwort „ja“ ist und*
- *höchstens $\frac{1}{3}$ akzeptieren, wenn die Antwort „nein“ ist.*

Nun sei

$$\delta : Z \times \Gamma \times Z \times \Gamma \times \{L, R, N\} \to [0, 1]$$

die Überführungsfunktion einer probabilistische Maschine M mit der Bedingung, dass für alle q und a gilt:

$$\sum_{p,c,d} \delta(q, a, p, c, d) = 1$$

Definition 12 (Übergangsmatrix BPP) *Mit dieser Überführungsfunktion erhält man wiederum eine Übergangsmatrix T mit $T(c_a, c_b) = x$, wobei M mit Wahrscheinlichkeit x von c_a nach c_b gelangt.*

Wegen der obigen Bedingung summieren sich die Einträge jeder Zeile der Übergangsmatrix T auf 1. Bei T handelt es sich um eine stochastische Matrix. T erfüllt die L_1-Norm, d. h. für jeden Vektor u gilt:

$$L_1(uT) = L_1(u).$$

Beobachtung 8 *Nun berechnet $T^t(c_I, c_A)$ die Akzeptanzwahrscheinlichkeit von M.*

Begründung:

Die Wahrscheinlichkeit eines Berechnungspfades ist das Produkt der Übergangswahrscheinlichkeiten jedes einzelnen Schrittes in diesem Pfad. In $T^t(c_I, c_A)$ steht die Summe der Wahrscheinlichkeiten aller möglichen Berechnungspfade der Länge t von der Anfangskonfiguration c_I zur akzeptierenden Konfiguration c_A. Dies ist die Akzeptanzwahrscheinlichkeit von M.

Definition 13 (BPP) *Eine Sprache L ist in BPP genau dann, wenn es eine solche probabilistische Matrix T und ein Polynom t gibt, so dass:*

- $x \in L \Rightarrow T^t(c_I, c_A) \geq 2/3$ *und*
- $x \notin L \Rightarrow T^t(c_I, c_A) \leq 1/3$.

3.5 Komplexitätsklasse BQP

Definition 14 ($\mathbf{BQP_{Toran}}$) *Sei $L \in BQP$ genau dann, wenn es eine uniforme Familie von Quantenschaltkreisen $\{C_n\}$ und ein Polynom p gibt, so dass für alle $x \in \Sigma^*$ gilt:*

- $x \in L \Rightarrow Prob(C_{|x|}(x) = 1) \geq \frac{2}{3}$ *und*
- $x \notin L \Rightarrow Prob(C_{|x|}(x) = 0) \geq \frac{2}{3}$ *und*
- C_n *hat höchstens $p(n)$ Gatter.*

Nun soll die Überführungsfunktion δ auch negative rationale Zahlen annehmen können.

$$\delta : Z \times \Gamma \times Z \times \Gamma \times \{L, R, N\} \to \mathcal{Q}.$$

Da dadurch T^t auch negativ werden kann, werden wir $[T^t(c_I, c_A)]^2$ als unsere Akzeptanzwahrscheinlichkeit benutzen.

Weiter verlangen wir, dass T die L_2-Norm erfüllt, d.h. für jeden Vektor u gilt:

$$L_2(T(u)) = L_2(u) = \sqrt{\sum_a u_a^2}.$$

Dies ist gleichbedeutend mit der Forderung, dass die Matrix T unitär ist, da in einem Vektorraum V für jedes $x \in V$ gilt: Das Quadrat der Norm von Tx ist $(Tx, Tx) = (x, T^*Tx)$. Aus $T^*T = E$ folgt, dass T die Norm erhält. Zum Beweis der Rückrichtung sei $B = T^*T - E$. Da T die Norm erhält, gilt für jedes $x \in T$: $(x, T^*Tx) = (x, x)$. Daher folgt für jedes $x \in V$: $(x, Bx) = 0$. Daraus folgt, dass $B = 0$ und damit $T^*T = E$ ist.

Definition 15 ($\mathbf{BQP_{Fortnow}}$) *Eine Sprache L ist in BQP genau dann, wenn es eine solche Quanten-Matrix T und ein Polynom t gibt, so dass:*

- $x \in L \Rightarrow [T^t(c_I, c_A)]^2 \geq 2/3$ *und*
- $x \notin L \Rightarrow [T^t(c_I, c_A)]^2 \leq 1/3$.

Bernstein und Vazirani zeigen in [BV97], dass gilt:

$$BQP_{Toran} = BQP_{Fortnow}$$

4 Beziehung zu klassischen Komplexitätsklassen

Quantenschaltkreise sind in der Lage, klassische und probabilistische Schaltkreise zu simulieren[Tor03]. Damit ist klar, dass die Klasse BQP die Klasse BPP und P enthält:

$$P \subseteq BPP \subseteq BQP$$

Ob diese Klassen allerdings echt in der Klasse BQP enthalten sind, ist bisher nicht geklärt.
Ebenfalls interessant ist eine Abschätzung von BQP nach oben. Dazu müsse wir erst noch einige neue Komplexitätsklassen, nämlich die Klassen FP, GapP, PP und AWPP, einführen:

Definition 16 (FP) *Die Klasse FP besteht aus Funktionen f, für die es eine polynomiell-zeitbeschränkte Turing-Maschine M gibt, die f berechnet.*

Definition 17 (GapP) *Die Klasse GapP besteht aus Funktionen f, so dass für eine polynomiell-zeitbeschränkte nichtdeterministische Turing-Maschine M, $f(x)$ ist die Differenz zwischen akzeptierenden und verwerfenden Berechnungspfaden von $M(x)$.*

Definition 18 (PP) *Die Klasse PP besteht aus Sprachen L, so dass für ein $f \in$ GapP und für alle $x \in \Sigma^+$ gilt:*

- $x \in L \Rightarrow f(x) > 0$ *und*
- $x \notin L \Rightarrow f(x) < 0$.

Definition 19 (AWPP) *Die Klasse AWPP besteht aus Sprachen L, so dass für ein $f \in$ GapP und ein $g \in$ FP und für alle $x \in \Sigma^+$ gilt:*

- $x \in L \Rightarrow \frac{f(x)}{g(x)} > \frac{2}{3}$ *und*
- $x \notin L \Rightarrow \frac{f(x)}{g(x)} < \frac{1}{3}$.

Nach [For00] entspricht die Klasse AWPP der Menge der oben definierten Quanten-Matrizen, wenn man auf die Bedingung der Unitarität verzichtet. Damit wird deutlich, dass

$$BQP \subseteq AWPP$$

gilt (und auch in [FR98] gezeigt wird) und

$$AWPP \subseteq PP \subseteq PSPACE,$$

nach [Li93] und [FFKL93] gilt.

Zusammenfassend:

$$P \subseteq BPP \subseteq BQP \subseteq AWPP \subseteq PP \subseteq PSPACE$$

Damit liegt die Klasse der auf einem Quantenrechner effizient zu entscheidender Sprachen genau zwische den Klassen BPP, der Klasse der Sprachen, für die es eine Turing-Maschine gibt, die über doppelt so viele akzeptierende wie verwerfende Berechnungspfade verfügt, und die zur Zeit als die Klasse der effizient zu entscheidenden Sprachen angesehen wird, und der Klasse PP, der Klasse der Sprachen, für die es eine Turing-Maschine gibt, die über mehr akzeptierende als verwerfende Berechnungspfade verfügt, und die damit eine wesentlich größere Komplexitätsklasse ist.

Literatur

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.

[FFKL93] Fenner, Fortnow, Kurtz, and Li. An oracle builder's toolkit. In *SCT: Annual Conference on Structure in Complexity Theory*, 1993.

[For00] Lance Fortnow. One complexity theorist's view of quantum computing. *Electronic Notes in Theoretical Computer Science*, 31, 2000.

[FR98] Lance Fortnow and John D. Rogers. Complexity limitations on quantum computation. In *IEEE Conference on Computational Complexity*, pages 202–209, 1998.

[Li93] L. Li. *On the counting functions.* PhD thesis, University of Chicago, 1993.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge, United Kingdom, first edition, 2000.

[Sch01] Uwe Schöning. *Theoretische Informatik – kurzgefasst.* Spektrum, Akademischer Verlag, Heidelberg, Berlin, 4. edition, 2001.

[Tor03] Jacobo Torán. Skript zur Vorlesung Quantencomputer im SS 2003 an der Universität Ulm, 2003.