Die Berechnungsstärke von Quanten Turing Maschinen

Tassilo Singhammer

17.04.2003

Abstract

In dieser Arbeit soll diskutiert werden, welche theoretischen Möglichkeiten in einer Quanten-Turing-Maschine stecken. Zunächst sollen Problem-Klassen analog denen klassischer Turing Maschinen definiert werden. Danach wird versucht Schranken, sowohl obere als auch untere, für die Berechnungsstärke von QTMs zu finden. Schließlich soll anhand eines beispielhaften Problems, der rekursiven Fourier-Analyse, ein wichtiger Hinweis gegeben werden, dass die Berechnungsstärke von QTMs särker als die klassischer (probabilistischer) Turing Maschinen ist, in dem man zeigt, dass dieses Problem auf einer QTM in polynomialer Zeit gelöst werden kann, von einer klassischen TM jedoch (vermutlich) nicht.

1 Einleitung

Diese Ausarbeitung beruht auf Auszügen aus dem Buch "*Quantum complexity theory*" von E. Bernstein und Umesh Vazirani, Professor an der Berkley Universität. Wie aus dem Vortrag von Matthias Schmeißer und der Einführung von Martin Lange ersichtlich wurde, ziehen Quantencomputer ihre Stärke aus der Fähigkeit, Berechnungspfade zu überlagern. Ferner wurde in der Einführung erklärt, dass Quanten-Turing-Maschinen, im Folgenden meist durch **QTM** abgekürzt, durch klassische Turing-Maschinen, im Folgenen meist **TM**, simuliert werden können, jedoch mit exponentiellem Zeitaufwand. Dies legt die Vermutung nahe, dass es, mit der erfolgreichen Konstruktion eines Quantencomputers, möglich ist, mehr Probleme bewältigen zu können als mit klassischen Computern. Die in der Einführung vorgestellte Hierarchie der Komplexitäts-Klassen $P \subseteq NP \subseteq PSPACE \subseteq EXPTime$ wird verwendet um die QTM in dieser Hierarchie einzuordnen.

Dies soll und kann hier jedoch nur in einer groben Einschätzung erfolgen, indem theoretisch bestehende, obere Schranken für die Mächtigkeit von QTMs eruiert werden. Hauptsächlich geht es dabei um die hierarchische Stufe zwischen P und PSPACE was in Anschnitt 5 behandelt wird. Um diese Abschätzung vornehmen zu können werden in Abschnitt 3 die sog. Orakel-QTMs vorgestellt. Um jedoch auch optimistischere Aussichten zu eröffnen werden in Abschnitt 6 starke Indizien dafür geliefert, dass die Berechnungsstärke von QTMs über der klassischer (probabilistischer) TMs liegt. Die "Beweisführung" aus *Quantum complexity theory* gestaltet sich jedoch als äußerst diffizil, weshalb in Abschnitt 6 ein Beispiel aus [6] verwendet wird um die Realisierung ein Fourier-Transformation darszustellen.

2 Wie werden Sprachen von einer QTM akzeptiert?

`Definition 2.1` *Sei M eine stationäre Mehrband QTM in Normalform deren letzte Spur das Alphabet {#,0,1} benutzt. Lässt man M mit einem String x auf der ersten Spur und lehren Strings auf allen anderen Spuren laufen, wartet bis M anhält und betrachtet die letzte Spur der Startzelle, so erhält man mit Wahrscheinlichkeit p eine* 1*. M akzeptiert x mit Wahrscheinlichkeit p und lehnt x mit Wahrscheinlichkeit* $1-p$ *ab.*
`Definition 2.2` *Gegeben sei eine beliebige Sprache* $L \subseteq (\Sigma - \#)^*$*. Eine QTM akzeptiert eine Sprache* $L \subseteq (\Sigma - \#)^*$ *mit einer Wahrscheinlichkeit von mindestens p wenn sie mit einer Wahrscheinlichkeit von mindestens p jeden String* $x \in L$ *akzeptiert und jeden String* $x \in (\Sigma - \#)^* - L$ *mit einer Wahrscheinlichkeit von mindestens p ablehnt.*
`Definition 2.3` *Einen QTM M akzeptiert L genau dann, wenn M jeden String* $x \in L$ *mit Wahrscheinlichkeit* 1 *akzeptiert und jeden String* $x \in (\Sigma - \#)^* - L$ *mit Wahrscheinlichkeit 1 ablehnt.*

3 Orakel-QTMs

In diesem Abschnitt wollen wir ohne Einschränkung der Allgemeinheit davon ausgehen, dass als Aplphabet der TM für jede Spur $\{0, 1, \#\}$ verwendet wird. Alle Spuren, mit Ausnahme der Eingabe-Spur, sollen sollen mit *Blanks* initialisiert werden. Die Eingabe-Spur besteht aus der aktuellen Eingabe, umgeben von Blanks.
Σ soll $\{0, 1\}$ bezeichnen.
In der klassisschen Computerwelt kann ein Orakel informell als eine Maschine bezeichnet werden welche eine boolsche Funktion $f : \Sigma^* \to \Sigma$, mit

willkürlich gewählten Argumenten, berechnet. Dies ermöglicht uns Fragen der Form "wenn f auf einer TM effizient berechenbar ist, welche anderen Funktionen bzw. Sprachen können effizent von TMs verarbeitet werden?" Eine Orakel-QTM soll die äquivalente Frage im Quantenfall beantworten. Als Orakel-QTM O bezeichnet man eine QTM, welche eine spezielle "Orakel" Spur besitzt auf der die QTM Fragen an das Orakel schreiben kann. M^O bedeutet, dass es sich im eine QTM M handelt welcher das Orakel O zur Verfügung steht. Die Orakel-QTM bestitzt zwei besondere innere Zustände: einen *prequery* Zustand q_q und einen *postquery* Zustand q_a. Eine Anfrage (Query) wird bearbeitet, sobald die Maschine in den *prequery* Zustand eintritt wobei sich ein Block nicht leerer Zellen auf der Query-Spur befinden muss. Angenommen, der nicht leere Bereich der Query-Spur befindet sich im Zustand $|x \cdot b\rangle$ beim Eintritt in den prequery Zustand, wobei $x \in \Sigma^*$, $b \in \Sigma$ und "$\cdot$" die Konkatenation bedeutet. Sei f die boolsche Funktion die das Orakel berechnet. Das Ergebnis der Orakel-Anfrage ist der Übergang des Zustandes der Query-Spur in den Zustand $|x \cdot b \oplus f(x)\rangle$, "$\oplus$" bezeichnet das exklusive Oder. Andere Teile der Oracle QTM, abgesehen von der Query-Spur bleiben während der Anfrage unverändert. Wenn also das Ziel-Bit $|b\rangle$ mit $|0\rangle$ initialisiert war, wird sein End-Zustand $|f(x)\rangle$ sein, falls $|b\rangle$ bereits im Zustand $|f(x)\rangle$ war wird sein End-Zustand auf $|0\rangle$ zurück gesetzt.
Analog zu QTMs ziehen Oracle QTMs ihre Berechnungsstärke aus der Möglichkeit Anfragen zu überlagern. Wird beispielsweise ein Orakel für eine boolsche Funktion f im Spur-Zustand $|\Psi, 0\rangle = \sum_x \alpha_x |x, 0\rangle$ aufgerufen, so wird der Query-String in den Zustand $\sum_x \alpha_x |x, f(x)\rangle$ überführt.

`Lemma 3.1` *Sei M eine unidirektionale Orakel-QTM in Normalform mit den Anfangs- und Endzuständen q_0 und q_f und den Query-Zuständen $q_q \neq q_f$ und q_a. M' wird aus M genau wie im Beweis des Umkehrungs-Lemma [5] konstruiert. Dann existiert eine Orakel-QTM M' in Normalform, so dass M'^O für jedes Orakel O die Berechung von M^O umkehrt. Dabei werden zwei Zusätzliche Arbeitsschritte benötigt.*

4 Komplexitätsklassen

Analog zu den Komplexitätsklassen klassischer Turing Maschinen werden für die Quanten Turing Maschine Komplexitätsklassen gebildet. Probabilistische Turing Maschinen sind Turing Maschinen, welche eine Sprache mit einer gewissen Fehlerwahrscheinlichkeit annehmen bzw. ablehnen. Als Klasse **BPP** (Bounded error Probability in Polynomial time) wird diejenige Klasse von Problemen bezeichnet, welche von einer probabilistischen Turing Maschine, die, zwischen endlich vielen Alternativen, zufällige Entscheidungen trifft und mit einer Wahrscheinlichkeit von mindestens $\frac{2}{3}$ korrekter Weise akzeptiert bzw. ablehnt. In ähnlicher Weisse führt man die Klasse **BQP** für Quanten Turing Maschinen ein als die Klasse der effizient berechenbaren Sprachen auf einer QTM.

Allgemein wird die Klasse $\mathbf{BQTime}(T(n))$ definiert als die Menge aller Sprachen welche von einer QTM mit einer Wahrscheinlichkeit von $\frac{2}{3}$ bei einer Eingabe der Länge n mit maximalem Zeitaufwand $T(n)$ akzeptiert wird.

BQP wird, analog zu BPP, definiert als die Menge aller Sprachen die von einer QTM in polynomialer Laufzeit mit einer Wahrscheinlichkeit von mindestens $\frac{2}{3}$ akzeptiert wird.

Speziell wird die Klasse $\mathbf{EQTime}(T(n))$ wird definiert als die Menge aller Sprachen die von einer QTM mit Wahrscheinlichkeit 1 mit maximalem Zeitaufwand $T(n)$ akzeptiert. werden.
Die Klasse **EQP**(Exact bzw. Error-free Quantum Polynomial Time) wird definiert als die Menge der Sprachen welche von einer QTM mit Wahrscheinlichkeit 1 in polynomialer Zeit akzeptiert werden.

$\mathbf{BQTime}(T(n))^{O}$ sei definiert als die Menge der Sprachen, welche von einer Orakel-QTM, mit einer Wahrscheinlichkeit von mindestens $\frac{2}{3}$ in $T(n)$ akzeptiert wird.

5 Obere und untere Grenzen der Berechnungsstärke von QTMs

Offensichtlich gilt $\mathbf{EQP} \subseteq \mathbf{BQP}$, da EQP ein Spezialfall von BQP ist. Weiterhin ist **BQP** in **EXPTIME** enthalten. Dies ergibt sich aus der Tatsache, dass QTMs auf klassischen TMs simuliert werden können, jedoch mit exponentiellem Zeitaufwand.

`Satz 5.1` $\mathbf{P} \subseteq \mathbf{BQP}$.
Beweis. Da reversible Turing Maschinen Spezialfälle von Quanten Turing Maschinen sind gilt nach Bennett [1]: $\mathbf{BPP} \subseteq \mathbf{BQP}$. D.h. alle Probleme die mit einer probabilistischen Turing Maschine in polynomialer Zeit lösbar sind, sind auch auf einer Quanten-Turing-Maschine in polynomialer Zeit lösbar,

da diese eine porbabilistische TM mit maximal polynomialem Zeitaufwad simulieren kann.

`Satz 5.2` **BQP** $\subseteq$ **PSPACE**.
Beweis. Sei $M = (\Sigma, Q, \delta)$ eine **BQP** Maschine welche in der Zeit $p(n)$ hält. Nach Theorem 3.9[2] simuliert jede QTM *M'* die $\frac{\epsilon}{24(|\sigma|)(|Q|)p(n)} - nah$ an M arbeitet, M $p(n)$ Schritte mit Genauigkeit ϵ. Soll M mit einer Genauigkeit von $\frac{1}{12}$ simuliert werden, so Bleibt die Wahrscheinlichkeit eines erfolgreichen Durchgangs bei mindestens $\frac{7}{12}$. Es genügt also mit *M'* zu rechnen, wobei die Amplitude jeder Tranistion von M bis zu ihren ersten $\log(288card(\sigma)card(Q)p(n))$ berechnet werden muss. Die Amplitude jeder Konfiguration zu einer Zeit T ist die Summe der Amplituden jedes möglichen berechneten Pfades von *M'* der Länge T. Diese Amplitude kann in polynomialer Zeit exakt berechnet werden. Auf dem resultierenden Stack von höchstens $p(n)$ Zwischen-Konfigurationen kann nun eine Tiefen-Suche (engl. *depth-first*) des Berechnungsbaumes mit polynomialem Speicherbedarf durchgeführt werden (jedoch mit exponentiellem Zeitaufwand). Schließlich lässt sich feststellen, ob ein String x von M akzeptiert wird, indem wir die Summe der Quadrate der Amplituden aller Konfigurationen die M' erreichen kann und eine 1 in der Startzelle haben, mit $\frac{7}{12}$ vergleichen. Polynomialer Speicheraufwand genügt also, um die Summe der Quadrate der Amplituden aller Konfigurationen zu bilden.

Einschub
Die Klasse #**P**.
Ein Problem liegt in #**P** genau dann, wenn es eine nichtdeterministische Turing-Maschine mit polynimialer Laufzeit gibt, welche für jede Instanz I des Problems die Anzahl der akzeptierenden berechnungen angibt die der exakten Anzahl der eindeutigen Lösungen für die Instanz I entspricht.
Nach *Quantum complexity theory*: #**P** ist definiert als die Menge der Funktionen *f* welche Strings auf Integerwerte abbildet. Ferner gilt: Es gibt ein Polynom $p(n)$ und eine Sprache $L \in \mathbf{P}$, so dass, für jeden String x *f*(x) die Anzahl der Strings y mit Länge $p(|x|)$ berechnet für die gilt $xy \in L$. Die beiden Definitionen sind äquivalent wenn man sich die zweite Definition folgendermaßen vorstellt: Wenn man ein Problem als String bzw. Sprache sieht und die Anzahl seiner Lösungen mit der Eingabe konkateniert, in der Form xy, wobei x die Eingabe und y die Anzahl der Lösungen bedeutet, dann bedeutet #**P** die Sprache, die sich aus der ursprünglichen Sprache, welche x akzeptiert, und der Anzahl dieser Lösungen zusammensetzt.

Nach Vailiant [3] kann die obere Berechnebarkeitsgrenze weiter eingeschränkt werden auf $\mathbf{P}^{\#\mathbf{P}}$.
Jede **BQP** kann nach einem Theorem aus [4] in eine "saubere" Version M verwandelt werden welche bei einer Eingabe x eine End-Superposition mit dem größten Gewicht auf $x; M(x)$ erzeugt wobei $M(x) = 1$ falls Mx akzeptiert und 0 falls nicht.
`Theorem 5.3` *Wenn eine Sprache L in der Klasse **BQTime** enthalten ist, mit $T(n) > n$, so gibt es für jedes $\epsilon > 0$ eine QTM M' welche L mit Wahrscheinlichkeit $1 - \epsilon$ akzeptiert und und folgende Eigenschaft besitzt: Bei Eingabe von x mit Länge n läuft M' maximal mit Zeit $cT(n)$, wobei c polynomial in $\log \frac{1}{\epsilon}$ ist und erzeugt eine End-Superposition in der $|xL(x)\rangle$ die quadratische Amplitude von mindesten $1-\epsilon$ bestitzt, mit $L(x) = 1$ falls $x \in L$ und 0 sonst.*

`Satz 5.4` $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$
Beweis. Sei $M = (\Sigma, Q, \delta)$ eine BQP-Maschine mit Beobachtungszeit $p(n)$. Nach Theorem 3.2 können wir ohne Einschränkung der Algemeinheit davon ausgehen, dass M ein "saubere BQP Maschine" ist. Bei Eingabe von x beispielsweise bestizt die quadratische Amplitude der End-Konfiguration eine Wahrscheinlichkeit von mindestens $\frac{2}{3}$ für die Ausgabe von $x; M(x)$. Wiederum kann hier M durch M' ersetzt werden bei der die ersten
$b = \log(288(|\Sigma|)card(|Q|)p(n))$ Bit verarbeitet werden. Bei Eingabe von x erhalten wir, nach der Zeit $p(|x|)$, bei Ausgabe von $x; M(x)$, eine quadratische Amplitude der End-Konfiguration von mindestens $\frac{7}{12}$. Wir führen den Beweis, indem wir zeigen, wie ein Orakel für Probleme der Klasse $\#\mathbf{P}$ die Amplitude der finalen Konfiguration $x; 1$ von M', zur Zeit T, mit einer Fehlerwahrscheinlichkeit geringer als $\frac{1}{36}$ effizient berechnen kann. Da die tatsächliche Amplitude maximal 1 annehmen kann, muss das Quadrat der hier angenäherten Amplitude kleiner $\frac{1}{12}$ dem Quadrat der tatsächlichen Amplitude sein. Dies ist ersichtlich aus
$\left| \|\alpha\|^2 - \|\alpha^{'}\|^2 \right| \leq \|\alpha^{'} - \alpha\|^2 + 2\|\alpha\|\|\alpha^{'} - \alpha\| \leq \frac{1}{36}\left(2 + \frac{1}{36}\right) < \frac{1}{12}$.
Da die Erfolgswahrscheinlichkeit von M' mindestens $\frac{7}{12}$ beträgt, ist es möglich einen String x durch Vergleichen des Quadrats der genäherten Amplitude mit $\frac{1}{2}$ korrekt zu klassifizieren.
Es soll nun gezeigt werden, wie die oben beschribene Amplitude angenähert werden kann. Wir erinnern uns, dass sich die Amplitude einer Konfiguration zur Zeit T aus der Summe der Amplituden jedes Berechnungspfades der Länge T, von der Start-Konfiguration aus bis zur gewünschten Konfiguration, zusammensetzt. Die Amplitude jedes einzelen Pfades ist das Produkt

der Amplituden bei jeder Transition (bestimmt durch die Transitions Funktion von M') des Berechnungspfades. Da jede Amplitude aus einem reellen und einem imaginären Teil besteht, kann man sich dieses Produkt als Summe von 2^T Termen vorstellen von den jeder entweder rein reell oder rein imaginär ist. Die Amplitude der gewünschten Konfiguration setzt sich also aus der Summe dieser 2^T Terme über jeden Berechnungspfad zusammen. Teilt man diese Summe in vier Teile, die Summe der positiven reellen Terme, die Summe der negativen reellen Terme, die Summe der positiven imaginären Terme, die Summe der negativen imaginären Terme und berechnet jede mit einer Fehlerwahrscheinlichkeit geringer als $\frac{1}{144}$ mit Hilfe eines $\#\mathbf{P}$ Algorithmus, so erhält man aus der Differenz der ersten beiden und der letzten beiden die Amplitude der gewünschten Konfiguration mit einer Fehlerwahrscheinlichkeit von höchstens $\frac{1}{36}$.
Die Summe der positiven reellen Beiträge jedes Berechnungspfades kann wie folgt berechnet werden: Angenommen wir nehmen zu einer Konstante c, polynomial in T, die folgenden drei Eingaben welche in ihrer Länge polynomial in T sind: Eine Auswahl eines T-Schritte langen Berechnungspfades p von M', eine Wahl t eines der 2^T Terme und eine ganze Zahl w zwischen 0 und 2^{cT}. Man sieht leicht, dass es in polynomialer Zeit möglich ist herauszufinden, ob p tatsächlich ein Pfad von der Startkonfiguration M basierend auf der Eingabe x ist, ob t reell und positiv ist und ob der t-te Term der Amplitude des Pfades p größer als $\frac{w}{2^{cT}}$ ist.Wählt man einen Berechnungspfad p und einen Term t die diese Bedingungen erfüllen, ergibt sich die Anzahl w der Algorithmen die akzeptiert werden geteilt durch 2^{cT} zu $\frac{1}{2^{cT}}$ des Wertes des t-ten Terms aus Pfad p. Es genügt als, einen Pfad p zu messen der diese Bedingungen erfüllt und die Anzahl w der Terme t für die der Algorithmus akzeptiert wird, geteilt durch 2^{cT}, ist kleiner oder gleich $\frac{N}{2^{(c-1)T}}$ mal der Summe aller positiven reellen Terme aller Pfade von M' zur Zeit T. Da höchstens $2card(\Sigma)card(Q)$ mögliche Nachfolger für jede Konfiguration eines erlaubten Pfades von M existieren, liefert uns eine Wahl von $c > 1 + \frac{\log 144}{T} + \frac{2card(\Sigma)card(Q)\log T}{T}$ wie gewünscht $\frac{N}{2^{(c-1)T}} < 144$.
Analog existieren Algorithmen aus $\mathbf{P}^{\#\mathbf{P}}$ um die anderen drei Summen der Terme anzunähern.

6 Fourieranalyse und die Berechnungsstärke von QTMs

In diesem Abschnitt wird ein wichtiger Hinweis darauf gegeben, dass Quanten-Turing-Maschinen mächtiger als (probabilistische) Turingmaschinen sind. Wir definieren das rekursive Fourie-Analyse-Problem welches die Werte 0

oder 1 als Eingabe für eine boolsche Funktion nimmt. Ferner zeigen wir, dass das rekursive Fourieanalyse-Problem in **BQP** liegt. Auf der anderen Seite kann man beweisen, das, wenn eine solche boolsche Funktion von einem Orakel verarbeitet wird, das rekursive Fourieranalyse-Problem nicht in $\mathbf{BQTime}(n^{\log n})$ liegt. D.h. Das Problem der rekursiven Fourier-Analyse kann auf einer QTM in polynomialer Zeit gelöst werden, würde jedoch, verglichen mit einem Orakel, auf einer klassischen, probabilistischen TM superpolynomialen Zeitaufwand in Anspruch nehmen. In diesem Abschnitt wird gezeigt, dass sich die diskrete Fourier-Analyse auf einer QTM in $O(n^2)$ durchführen lässt. Der beste, (bisher) existierende Algorithmus für eine klassische TM, der sog. *Fast Fourier Transform* Algorithmus, benötigt jedoch $O(n2^n)$ Rechenschritte.
Warum betrachten wir hier ausgerechnet das Problem der diskreten Fourier-Transformation?
Die Quanten-Fourier-Transformation bildet die Grundlage für viele Quanten-Algorithmen wie z.B. die Quanten-Faktorisierung. Des weiteren ermöglicht die Quanten-Fourier-Transformation die sog. *Phasen-Schätzung*, die Näherung von Eigenwerten eines unitären Operators unter gewissen Umständen. Dies wiederum ermöglicht es Sortier- und Such-Algorithmen zu implementieren.

Eine *diskrete Fourier-Transformation* nimmt als Eingabe einen Vector komplexer Elemente, $x_0, \ldots, x_{N-1}$, mit Länge N als festem Wert. Als Ausgabe erhält man den trnsformierten Vector mit komplexen Elementen $y_0, \cdots, y_{N-1}$, definiert durch

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N}$$

Bei der *Quanten-Fourier-Transformation* handelt es sich um genau die gleiche Transformation mit einer leicht abgewandelten Notation. Die QFT auf einer Orthonormal-Basis $|0\rangle, \ldots, |N-1\rangle$ wird definiert durch

$$|i\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle$$

und kann für einen willkürlichen Zustand geschrieben werden als

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle$$

wobei die Amplituden y_k die diskreten Fourier-Transformationen der Amplituden von x_j bedeuten. Aus dieser Definition ist noch nicht ersichtlich, dass es sich bei der diskreten QFT um eine unitäre Transformation handelt.

Wir konstruieren daher einen unitären Quanten-Schaltkreis der die Fourier Transformation berechnet.
Im Folgenden wählen wir $N = 2^n$, mit einem Integer n, und als Basis $|0\rangle, \cdots, ket 2^n - 1$ für einen n qubit Quanten-Computer. Nützlicher weise schreiben wir den Zustand $|j\rangle$ auch in binärer Repräsentation $j = j_1 j_2 \dots j_n$. Formeller: $j = j_1 2^{n-1} + j_2 2^{n-2} + \dots + j_n 2^0$. Außerdem bezeichnet $0.j_l j_{l-1} \dots j_m$ die binäre Aufteilung $\frac{j_l}{2} + \frac{j_l+1}{4} + \dots + \frac{j_m}{2^{m-l+1}}$.
In [6] wird mit Hilfe algebraischer Umformungen gezeigt, dass sich die Quanten-Fourier-Transformation folgendermaßen repräsentieren lässt:

$$|j_1, \cdots, j_n\rangle \rightarrow \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)}{2^{n/2}}. \quad (6.1)$$

Diese Repräsentation ermöglicht es uns einen effizienten Schaltkreis mit Hilfe folgender, unitärer, Transformationen zu konstruieren:
Das Hadamard Gatter H und as Gatter R_k mit

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$R \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

Um zu sehen, dass der in Abb. 1 im Anhang abgebildete Schaltkreis eine Fourier-Transformation berechnet betrachten wir was passiert, wenn der Zustand $|j_1 \dots j_n\rangle$ Die anwendung des Hadamard Gatters auf das erste Bit führt in den Zustand

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right) |j_2 \dots j_n\rangle$$

überführt, da $e^{2\pi i 0.j_1} = -1$, falls $j_1 = 1$ und $+1$ sonst. Die Anwendung des kontrollierten R_2 Gatters erzeugt den Zustand

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right) |j_2 \dots j_n\rangle.$$

Weiter werden die kontrollierten Gatter R_3 bis R_n angewendet von denen jedes ein zusätzliches Bit zur Phase des Koeffizienten des ersten $|1\rangle$ hinzufügt. Am Ende dieser Prozedur erreichen wir folgenden Zustand

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right) |j_2 \dots j_n\rangle$$

Als nächstes führen wir die selbe Prozedur mit dem zweiten Qubit durch.

Das Hadamard Gatter liefert uns Zustand

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right)|j_3 \dots j_n\rangle$$

und die kontrollierten R Gatter, $R2$ bis R_{n-1} ergeben den Zustand

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right)|j_3 \dots j_n\rangle.$$

Dies führen wir in der selben Weise für jedes Qubit durch und erhalten einen End-Zustand

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)$$

Vertauschungsoperationen (gezeigt in [6], Kapitel 1.3.4) werden dann dazu benutzt, die Reihenfolge der Qubits umzukehren. Danach erhalten wir den Zustand

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle)\right)$$

Wenn wir das mit 6.1 vergleichen sehen wir, dass wir die gewünschte Ausgabe erhalten haben. Diese Konstruktion zeigt außerdem, dass die Fourier-Transformation unitär ist, da jedes Gatter in dem von uns verwendeten Schaltkreis unitär ist. Abbildung 6.3 zeigt ein Beispiel dieses Schaltkreises für die Fourier-Transformation von drei Qubits.

Wieviele Gatter braucht solch ein Schaltkreis nun? Wir beginnen mit einem Hadamard Gatter und $n-1$ bedingten Rotationen des ersten Qubits, insgesamt n Gatter. Es folgt ein Hadamard Gatter und $n-2$ bedingte Rotationen des zweiten Qubits, insgesamt $n+(n-1)$. Wenn wir so weiter machen erhalten wir, dass $n+(n-1)+\dots+1 = n(n+1)/2$ Gatter notwendig sind plus der Gatter für die Vertauschungen. Höchstens $n/2$ Vertauschungen werden benötigt von denen jede wiederum drei kontrollierte NOT-Gatter benötigt. Der Schaltkreis liefert also einen $O(n^2)$ Algorithmus um die Quanten-Fourier-Transformation durchzuführen.

Im Vergleich dazu benötigt der beste klassische Algorithmus der die diskrete Fouier-Transformation berechnet, wie z.B. der *Fast Fourier Transform* Algorithmus (FFT), $O(n(2^n))$ Rechenschritte. Das bedeutet, dass es exponentiell mehr Operationen auf einem klassischen Computer bedarf als auf einem Quantenrechner.

Abb. 2 im Anhang zeigt ein Beispiel eines solchen Schaltkreises für drei Qubits.

References

[1] C. H. Bennett, *Logical reversibility of computation*, IBM J. Res. Develop., 17 (1973)

[2] E. Bernstein und U. Vazirani, *Quantum complexity theory*, SIAM Journal on Computing, 26(5): 1423, October 1997

[3] L. Vailiant, Personal Communication, 1992

[4] H. Bennet, E. Bernstein, G. Brassard, und U. Vazirani, *Strength and weakness of quamtum computing*, SIAM J. Comput., 26 (1997), pp. 1510-1523

[5] M. Schmeisser, *Die universelle Quanten-Turing-Maschine*

[6] Michael A. Nielsen und Isaac L. Chuang, *Quantum Computation and Quantum Inforation*, Cambridge, 2000