Quantencomputer

Gregor Leander

August 2001

Inhaltsverzeichnis

1 Einleitung **1**

2 Das mathematische Modell **3**
2.1 Grundlegende Definitionen 3
2.2 Unitäre Transformationen 5
2.3 Reine Zustände und Qubits 6
2.4 Zusammengesetzte Systeme 8

3 Grundlegende Operationen **13**
3.1 Grundlegende Operationen 14
3.1.1 Logisches Nicht 14
3.1.2 XOR 15
3.1.3 Die Hadamard-Transformation 16
3.1.4 Die unitäre Transformation U_f 17
3.1.5 Ein universeller Werkzeugkasten 18
3.2 Komplexere Operationen 20
3.2.1 Addition 20
3.2.2 Addition modulo N 22
3.2.3 Multiplikation modulo N 24
3.2.4 Potenzieren mod N 25
3.2.5 Quanten-Fourier-Transformation 27
3.2.6 Berechnung von Eigenwerten 29

4 Die Quantenalgorithmen **33**
4.1 Der Algorithmus von D. Deutsch 33
4.2 Simons Algorithmus 35
4.3 Shors Algorithmus zur Faktorisierung 37
4.3.1 Einige Bemerkungen 37
4.3.2 Vorberechnungen zu Shors Algorithmus 39
4.3.3 Weitere Möglichkeiten, einen Faktor zu bestimmen 41
4.3.4 Der Algorithmus 42
4.3.5 Kettenbrüche 46
4.3.6 Eigenwerte in Quantenalgorithmen 48

4.3.7 Zwei Versionen des gleichen Algorithmus 50
4.3.8 Das RSA-Public-Key-Verfahren 51
4.4 Grovers Suchalgorithmus 51
4.4.1 Das Problem und die Lösung 51
4.4.2 Analyse des Algorithmus 54
4.4.3 Untere Grenzen für Suchalgorithmen 59

5 Simulationen mit jaQuzzi 63
5.1 Die CDRom 63
5.2 jaQuzzi 64
5.2.1 ein 3-Qubit-Addierer 64
5.2.2 Grovers Algorithmus für 5 Qubits 65

Kapitel 1

Einleitung

Die Idee einer Rechenmaschine, die quantenmechanische Prozesse ausnutzt, stammt ursprünglich von Feyman. Er bemerkte, daß es für klassische Rechner sehr aufwendig sei, quantenmechanische Prozesse zu simulieren, aber anstatt dies als ein Hindernis zu verstehen, sah Feyman hierin eine Chance. Er fragte, ob es im Umkehrschluss nicht möglich sei, mit Hilfe von geeigneten quantenmechanischen Experimenten Probleme zu lösen, die für klassische Computer praktisch nicht mehr lösbar sind. David Deutsch formalisierte in den achtziger Jahren den Begriff eines Quantencomputers und lieferte auch den ersten Algorithmus für diese fiktive Maschine, der kein klassisches Analogon besitzt.

Aufbauend auf den Ergebnissen von Deutsch entwickelte Daniel Simon einen Quantenalgorithmus, der die Periode einer Funktion bestimmt; diese Idee war die Grundlage für Peter Shors Lösung des Faktorisierungs-Problems. Shor beschreibt eine Methode, um mit Hilfe eines Quantencomputers eine natürliche Zahl mit logarithmischen Aufwand in Primfaktoren zu zerlegen. Diese Idee war für das gesamte Forschungsgebiet ein großer Durchbruch, denn das Problem der Faktorisierung spielt eine entscheidende Rolle in der modernen Public-Key-Kryptographie. Der am meisten verbreitete Algorithmus RSA ist nämlich gebrochen, wenn das Faktorisierungs-Problem lösbar wird. Shors Algorithmus ist damit nicht nur von theoretischem Interesse, sondern hat großen Einfluß auf die Sicherheit der Datenübertragung in modernen Computernetzen. Lov Grover entwickelte einen weiteren für die Kryptographie bedeutenden Algorithmus, der Einfluß auf die Sicherheit von symmetrischen Chiffrierverfahren hat. Diese Methode ermöglicht eine sehr viel effizientere Schlüsselsuche als sie auf herkömmlichen Computern möglich ist.

In dieser Arbeit gehe ich nicht auf die physikalischen Probleme ein, die bei dem Versuch auftreten, einen Quantencomputer zu realisieren. Vielmehr werden auf der Grundlage eines mathematischen Modells zunächst die grundlegenden Operationen erläutert und die wichtigsten Algorithmen beschrieben. Ausgangspunkt für meine Untersuchungen war dabei [Pi00].

Im ersten Kapitel wird das mathematische Modell erläutert, welches für die

Darstellung der Algorithmen genutzt wird. Der Zustand eines Quantencomputers ist nach diesem Modell ein Vektor in einem komplexen Vektorraum, und ein Rechenschritt wir als unitäre Transformation modelliert. Weiter wird erläutert, welche Bedeutung die Messung eines Quantenzustandes hat und wie sich Informationen in solchen Zuständen repräsentieren lassen. Der Begriff des Qubits und der Unterschied zum klassischen Bit werden erläutert.

Im zweiten Kapitel werden spezielle Transformationen erörtert, die für die Quantenalgorithmen eine entscheidende Rolle spielen. So wird unter anderem gezeigt, wie elementare logische und arithmetische Gatter für eine Quanten-Computer aufzubauen und zu komplexeren Schaltungen zusammenzusetzen sind.

Darauf aufbauend werden im dritten Kapitel die oben erwähnten Quantenalgorithmen dargestellt und detailliert beschrieben. Am Algorithmus von Deutsch erkennt man am besten die neuen Möglichkeiten von Quantencomputern. Die Idee von Simon zur Bestimmung der Periode einer Abbildung liefert die Grundlage für Shors Faktorisierungsverfahren. Dabei gelang es mir, einen Teil der Quantenrechnungen in einen klassischen Algorithmus auszulagern. Schließlich wird noch die Optimalität von Grovers Suchalgorithmus gezeigt.

Im letzten Kapitel wird einen Simulations-Programm für Quantencomputer vorstellt. Man hat die Möglichkeit, auf kleinen Quantencomputern Algorithmen zu testen und die Änderung der Zustände zu beobachten.

Es gibt viele Themen, die über den Rahmen dieser Arbeit hinausgehen. So gibt es Vorschläge für eine ganz neue Art der Schlüsselverteilung, die, basierend auf quantenmechanischen Prinzipien, eine beweisbar sichere Datenübermittlung ermöglicht. Ein anderes großes Gebiet sind die Fehlerkorrektur Verfahren für Quantencomputer. Ein weiteres für die Kryptographie spannendes Thema ist, wie man mit Hilfe von Quantencomputern neue Methoden der Public-Key-Kryptographie entwickeln kann, die gegen die Angriffe von Quantencomputern resistent sind.

Kapitel 2

Das mathematische Modell

In diesem Kapitel soll das Modell entworfen werden, auf welchem die Quantencomputer aufbauen. Dabei werden die Postulate der Quantenmechanik umgesetzt und in das Modell eingearbeitet.

2.1 Grundlegende Definitionen

Das erste Postulat besagt, wo sich die gesamte Quantenmechanik abspielt, und wie sich ein Zustand eines physikalischen Systems beschreiben läßt.

Postulat 1 *(Zustand eines Systems) Eine Zustands- oder Dichte-Matrix eines Systems ist ein positiv semidefiniter Operator ρ auf einem endlich dimensionalen Hilbertraum H mit Spur 1. Das Paar (H, ρ) wird ein Quanten-System genannt.*

□

Wir legen fest, daß das Skalarprodukt für alle Hilberträume linear in der ersten Komponente sein soll.

Da positiv semidefinite Abbildungen insbesondere hermitisch sind, läßt sich ρ als Summe von Projektionen auf Eigenräume darstellen. Hierbei sind alle Eigenwerte positiv und für Zustands-Matrizen ist die Summe aller Eigenwerte gezählt mit Vielfachheit 1.

Weiter unten werden wir einen einfacheren Speziallfall betrachten, wobei den Zustand eines Systems dann ein Vektor in einem Hilbertraum darstellt.

Im folgenden soll nun beschrieben werden, wie die Messung eines Zustandes zu modellieren ist; hierzu führen wir zunächst folgenden Begriff ein:

Definition 2.1.1 *(Observable) Ein Observable A in einem endlich dimensionalen Hilbertraum H ist ein hermitscher Endomorphismus.*

□

Bemerkung 2.1.2 *Jede Observable hat eine Darstellung als Spektral-Zerlegung, d.h. jede Observable ist von der Form*

$$A = \sum_{\alpha \in Sp(A)} \alpha P_\alpha$$

wobei $Sp(A)$ die Menge aller Eigenwerte von A und P_α jeweils die Projektion auf den Eigenraum von A mit Eigenwert α bezeichnet.

Postulat 2 *(Messung eines Systems) Eine Messung eines Systems im Zustand ρ mit Hilfe einer Observablen A ist eine Realisierung der Zufallsvariablen*

$$M_A : \Omega \to \mathbb{R}$$

mit

$$P(M_A = \alpha|\rho) := tr(\rho P_\alpha),$$

wobei P_α die Projektion auf den Eigenraum von A mit Eigenwert α darstellt, bzw. $P_\alpha \equiv 0$ wenn α kein Eigenwert von A ist. Der Maßraum $(\Omega, P, \mathcal{A})$ wird nicht weiter spezifiziert.

Ein System im Zustand ρ kollabiert nach der Messung in den gemessenen Zustand. Genauer: Liefert die Messung eines Systems den Eigenwert α von A so befindet sich das System nach der Messung in dem Zustand

$$\rho' := \frac{1}{tr(\rho P_\alpha)} P_\alpha \rho P_\alpha$$

Der Nenner dieses Ausdruckes ist immer ungleich Null, da dieser ja der Wahrscheinlichkeit entspricht, das entsprechende Ergebniss zu messen. Ein Nenner mit Wert Null tritt mit Wahrscheinlichkeit Null auf, also gar nicht. □

Eine Observable spiegelt also die möglichen Ergebnisse einer Messung wider. Jeder Eigenwert einer Observablen ist ein mögliches Ergebnis. Die Wahrscheinlichkeit, daß ein bestimmtes Ergebnis gemessen wird, hängt sowohl von der Observablen als auch von dem Zustand des Systems ab.

Bemerkung 2.1.3 *Für die Spur des neuen Zustandes ρ' gilt wieder $tr(\rho') = 1$, denn*

$$tr(P_\alpha \rho P_\alpha) = tr(\rho P_\alpha P_\alpha) = tr(\rho P_\alpha)$$

Lemma 2.1.4 *Eine erneute Messung des Systems im Zustande ρ' bezüglich der Observablen A liefert den Eigenwert α mit Wahrscheinlichkeit 1, und das System wird durch die Messung nicht erneut verändert.*

Beweis: Die Wahrscheinlichkeit, daß erneut α gemessen wird, ist

$$\begin{aligned} P(M_A = \alpha|\rho') &= tr(\rho' P_\alpha) \\ &= \frac{tr(P_\alpha \rho P_\alpha P_\alpha)}{tr(\rho P_\alpha)} \\ &= \frac{tr(P_\alpha \rho P_\alpha)}{tr(P_\alpha \rho P_\alpha)} \\ &= 1 \end{aligned}$$

da $P_\alpha P_\alpha = P_\alpha$ gilt. Der neue Zustand ist folglich mit Wahrscheinlichkeit 1

$$\rho'' = P_\alpha \rho' P_\alpha = \rho'$$

Zur Vollständigkeit sei nun $\beta \neq \alpha$, dann gilt

$$P(M_A = \beta|\rho') = tr(\rho' P_\beta) = \frac{tr(P_\alpha \rho P_\alpha P_\beta)}{tr(\rho P_\alpha)} = tr(0) = 0,$$

denn Eigenvektoren hermitscher Matrizen mit unterschiedlichen Eigenwerten sind immer paarweise orthogonal, und damit gilt $P_\alpha P_\beta = 0$ □

2.2 Unitäre Transformationen

Um Systeme, wie sie im vorherigen Abschnitt definiert wurden, in irgendeiner Weise als Computer benutzen zu können, ist es unbedingt nötig, daß gewisse Operationen eine kontrollierte Veränderung des Zustandes erlauben. Diese Operationen sind im Fall der Quantencomputer unitäre Abbildungen.

Postulat 3 *(Zeitliche Entwicklung) Die zeitliche Entwicklung eines Systems wird durch eine unitäre Transformation beschrieben.* □

Für unser Modell folgt daraus:

Definition 2.2.1 *Die Anwendung einer unitären Abbildung U auf ein System im Zustand ρ liefert ein System im Zustand*

$$\rho' := U \rho U^{-1}$$

□

Es ist wichtig, den Unterschied von Messungen und Transformationen eines Systems hervorzuheben. Ein Transformation ist eine Veränderung des Systems, ohne Informationen über den Zustand zu erhalten. Dies ist entscheidend, da nur so gesichert ist, daß das System nicht wie bei einer Messung kollabiert.

2.3 Reine Zustände und Qubits

Definition 2.3.1 *Für ein Vektor $\psi \in H$ bezeichne P_ψ folgende Abbildung*

$$P_\psi(u) =< \psi, u > \psi$$

Für Einheitsvekoren ψ ist dies die Projektion auf den von ψ aufgespannten Untervektorraum, ansonsten die Projektion und eine skalare Multiplikation um dem Faktor $< \psi, \psi >= ||\psi||^2$. □

Definition 2.3.2 *(reiner Zustand) Ein System befindet sich in einem reinen Zustand, genau dann, wenn die Zustandsmatrix ρ des Systems eine Projektion auf einen eindimensionalen Untervektorraum von H ist.*

□

Damit sind reine Zustände ein Spezialfall der allgemeinen Definition. Ein Zustand ist im allgemeinen eine konvexe Linearkombination aus reinen Zuständen. Diese reinen Zustände sind für die Darstellung von Quantenalgorithmen eine entscheidende Vereinfachung. Sei also nun

$$\rho = P_\psi \text{ mit } \psi \in H \text{ und } ||\psi||^2 = 1$$

die Projektion auf den von ψ aufgespannten Untervektorraum. Der Zustand ρ ist damit durch ψ eindeutig festgelegt. Um die Veränderung eines reinen Zustandes durch Messungen oder unitäre Abbildungen darzustellen, reicht es somit, sich auf ψ statt auf P_ψ zu beziehen. Im folgenden sprechen wir auch von einem System im Zustand ψ und meinen damit ein System im Zustand P_ψ.

Lemma 2.3.3 *Sei ein System im Zustand $\psi \in H$ und eine Observable A über H gegeben. Weiterhin sei U eine unitäre Abbildung von H in sich. Dann gilt:*

1. *Das System wird nach der Anwendung der Abbildung U durch*

   $$\psi' = U\psi$$

   beschrieben.

2. *Nach der Messung eines Eigenwerts α der Observablen A bezüglich ψ befindet sich das System im Zustand*

   $$\psi'' = \frac{P_\alpha \psi}{||P_\alpha \psi||}.$$

   Weiter gilt:

   $$tr(P_\psi P_\alpha) =< \psi, P_\alpha \psi >= ||P_\alpha \psi||^2$$

   also ist die Wahrscheinlichkeit, den Eigenwert α zu messen, gegeben durch das Skalarprodukt des Zustandes vor mit dem Zustand nach der Messung.

Beweis:

1. Zu zeigen ist für den Fall unitärer Transformationen $P_{U\psi} = UP_\psi U^{-1}$.

$$\begin{aligned} P_{U\psi}(v) &= < U\psi, v > U\psi \\ &= < \psi, U^{-1}v > U\psi \\ &= U(< \psi, U^{-1}v > \psi) \\ &= UP_\psi U^{-1}v \end{aligned}$$

2. Der Beweis für den zweiten Teil ist sehr ähnlich. Für den Zustand nach der Messung ist zu zeigen, dass $P_\alpha P_\psi P_\alpha = P_{P_\alpha \psi}$ gilt.

$$\begin{aligned} P_{P_\alpha\psi}(v) &= < P_\alpha\psi, v > P_\alpha\psi \\ &= < \psi, P_\alpha v > P_\alpha\psi \\ &= P_\alpha(< \psi, P_\alpha v > \psi) \\ &= P_\alpha P_\psi P_\alpha v \end{aligned}$$

   Für die Wahrscheinlichkeit gilt dann:

$$tr(P_\psi P_\alpha) = tr(P_\alpha P_\psi P_\alpha) = tr(P_{P_\alpha\psi}) =< \psi, P_\alpha\psi >$$

   Denn der einzige Eigenvektor von $P_{P_\alpha\psi}$ ist $P_\alpha\psi$, und dieser hat den Eigenwert $< \psi, P_\alpha\psi >= ||P_\alpha\psi||^2$.

□

Die reinen Zustände entsprechen der Vorstellung, daß ein System keinen Einflüssen von außen unterliegt. Dies wird weiter unten an einem Beispiel deutlich gemacht werden.

Mit Lemma 2.3.3 wird klar, daß sich reine Zustände einfacher handhaben lassen. Für die Beschreibung von Quantencomputern wollen wir nun noch spezieller werden. Der Hilbertraum H wird für die Quantencomputer immer $\mathbb{C}^2$ sein, und somit erhalten wir 2 Basiszustände. Hierfür wollen wir nach [Sh95] folgende Notation einführen.

Definition 2.3.4 *(Qubit) Ein Einheitsvektor in $H = \mathbb{C}^2$ und damit ein reiner Zustand eines Systems, wird ein Qubit genannt. Für die Standard Basis führen wir folgende Notation ein:*

$$|\mathbf{0}\rangle := e_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \; und \; |\mathbf{1}\rangle := e_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

□

Diese Notation ist angelehnt an die in der Physik übliche "Bra" und "Ket" Notation. Die Tricks, die normalerweise mit dieser Notation verbunden sind, werden hier aber nicht genutzt.

Hier wird nun der Unterschied zu dem klassischen Begriff des Bits deutlich. Ein Qubit ist eine Kombination der Zustande $|\mathbf{0}\rangle$ und $|\mathbf{1}\rangle$, kann also in gewisser Weise beide Werte gleichzeitig annehmen, während ein klassisches Bit immer nur entweder 0 oder 1 sein kann. Wir können nun für ein Qubit eine Observable angeben, deren Eigenvektoren die Basisvektoren $|\mathbf{0}\rangle$ und $|\mathbf{1}\rangle$ sind, dies können wir dann als eine Messung verstehen, die ermittelt, ob das Qubit den Wert Eins oder Null trägt. Dies kann beispielsweise durch folgende Matrix geleistet werden:

$$A = \frac{1}{2}P_{|\mathbf{0}\rangle} - \frac{1}{2}P_{|\mathbf{1}\rangle} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.1}$$

Hierbei wird der Zustand des Qubits aber verändert. War der Zustand vor der Messung von der Form

$$\psi = a_0|\mathbf{0}\rangle + a_1|\mathbf{1}\rangle$$

dann wird mit einer Wahrscheinlichkeit von $|a_0|^2$ der Eigenwert von $P_{|\mathbf{0}\rangle}$ gemessen, denn $P(M = \frac{1}{2}|P_\psi) =< \psi, P_{|\mathbf{0}\rangle}\psi >= |a_0|^2$. Der neue Zustand wäre dann $\psi' = |\mathbf{0}\rangle$.

Bemerkung 2.3.5 *Wenn wir davon sprechen, daß ein Qubit gemessen werden soll, so meinen wir immer eine Messung mit der Observablen aus 2.1. Weiter werden wir auch sagen, für ein Qubit wird der Zustand $|\mathbf{0}\rangle$ gemessen, wenn der entsprechende Eigenwert gemessen wird.*

2.4 Zusammengesetzte Systeme

Um komplexe Aufgaben wie Faktorisieren oder die Suche in großen Datenmengen auf Quantencomputern durchzuführen, ist es nötig, Systeme mit mehreren, unter Umständen sehr vielen Qubits zu betrachten. Hier folgt nun ein weiteres Postulat:

Postulat 4 *(Zusammengesetzte Systeme) Seien zwei Zustandsräume V, W gegeben. Die adäquate mathematische Formulierung für ein aus diesen beiden Zustandsräumen zusammengesetzten System liefert der Begriff des Tensorproduktes.* □

Definition 2.4.1 *(Tensorprodukt) Seien H und H' endlichdimensionale Hilberträume mit Orthonormalbasis $\{w_1, ..., w_n\}$ bzw. $\{v_1, ..., v_n\}$ dann bezeichne $T = H \otimes H'$ das Tensorprodukt beider Räume. Eine Basis von T ist gegeben durch $\{w_i \otimes v_j\}_{i,j}$. Die Dimension von T ist das Produkt der Dimensionen der Teilräume. Weiter definieren wir für $u_{1,2}, v_{1,2} \in H$:*

$$< u_1 \otimes v_2, u_2 \otimes v_2 >:=< u_1, v_1 >< u_2, v_2 > .$$

Die lineare Fortsetzung liefert eine Skalarprodukt auf T, und T mit diesem Skalarprodukt ist selbst wieder ein Hilbertraum. □

Der Begriff läßt sich kanonisch auf endlich viele Faktoren fortsetzen und bildet so die Basis zum Aufbau komplexer Systeme. Ein System aus n Qubits wird somit durch einen 2^n dimensionalen $\mathbb{C}$ Vektorraum beschrieben, und ein reiner Zustand eines solchen Systems ist ein Vektor in diesem Raum, d.h. eine Kombination aus den 2^n verschiedenen Basisvektoren. Für das Tensorprodukt von Basis-Qubits führen wir noch folgende Notation ein

$$|\mathbf{a_n a_{n-1} ... a_1}\rangle := |\mathbf{a_n}\rangle \otimes |\mathbf{a_{n-1}}\rangle \otimes ... \otimes |\mathbf{a_1}\rangle \quad |\mathbf{a_k}\rangle \in \{|\mathbf{0}\rangle, |\mathbf{1}\rangle\}$$

Die a_i bilden also Multiindizes, die alle Basisvektoren durchlaufen. Wenn wir nun diese Multiindizes als Binärdarstellung einer natürlichen Zahl $0 \leq a \leq 2^n - 1$ verstehen, können wir den Basisvektor $|\mathbf{a_1 a_2 ... a_n}\rangle$ auch einfach mit $|\mathbf{a}\rangle$ bezeichen, wenn $a = \sum_{i=1}^n a_i 2^{i-1}$ gilt. Ein Vektor in dem Tensorprodukt der n Qubits hat dann allgemein folgende Darstellung

$$\psi = \sum_{a=0}^{2^n-1} \alpha_a |\mathbf{a}\rangle$$

Bemerkung 2.4.2 *Angelehnt an die Sprache der Informatik wird ein System aus n Qubits auch als Register der Länge n bezeichnet.*

Viele Eigenschaften der Quantenphysik werden erst deutlich, wenn man ein System aus mehreren Partikeln betrachtet, und einiges davon soll nun kurz dargestellt werden. Ein Zustand, der aus 2 Qubits zusammengesetzt ist, hat also im allgemeinen folgende Darstellung

$$\psi = a_{00}|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle + a_{01}|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle + a_{10}|\mathbf{1}\rangle \otimes |\mathbf{0}\rangle + a_{11}|\mathbf{1}\rangle \otimes |\mathbf{1}\rangle \tag{2.2}$$

Da sich obiger Zustand nicht immer als Produkt schreiben läßt, kann man sagen, daß es sich hierbei nicht immer um einen Zustand von zwei Qubits handelt, sondern vielmehr um einen Zwei-Qubit-Zustand.

Wir wollen nun in einem solchen Zustand das erste Qubit messen. Das bedeutet, wir messen den Zustand 2.2 bezüglich einer Observablen $A \otimes I$, für das A aus 2.1. Dies läßt sich auch schreiben als:

$$A \otimes I = \frac{1}{2} P_{|0x\rangle} - \frac{1}{2} P_{|1x\rangle}$$

wobei $P_{|0x\rangle}$ die Projektion auf den zweidimensionalen Untervektorraum ist, der von $|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle$ und $|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle$ aufgespannt wird, und $P_{|1x\rangle}$ analog.

Die Bedeutung dieser Theorie soll im folgenden weiter erläutert werden. Es sei folgender Produktzustand gegeben.

$$\psi = (a_1|\mathbf{0}\rangle + b_1|\mathbf{1}\rangle) \otimes (a_2|\mathbf{0}\rangle + b_2|\mathbf{1}\rangle)$$

Dies ist also ein Zustand aus zwei Qubits. Dies läßt sich dann auch schreiben als

$$\psi = a_1 a_2 |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle + a_1 b_2 |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle + b_1 a_2 |\mathbf{1}\rangle \otimes |\mathbf{0}\rangle + b_1 b_2 |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle$$

Wenn nun das erste Qubit gemessen wird, also mit Hilfe der Observablen $A \otimes I$, ergibt sich unabhängig vom zweiten Qubit mit Wahrscheinlichkeit $|a_1|^2$ der Wert 0 und mit Wahrscheinlichkeit $|b_1|^2$ der Wert 1. Wenn eine 0 gemessen wurde, ist der Zustand nach der Messung

$$\psi' = |\mathbf{0}\rangle \otimes (a_2 |\mathbf{0}\rangle + b_2 |\mathbf{1}\rangle)$$

und eine Messung des zweiten Qubits würde mit Wahrscheinlichkeit $|a_2|^2$ den Wert 0 und mit Wahrscheinlichkeit $|b_2|^2$ den Wert 1 ergeben. Dies wäre aber auch der Fall, wenn wir erst das zweite und dann das erste Qubit gemessen hätten. Anders ausgedrückt: die Meßergebnisse der zwei Qubits sind in diesem Fall unabhängige Zufallsvariablen.

Wenn wir nun den allgemeinen Fall 2.2 betrachten, sieht das schon anders aus. Sei also:

$$\psi = a_{00} |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle + a_{01} |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle + a_{10} |\mathbf{1}\rangle \otimes |\mathbf{0}\rangle + a_{11} |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle$$

Die Wahrscheinlichkeit, daß für das erste Qubit eine 0 gemessen wird, ist hier

$$p = < \psi, P_{|0x\rangle} \psi > = |a_{00}|^2 + |a_{01}|^2$$

und der Zustand nach der Messung ist

$$\begin{aligned} \psi' &= \frac{1}{\sqrt{|a_{00}|^2 + |a_{01}|^2}} P_{|0x\rangle} \psi && (2.3) \\ &= \frac{1}{\sqrt{|a_{00}|^2 + |a_{01}|^2}} \left(a_{00} |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle + a_{01} |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle \right) && (2.4) \\ &= \frac{1}{\sqrt{|a_{00}|^2 + |a_{01}|^2}} \left(|\mathbf{0}\rangle \otimes (a_{00} |\mathbf{0}\rangle + a_{01} |\mathbf{1}\rangle) \right) && (2.5) \end{aligned}$$

Eine Messung des zweiten Qubits ergibt nun mit Wahrscheinlichkeit

$$\frac{|a_{00}|^2}{|a_{00}|^2 + |a_{01}|^2}$$

den Wert 0. Hätten wir zuerst das zweite Qubit gemessen, also die Observable $I \otimes A$ benutzt, ergibt sich der Wert 0 mit Wahrscheinlichkeit $|a_{00}|^2 + |a_{10}|^2$. Das bedeutet, daß die Messungen der beiden Qubits in diesem Fall abhänige Zufallsvariablen darstellen.

In folgendem Zustand wird dies besonders deutlich

$$\psi = |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle + |\mathbf{1}\rangle \otimes |\mathbf{0}\rangle$$

wobei hier der Einfachheit halber der Normierungsfaktor $\frac{1}{\sqrt{2}}$ weggelassen wurde. Wenn das erste Qubit gemessen wird und als Ergebniss eine 0 liefert, so ist der Zustand des Systems nach der Messung:

$$\psi' = |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle$$

und eine Messung des zweiten Qubits ergibt mit Wahrscheinlichkeit 1 den Wert 1. Eine Messung des ersten Qubits legt also den Zustand des zweiten Qubits eindeutig fest.

Dieses Phänomen ist nach Einstein, Podolsky und Rosen auch als EPR-Phänomen bekannt und ließ viele Physiker an der Richtigkeit der Quanten-Theorie zweifeln, denn woher sollte das andere Qubit, welches unter Umständen einige Meter entfernt existiert von der Messung des ersten wissen?

Hier wird auch noch einmal die Bedeutung der Dichte Matrix deutlich. Denn wenn wir nun den Zustand des ersten Qubits beschreiben wollen, so stellen wir fest, das es sich hierbei nicht mehr um einen reinen Zustand handelt, sondern um einen Zustand, der sich nur als Dichte Matrix schreiben läßt. Ein Qubit verliert also dann, wenn es mit der Umwelt in Wechselwirkung tritt, seinen reinen Zustand. Wenn wir alle beteiligten Teilchen gemeinsam betrachten, so ergibt sich wieder ein reiner Zustand. Für weitere Informationen siehe [NC00], [Sh94] oder [Su86].

Diese Phänomene sollen hier nicht weiter erläutert werden, wir nehmen sie als Eigenschaften des mathematischen Modells zur Kenntnis ohne weiter auf ihre Bedeutung einzugehen.

Damit ist das mathematische Modell zur Beschreibung der Quantencomputer und ihrer Algorithmen beschrieben und wird in den folgenden Kaptiteln angewandt.

Kapitel 3

Grundlegende Operationen

In diesem Abschnitt soll untersucht werden, welche Operationen sich durch unitäre Transformationen realisieren lassen, und mit welchem Aufwand diese durchzuführen sind. Dazu werden aus einfachen Operationen komplexere zusammengesetzt. Weiter werden wir sehen, welche grundlegenden Operationen nötig sind, um beliebige Transformationen umzusetzen oder zumindest zu approximieren. Als Einstieg soll folgende Überlegung dienen:

Bei den heutigen experimentellen Konstruktionen von Quantencomputern scheint es nur schwer möglich, unitäre Transformationen zu realisieren, die viele Qubits auf einmal beeinflussen; dies ist allerdings, zumindest theoretisch, keine Einschränkung, wie folgendes Lemma zeigt.

Lemma 3.0.3 *(Deutsch 1985) Sei U eine unitäre Abbildung auf einem Hilbertraum H mit $dim(H) = n$, weiter sei $\{w_i\}$ eine Basis von H. Dann läßt sich U darstellen als Produkt aus $2n^2 - n$ unitären Matrizen, die jeweils nur auf einem von 2 Basisvektoren aufgespannten Untervektorraum operieren.*

Beweis: siehe [Pi00] □

Das bedeutet, daß es ausreichend ist, unitäre Operationen zu realisieren, die auf zwei Qubits operieren. Das Problem bei der Zerlegung einer beliebigen unitären Transformation in Operationen, die nur auf zwei Qubits operieren, ist, daß die Anzahl der nötigen Transformationen quadratisch mit der Dimension des Zustandsraumes und somit exponentiell mit der Anzahl der Qubits wächst. Deshalb reicht diese theoretische Bedingung nicht aus, sondern wir müssen für elementare Transformationen feststellen, mit wie vielen Teiltransformationen sie sich realisieren lassen. Weiterhin ist nicht zu erwarten, daß jede unitäre 2×2-Matrix technisch realisierbar ist, sondern daß es nur einen festen Satz von unitären Abbildungen gibt, die tatsächlich umsetzbar sind. Allerdings ist es möglich, durch geschickte Wahl dieser Abbildungen jede unitäre Abbildung zu approximieren, also einen endlichen Satz von Abbildungen zu wählen, die eine dichte Teilmenge aufspannen, wie wir in 3.1.5 sehen werden. Die geeignete Wahl wird aber sehr von der letztendlichen Realisierung eines Quantencomputers abhängen.

Als erstes wollen wir uns mit unitären Matrizen befassen, die auf den Basisvektoren eines Quantencomputers als logische Gatter wirken. Wie wir gesehen haben, werden die Berechnungen eines Quantencomputers immer durch unitäre Abbildungen dargestellt, das heißt, insbesondere jede Berechnung muß umkehrbar sein. Aus diesem Grund ist es auch nicht möglich, die klassischen logischen Gatter einfach zu übernehmen, da diese oft irreversible Berechnungen darstellen (zum Beispiel das logische UND auf 2 Bits). Ziel dieses Abschnittes ist es darzustellen, wie die elementaren logischen Operationen für Quantencomputer auszusehen haben, wobei hier eine Darstellung als Diagramm eingeführt wird, die sich bei der Beschreibung komplexerer Operationen als sehr nützlich erweisen wird. Weiter wird die Hadamard-Transformation vorgestellt, die in fast allen Algorithmen aus Kapitel 4 eine zentrale Rolle spielen wird. Außerdem soll erläutert werden, was es bedeutet, eine Funktion von $\mathbb{Z}_2^n$ nach $\mathbb{Z}_2^m$ auf einem Quantencomputer auszuwerten. Im zweiten Teil des Kapitels werden diese Gatter dann zu komplexeren Operationen zusammengesetzt, und ihre Durchführbarkeit wird erläutert.

3.1 Grundlegende Operationen

3.1.1 Logisches Nicht

Neben speziellen Transformation, wie die Hadamard-Transformation oder der Quanten-Fourier-Transformationen, die weiter unten erläutert werden, spielen natürlich auch einfache logische Operationen eine entscheidende Rolle bei der Entwicklung komplexer Algorithmen. So zum Beispiel das logische "Nicht", welches im Kontext der Quantencomputer eine unitäre Transformation auf einem Qubit darstellt. Das logische "NICHT" hat als Matrix folgende Gestalt.

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

und liefert somit auf den Basiszuständen die Operation

$$|\mathbf{0}\rangle \to |\mathbf{1}\rangle$$
$$|\mathbf{1}\rangle \to |\mathbf{0}\rangle$$

Wenn nun ein System mit zwei Qubits vorliegt und auf das erste ein "NICHT" angewandt werden soll, benötigt man die Operation $N \otimes I$, also ein "NICHT" auf dem ersten und die Identität auf dem zweiten Qubit. Die Darstellung dieser Operation als Matrix wird schon relativ kompliziert. Ein Zustand von zwei Qubits hat als Darstellung

$$\psi = a_{00}(|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle) + a_{01}(|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle) + a_{10}(|\mathbf{1}\rangle \otimes |\mathbf{0}\rangle) + a_{11}(|\mathbf{1}\rangle \otimes |\mathbf{1}\rangle) = \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix}$$

und bezüglich der Standardbasis ergibt sich folgende Matrix

$$N \otimes I = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Wie bei allen Operationen, die auf den Basisvektoren als logische Operation operieren, handelt es sich hierbei "nur" um eine Permutation der Basisvektoren. Da die Dimension des Zustandsraumes und damit auch der Matrix exponentiell mit der Anzahl der Qubits wächst, hat sich eine Notation als Schaltdiagramm etabliert. Hierbei ist die Vorstellung von Kabeln einer Schaltung und Operationen auf einzelnen Qubits dargestellt. Jedes Kabel steht für ein Qubit und die Schaltung soll von links nach rechts durchlaufen werden. Operationen sind durch unterschiedliche Symbole angedeutet, wobei die beteiligten Qubits durch vertikale Verbindungen mit angegeben werden müssen. Obige Operation hätte dann die Darstellung:

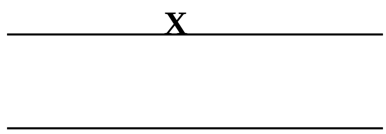

Abbildung 3.1: Das logische Nicht auf dem ersten von zwei Qubits. Der obere Strich steht für das erste Qubit, der untere für das zweite. Das X bezeichnet das logische Nicht.

3.1.2 XOR

Eine weitere grundlegende logische Transformation ist die XOR-Verknüpfung. Also die Abbildung:

$$|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle \rightarrow |\mathbf{a}\rangle \otimes |(\mathbf{a}+\mathbf{b})\rangle$$

d.h.

$$\begin{aligned} |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle &\rightarrow |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle \\ |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle &\rightarrow |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle \\ |\mathbf{1}\rangle \otimes |\mathbf{0}\rangle &\rightarrow |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle \\ |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle &\rightarrow |\mathbf{1}\rangle \otimes |\mathbf{0}\rangle \end{aligned}$$

Als Schaltdiagramm hat diese Abbildung folgendes Aussehen:

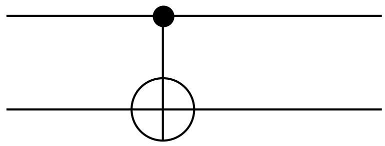

Abbildung 3.2: Eine XOR-Operation auf zwei Qubits. Das Symbol $\oplus$ steht für das XOR und der vertikale Strich gibt an, daß der Zustand des ersten Qubits mit in die Berechnung einfliesst.

Als Matrix ergibt sich

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Mit Hilfe von mehreren XOR-Transformationen kann auch ein Vertauschen von Zuständen einzelner Qubits realisiert werden. Das folgende Diagramm beschreibt die Vertauschung: $|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle \rightarrow |\mathbf{b}\rangle \otimes |\mathbf{a}\rangle$.

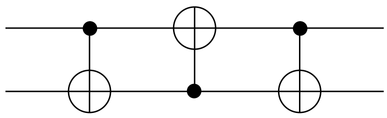

Abbildung 3.3: Vertauschen von zwei Zuständen.

Als Matrix ergibt sich hierfür:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3.1.3 Die Hadamard-Transformation

Eine sehr wichtige Transformation, die in jedem der später zu behandelnden Algorithmen eine zentrale Rolle spielt, ist die Hadamard-Transformation, die wir mit R bezeichnen. Sie operiert auf einem einzelnen Qubit und hat folgende Matrix Darstellung:

$$R = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Auf ein QuBit im Zustand $\psi_0 = |\mathbf{0}\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ angewandt liefert sie somit

$$\psi_1 = R\psi_0 = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle)$$

Eine Messung dieses Qubits bezüglich der Observablen

$$A = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{1}{2}P_{|\mathbf{0}\rangle} + \frac{-1}{2}P_{|\mathbf{1}\rangle}$$

liefert dann den Wert 0 mit einer Wahrscheinlichkeit von

$$P(M = 0|\psi_1) =< \psi_1, P_0\psi_1 >= \frac{1}{2}$$

Allgemein gilt für $a \in \{0, 1\}$:

$$R|\mathbf{a}\rangle = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + (-1)^a|\mathbf{1}\rangle) = \begin{cases} \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) & : \quad a = 0 \\ \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle) & : \quad a = 1 \end{cases}$$

Wenn nun bei einem System aus n Qubits im Zustand $|\mathbf{a}\rangle = |\mathbf{a_{n-1}}\rangle \otimes \cdots \otimes |\mathbf{a_0}\rangle$ auf jedes Qubit die Operation R angewandt wird, liefert dies:

$$\begin{aligned} R^n|\mathbf{a}\rangle &= R|\mathbf{a_{n-1}}\rangle \otimes \cdots \otimes R|\mathbf{a_0}\rangle && (3.1) \\ &= (\frac{1}{\sqrt{2}}|\mathbf{0}\rangle + (-1)^{a_{n-1}}|\mathbf{1}\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + (-1)^{a_0}|\mathbf{1}\rangle) && (3.2) \\ &= 2^{-\frac{n}{2}} \sum_{u=0}^{2^n-1} (-1)^{(a,u)}|\mathbf{u}\rangle && (3.3) \end{aligned}$$

Wobei (a, u) das Skalarprodukt zweier Vektoren in $\mathbb{Z}_2^n$ darstellt, also $(a, u) = (\sum a_i u_i) \bmod 2$. Insbesondere liefert R^n, angewandt auf den Zustand $|\mathbf{0}\rangle$, eine gleichmäßige Verteilung aller möglichen Basiszustände der Qubits und genau dies wird in allen Algorithmen eine entscheidende Rolle spielen.

3.1.4 Die unitäre Transformation U_f

Bei vielen klassischen Algorithmen und somit auch bei Quantencomputern muß eine Funktion ausgewertet werden. Dazu sei

$$f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$$

gegeben. Auf einem klassischen Rechner würde man für einen Eingabewert x den Funktionswert $f(x)$ berechnen und speichern. Auf Quantencomputern sind nur unitäre Transformationen erlaubt, so daß wir nun die zu f passende unitäre

Transformation suchen müssen. Wir nehmen nun an, daß f bijektiv ist, dann wird durch

$$U_f : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$$
$$U_f(|\mathbf{b}\rangle) = |\mathbf{f}(\mathbf{b})\rangle$$

eine unitäre Transformation definiert, die der klassischen Berechnung von f entspricht. U_f ist nichts anderes als eine Permutation der Basisvektoren, bei der ein Basisvektor e_i auf $e_{f(i)}$ abgebildet wird. Durch U_f können wir nun die Funktion f nicht nur an einer Stelle auswerten, sondern an mehreren gleichzeitig.

Ein Problem gibt es allerdings, wenn f nicht bijektiv ist, denn dann ist auch die Transformation U_f nicht mehr invertierbar und insbesondere nicht mehr unitär. Sei also nun allgemeiner

$$f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$$

nicht notwendigerweise bijektiv; in solchen Fällen kann man sich mit folgender Variante helfen:

$$U_f : \mathbb{C}^{2^n} \otimes \mathbb{C}^{2^m} \to \mathbb{C}^{2^n} \otimes \mathbb{C}^{2^m}$$
$$U_f(|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle) = |\mathbf{a}\rangle \otimes (|\mathbf{f}(\mathbf{a}) + \mathbf{b}\rangle$$

Dies ist sicher eine umkehrbare Operation, und da hier Basisvektoren nur vertauscht werden, sogar eine unitäre.

Somit ist es theoretisch immer möglich, eine Funktion f wie oben als unitäre Transformation aufzufassen, allerdings kann diese Transformation beliebig kompliziert und undurchführbar werden, so daß für jede Funktion f zusätzliche Überlegungen eine durchführbare Umsetzung von U_f liefern müssen. Einige spezielle Funktionen f, nämlich "NICHT" und "XOR", wurden schon als unitäre Transformationen umgesetzt, und dies soll im Abschnitt 3.2 mit weiteren häufig genutzten Funktionen geschehen.

3.1.5 Ein universeller Werkzeugkasten

Wenn ein Quantencomputer realisiert werden sollte, so wird man nicht für jede nötige Transformation eine neue Apparatur bauen, sondern es ist vielmehr nötig, mit einem festen Satz von Transformationen jede weitere zu konstruieren oder wenigstens jede weitere Transformation beliebig gut durch gegebene Transformation zu approximieren. Bei klassischen Rechnern kann zum Beispiel das sogenannte Toffoli-Gatter als universelle Transformation eingesetzt werden. Dies ist eine Operation auf 3 Bits, die wie folgt definiert ist:

$$T : \mathbb{Z}_2^3 \to \mathbb{Z}_2^3$$

$$T(a, b, c) = (a, b, ab + c)$$

Das logische UND auf a und b läßt sich mit Hilfe der Toffoli Operation umsetzen, indem einfach $c = 0$ gesetzt wird, für ein logisches NICHT auf c werden einfach $a = b = 1$ gesetzt, und da sich jede Operation mit Hilfe von UND und NICHT darstellen läßt, zeigt sich, daß dies auch für die Toffoli-Operation gilt.

Diese Operation reicht für Quantencomputer sicherlich nicht aus, denn zusätzlich zu den logischen Gattern gibt es hier noch unitäre Operationen, die darüber hinaus gehen. Hier ist es auch nicht möglich, daß wir mit einer endlichen Anzahl von Transformationen jede Transformation umsetzen können; vielmehr müssen wir uns damit zufrieden geben, durch geschickt gewählte Transformationen eine dichte Teilmenge von $U(n)$ zu erzeugen.

Deutsch war der erste, der eine solche Menge darstellte, und diese soll hier beschrieben werden. Hierzu verallgemeinerte Deutsch die Toffoli-Operation zu folgender unitären Transformation:

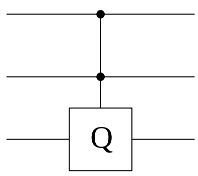

Abbildung 3.4: Das verallgemeinerte Toffoli Gatter.

Das kontrollierte Nicht (XOR) wird durch die unitäre Operation Q ersetzt, die auf einem einzelnen Qubit operiert. Dies ergibt für $a \in \{0, 1\}$ die Abbildung

$$\begin{array}{ccc}
|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{a}\rangle & \to & |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{a}\rangle \\
|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle \otimes |\mathbf{a}\rangle & \to & |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle \otimes |\mathbf{a}\rangle \\
|\mathbf{1}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{a}\rangle & \to & |\mathbf{1}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{a}\rangle \\
|\mathbf{1}\rangle \otimes |\mathbf{1}\rangle \otimes |\mathbf{a}\rangle & \to & |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle \otimes Q|\mathbf{a}\rangle
\end{array}$$

Weiter definieren wir folgende Abbildungen

$$U = \begin{pmatrix} \cos(2\pi\alpha) & \sin(2\pi\alpha) \\ -\sin(2\pi\alpha) & \cos(2\pi\alpha) \end{pmatrix}, W = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i\alpha} \end{pmatrix}$$

U_3 und W_3 bezeichnen das verallgemeinerte Toffoli Gatter mit U bzw. W anstelle von Q. Dann gilt folgendes Lemma:

Lemma 3.1.1 *Für geeignetes $\alpha \in [0, 1[$ erzeugen U_3 und W_3 eine dichte Teilmenge von $U(n)$ und dienen damit als universelle Transformationen.*

□

Beweis: siehe [Ah98]

3.2 Komplexere Operationen

Aufbauend auf den vorangegangenen einfachen Operationen soll nun erläutert werden, wie komplexere Operationen gestaltet werden können. Bei der Erläuterung und Darstellung der folgenden Transformationen wird auf eine Darstellung als Matrix verzichtet, da die Darstellung als Schaltskizze eine größere Einsicht vermittelt und die Darstellung als Matrix unmittelbar daraus hervorgeht. Bei der Darstellung als Matrix ist immer zu bedenken, daß die Dimension exponentiell mit der Anzahl der Qubits wächst und die Matrix so schnell unübersichtlich wird.

3.2.1 Addition

Als erste Operation wollen wir nun beschreiben, wie die Addition aus den Grundoperationen zusammengesetzt werden kann. Dazu seinen zwei Register der Länge n gegeben und ein zusätzliches Qubit, um eventuelle Überläufe zu speichern. Die gewünschte Operation ist also

$$A(|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle \otimes |\mathbf{c}\rangle) = |\mathbf{a}\rangle \otimes (|\mathbf{a}+\mathbf{b}\rangle)$$

Die Bedeutung des Qubits $|\mathbf{c}\rangle$ werden wir später genauer untersuchen.

Für diese Operation benötigen wir folgende Abbildung, die auf 3 Qubits operiert:

$$S(|\mathbf{x_1}\rangle \otimes |\mathbf{x_2}\rangle \otimes |\mathbf{x_3}\rangle) = |\mathbf{x_1}\rangle \otimes |\mathbf{x_2}\rangle \otimes |(\mathbf{x_1}+\mathbf{x_2}+\mathbf{x_3})\rangle$$

mit $|\mathbf{x_1}\rangle, |\mathbf{x_2}\rangle, |\mathbf{x_3}\rangle \in \{|\mathbf{0}\rangle, |\mathbf{1}\rangle\}$, welche in Abbildung 3.5 dargestellt ist. Die Addition hierbei ist modulo 2 zu verstehen. Die Operation S ist zu sich selbst invers.

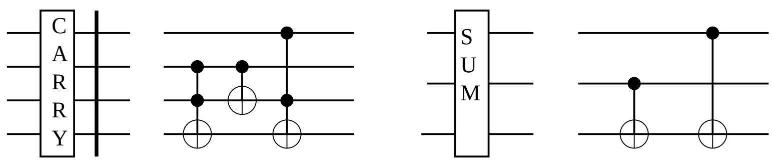

Abbildung 3.5: Diese Operationen sind die Basis Operationen, aus denen die Addition zusammengesetzt wird. Bei der Carry Operation gibt der senkrechte Strich die Richtung an, in der das Gatter abgearbeitet wird.

Weiter benötigen wir eine Operation, die einen Übertrag bei der Addition verarbeitet. Dies ist folgende Abbildung auf 4 Qubits:

$$C(|\mathbf{c_1}\rangle \otimes |\mathbf{a}\rangle \otimes |\mathbf{b}\rangle \otimes |\mathbf{c_2}\rangle) = |\mathbf{c_1}\rangle \otimes |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b}\rangle \otimes |\mathbf{c_2}+\mathbf{ab}+\mathbf{ac_1}+\mathbf{bc_1}\rangle$$

mit $|\mathbf{c_1}\rangle, |\mathbf{a}\rangle, |\mathbf{b}\rangle, |\mathbf{c_2}\rangle \in \{|\mathbf{0}\rangle, |\mathbf{1}\rangle\}$. Bei der Additionsoperation wird c_1 den Überlauf aus der vorhergehenden Stelle anzeigen und c_2 das Überlauf-Bit der aktuellen Stelle sein. c_2 muß also immer dann auf eins gesetzt werden, wenn mindestens zwei der Bits c_1,a oder b den Wert eins haben, und genau das leistet obige Abbildung. Sie ist in Abbildung 3.5 dargestellt.

Für die Addition benötigen wir auch die inverse Abbildung zu C. Dies ist die Abbildung

$$C^{-1}(|\mathbf{a_0}\rangle \otimes |\mathbf{a_1}\rangle \otimes |\mathbf{a_2}\rangle \otimes |\mathbf{a_3}\rangle) = |\mathbf{a_0}\rangle \otimes |\mathbf{a_1}\rangle \otimes |\mathbf{a_2} + \mathbf{a_1}\rangle \otimes |\mathbf{a_3} + \mathbf{a_0}\mathbf{a_2} + \mathbf{a_1}\mathbf{a_2} + \mathbf{a_1}\rangle$$

Im Diagramm wird die Operation C^{-1} durch einen senkrechten Strich links der Carry-Abbildung dargestellt.

Aus diesen Operationen können wir nun die Additionsroutine zusammensetzten. Für die Addition benötigen wir zusätzliche ein Register der Länge n, um die Überläufe zwischen zu speichern. Als Eingabe für die Operation dienen also eine Zahl $a = \sum_{i=0}^{n-1} a_i 2^i$ und eine Zahl $b = \sum_{i=0}^{n-1} b_i 2^i$.

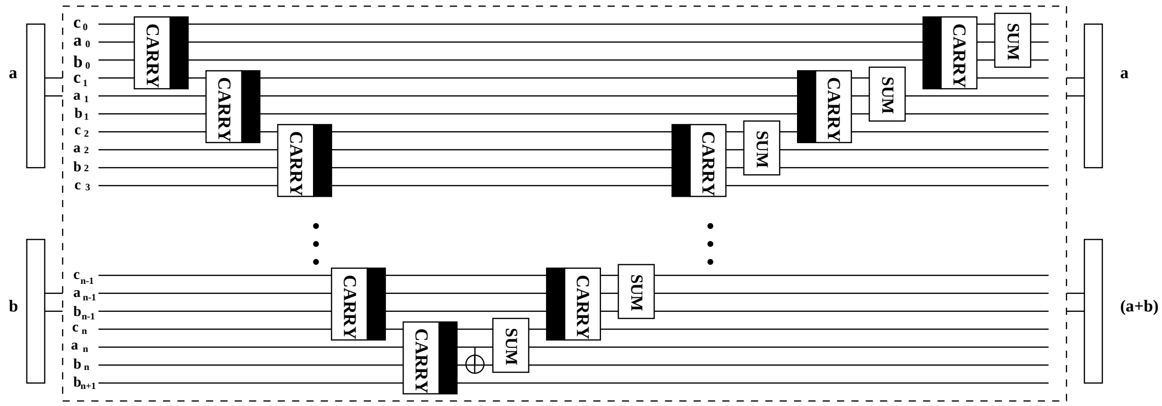

Abbildung 3.6: Das Additionsnetzwerk. Als erstes wird durch die Carry-Operationen das höchstwertige Bit berechnet, daraufhin alle weiteren. Wichtig zu bemerken ist, daß das höchste Überlaufbit nicht in das Zusatzregister geschrieben wird, sondern direkt nach c. Der senkrechte Strich an den Carry-Operationen gibt an, in welcher Richtung diese durchlaufen werden. Ein Strich auf der rechten Seite bezeichnet so die Abbildung C und ein Strich auf der linken Seite die Operation C^{-1}

Die Additionsoperation ist in Abbildung 3.6 dargestellt und soll nun erläutert werden. Die Qubits c_i sind anfangs alle im Zustand $|\mathbf{0}\rangle$. Die ersten "CARRY"-Operationen verändern die Zustande der a_i nicht. Sie berechnen allerdings die c_i, also die Überlaufbits, und setzen b_i auf den Wert $a_i + b_i$. Insbesondere ist damit schon das höchste Bit der Summe berechnet und in c gespeichert. Die folgende XOR Operation bringt b_{n-1} wieder in den Ausgangszustand, und die SUM Operation setzt nun $b_{n-1} = a_{n-1} + b_{n-1} + c_{n-1}$ und berechnet so das zweithöchste

Bit der Summe. Jetzt folgt ein C^{-1} und diese setzt damit c_{n-1} und b_{n-2} auf die Ausgangswerte, so daß die folgende SUM-Operation das nächste Bit berechnen kann. Dies wird nun bis zum letzten Bit fortgesetzt und die Summe wird somit berechnet. Der Vorteil dieser auf den ersten Blick unnötig komplizierten Vorgehensweise ist, daß die c_i wieder auf den Anfangswert, also normalerweise wieder auf 0 gesetzt werden. Damit ist es möglich, die Additionsoperation mehrmals hintereinander auszuführen.

Es stellt sich nun die Frage, was passiert, wenn das Überlaufqubit c am Beginn nicht den Wert 0, sondern den Wert 1 hat. Da dieser Wert erst bei Berechnung des höchstwertigen Qubits in das Ergeniss eingeht, bleiben die ersten n Qubits unverändert, das höchste Qubit wird "gekippt", und das führt zu folgenden drei Fällen:

$$\begin{aligned} A(|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle \otimes |\mathbf{0}\rangle) &= |\mathbf{a}\rangle \otimes (|\mathbf{a}+\mathbf{b}\rangle) \\ A(|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle \otimes |\mathbf{1}\rangle) &= \begin{cases} |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b}-\mathbf{2^n}\rangle & : \ a+b \geq 2^n \\ |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b}+\mathbf{2^n}\rangle & : \ a+b \leq 2^n \end{cases} \end{aligned}$$

Die Additionsoperation ist unitär, da sie aus einfachen unitären Operationen zusammengesetzt ist. Wenn die Additionsoperation umgekehrt ausgeführt wird, ergibt sich so einen Subtrahierer mit einer n und einer $n+1$ Bit Zahl als Eingeabewert. In Analogie zu den obigen drei Fällen erhält man nun folgende Abbildung:

$$A^{-1}(|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle) = \begin{cases} |\mathbf{a}\rangle \otimes |\mathbf{b}-\mathbf{a}+\mathbf{2^n}\rangle \otimes |\mathbf{1}\rangle & : \ a \geq b \\ |\mathbf{a}\rangle \otimes |\mathbf{b}-\mathbf{a}-\mathbf{2^n}\rangle \otimes |\mathbf{1}\rangle & : \ b-a \geq 2^n \\ |\mathbf{a}\rangle \otimes |\mathbf{b}-\mathbf{a}\rangle \otimes |\mathbf{0}\rangle & : \ a \leq b \text{ und } b-a \leq 2^n \end{cases}$$

Wenn wir zusätzlich vorrausetzen, daß auch $b < 2^n$ ist, so kann der zweite Fall nicht auftreten. In diesem Fall zeigt das Qubit des dritten Registers an, ob $a \leq b$ ist oder nicht, und diese Tatsache können wir im nächsten Abschnitt nutzen.

3.2.2 Addition modulo N

Als nächstes soll die Addition modulo einer Zahl N realisiert werden. Also die Abbildung

$$A_N(|\mathbf{a}\rangle \otimes |\mathbf{b}\rangle) = |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b} \bmod \mathbf{N}\rangle$$

Diese Abbildung wird mit Hilfe der "normalen" Addition aus dem vorangegangenen Abschnitt umgesetzt.

Wir nehmen als Eingabe N kleiner als 2^n und $0 \leq a, b < N$. Hierbei seien sowohl a,b als auch N in je einem Register der Länge n gespeichert. Das höchstwertige Bit bei der Addition bezeichnen wir mit s. Zusätzlich benötigen wir noch

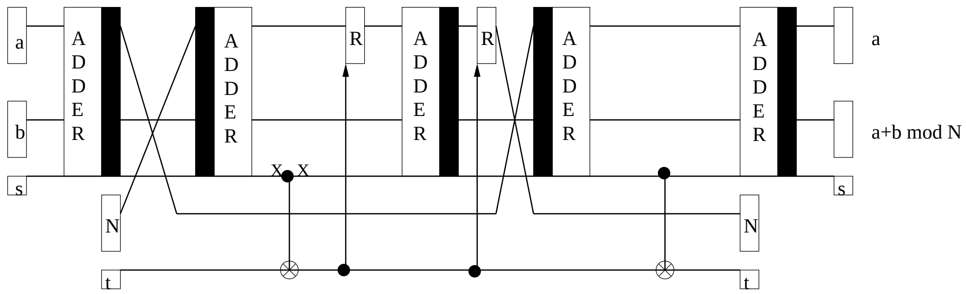

Abbildung 3.7: Addition modulo N. Auch hier zeigt ein senkrechter Strich wieder an, in welcher Richtung die “normale” Addition durchlaufen werden soll.

ein Kontrollbit, welches wir mit t kennzeichnen und welches anfänglich auf den Wert 0 gesetzt wird. Als Eingabe für die Operation dient also insgesamt:

$$\psi_0 = |\mathbf{a}\rangle \otimes |\mathbf{b}\rangle \otimes |\mathbf{s}\rangle \otimes |\mathbf{N}\rangle \otimes |\mathbf{t}\rangle$$

wobei am Anfang $s = t = 0$.

Die Behauptung ist nun, daß die Abbildung 3.7 die Abbildung A_N liefert mit

$$A_N\psi_0 = |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b} \bmod \mathbf{N}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{N}\rangle \otimes |\mathbf{0}\rangle$$

Auch hier ist es wieder für ein erneutes Durchlaufen der Operation wichtig, daß die Hilfsbits auf ihren Anfangswert zurückgesetzt werden. Die erste Additionsroutine liefert den Wert

$$\psi_1 = |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b}\rangle \otimes |\mathbf{N}\rangle \otimes |\mathbf{0}\rangle$$

wobei $|\mathbf{a}+\mathbf{b}\rangle$ wie üblich in einem $n+1$ Qubit-Register gespeichert wurde.

Als nächstes wird das Inverse der Addition angewandt, mit N als erstem Eingabewert und $a+b$ als zweitem. Dies führt zu dem Zustand

$$\psi_2 = \begin{cases} |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b}-\mathbf{N}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{N}\rangle \otimes |\mathbf{0}\rangle & : \quad N \leq a+b \\ |\mathbf{a}\rangle \otimes |\mathbf{2^n}+\mathbf{a}+\mathbf{b}-\mathbf{N}\rangle \otimes |\mathbf{1}\rangle \otimes |\mathbf{N}\rangle \otimes |\mathbf{0}\rangle & : \quad \text{sonst} \end{cases}$$

wobei der Wert von s uns an zeigt, welcher der beiden Fälle vorliegt.

Im ersten Fall sind wir nun schon fertig, denn wir wissen sicher, daß $a+b \leq 2N$, und wenn nun $a+b \geq N$ gilt, müssen wir von $a+b$ nur noch N abziehen, um $a+b \bmod N$ zu berechenen, und genau das ist schon geschehen. Im zweiten Fall ist $a+b = a+b \bmod N$ und wir müssen die letzte Operation wieder rückgängig machen. Dies wird erreicht, indem abhänig vom Wert von s entweder 0 (erster Fall) oder N (zweiter Fall) auf das Ergebniss addiert wird.

Auf s wird zuerst eine *NOT*-Operation angewandt; dies ist im Bild mit X gekennzeichnet, und der Wert wird nach t addiert und damit kopiert. Das heißt $t = 1$ genau dann, wenn $N \leq a+b$. Um den Wert von s wieder herzustellen wird nun erneut eine NOT-Operation angewandt. Abhängig von dem Qubit t wird nun das erste Register der Addition R mit dem Wert N XOR verknüpft oder nicht. Ist $t = 0$, das heißt $N > a + b$ und somit $a + b = a + b \bmod N$, findet keine Verknüpfung statt und die folgende Addition macht die Subtraktion von N wieder rückgängig. Ist aber $t = 1$, so findet eine XOR-Verknüpfung von R und N statt. Diese bedeutet aber nichts anderes, als daß 0 als erster Eingabewert in die Addition eingeht, und diese somit keinen Einfluß hat. In jedem Fall liefert das den Zustand

$$\psi_3 = \begin{cases} |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b} \bmod \mathbf{N}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{N}\rangle \otimes |\mathbf{1}\rangle & : \quad N \leq a+b \\ |\mathbf{a}\rangle \otimes |\mathbf{a}+\mathbf{b} \bmod \mathbf{N}\rangle \otimes |\mathbf{0}\rangle \otimes |\mathbf{N}\rangle \otimes |\mathbf{0}\rangle & : \quad \text{sonst} \end{cases}$$

Das bedeutet, daß $a + b \bmod N$ erfolgreich berechnet wurde. Der Rest des Diagramms dient nun nur noch dazu, die Hilfsbits in ihre Anfangszustände zurück zu setzten, damit diese für weitere Operationen ohne Verschränkungenohne wieder zur Verfügung stehen. Schön bei der Addition modulo N ist auch, daß man sich nicht um Überläufe zu kümmern braucht, da das Ergebniss mit Sicherheit kleiner als N und damit auch kleiner als 2^n ist.

3.2.3 Multiplikation modulo N

Als nächstes soll die Multiplikation modulo N beschrieben werden, auf der dann weiter unten das Potenzieren aufbauen wird. Folgende Abbildung soll konstruiert werden:

$$M'_{N,a}(|\mathbf{x}\rangle \otimes |\mathbf{0}\rangle) = |\mathbf{x}\rangle \otimes |\mathbf{ax} \bmod \mathbf{N}\rangle$$

Eine wichtige Bemerkung hierbei ist, daß der Wert a nicht als Eingabe, sondern als Parameter in die Berechnung eingeht, und diese Tatsache werden wir weiter unten benötigen.

Die Operation kann mit Hilfe der Addition modulo N geschehen. Denn mit

$$x = \sum_{i=0}^{n-1} x_i 2^i$$

gilt

$$ax \bmod N = \sum_{i=0}^{n-1} x_i a 2^i \bmod N.$$

Wir können also die Multiplikation modulo N realisieren, indem wir der Reihe nach für $i \in \{0, ..., n-1\}$ die Werte $2^i a \bmod N$ aufsummieren, für die $x_i = 1$ gilt. Da a bei dieser Operation nur ein Parameter ist, können wir die Werte $2^i a \bmod N$ zuerst klassisch berechnen und dann in ein Quanten-Register schreiben. Wir benötigen also eine Teiloperation, die man als bedingte Addition bezeichnen kann.

Weiterhin soll die gesamte Multiplikation nur stattfinden, wenn ein weiteres Kontrollbit gesetzt ist, dies wird wichtig im Hinblick auf das Potenzieren. Wir wollen also folgende Operation realisieren:

$$M_{N,a}(|\mathbf{c}\rangle \otimes |\mathbf{x}\rangle \otimes |\mathbf{0}\rangle) = \begin{cases} |\mathbf{c}\rangle \otimes |\mathbf{x}\rangle \otimes |\mathbf{ax} \bmod \mathbf{N}\rangle & : \quad c = 1 \\ |\mathbf{c}\rangle \otimes |\mathbf{x}\rangle \otimes |\mathbf{x}\rangle & : \quad \text{c=0} \end{cases}$$

Dies ist in Abbildung 3.8 dargestellt.

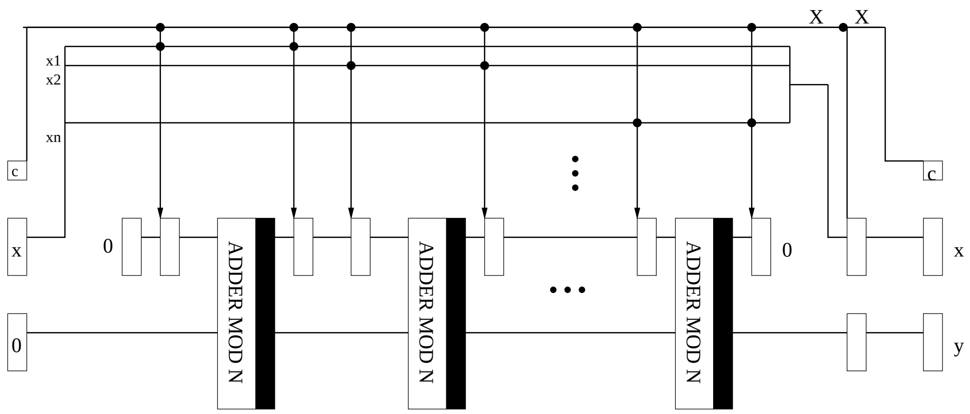

Abbildung 3.8: Multiplikation modulo N mit einem Kontrollbit. Für diese Operation wird jeweils abhängig von dem Kontrollbit und dem Wert von x_i $2^i a$ zu dem Ergebnis addiert. Falls $c = 0$ war, wird als letztes noch der Wert des ersten Registers in das zweite kopiert.

Das erste Register für die Addition wird hierbei im i.ten Schritt mit dem Wert $2^i a$ beschrieben, wenn sowohl x_i also auch c den Wert eins haben. Dies geschieht mit Hilfe des Toffoli-Gatters. Nach jeder Addition wird durch dasselbe Gatter das erste Register wieder in den Ausgangszustand zurück gesetzt, um anschließend mit dem nächsten Wert beschrieben zu werden. Wenn $c = 0$ war, ist der Zustand nach den Additionen $|\mathbf{0}\rangle \otimes |\mathbf{x}\rangle \otimes |\mathbf{0}\rangle$; da wir aber $|\mathbf{0}\rangle \otimes |\mathbf{x}\rangle \otimes |\mathbf{x}\rangle$ als Endzustand erhalten wollen, wird in diesem Fall der Wert x in das letzte Register kopiert. Dies kann mit einer Reihe von Toffoli Gattern erreicht werden.

Nun kommen wir zu einer für Shors Faktorisierungs-Algorithmus entscheidenden Operation.

3.2.4 Potenzieren mod N

Eine entscheidende Operation für Shors Algorithmus ist das Potenzieren in $\mathbb{Z}_N$, also die Funktion

$$f_a(x) : \mathbb{Z}_N \to \mathbb{Z}_N$$

$$f_a(x) = a^x \bmod N$$

und diese Funktion soll nun als unitäre Operation auf einem Quantencomputer konstruiert werden. Hier wird nun die Bedeutung des zusätzlichen Kontrollbits deutlich werden, welches bei der Multiplikation eingeflossen ist. Mit $x = \sum_{i=0}^{n-1} x_i 2^i$ ergibt sich

$$a^x = a^{\sum_{i=0}^{n-1} x_i 2^i} = a^{x_0} a^{2x_1} \cdots a^{2^{n-1} x_{n-1}}$$

und dies läßt sich berechnen, indem wir zu einem Anfangswert 1 jeweils abhängig von x_i den Wert a^{2^i} multiplizieren. Da bei einem nicht gesetzten Kontrollbit der Wert des ersten Registers in das zweite kopiert wurde, steht nach jeder dieser Multiplikationen unabhängig von x_i ein Wert der Form

$$a^{x_0} a^{2x_1} \cdots a^{2^i x_i} \otimes a^{x_0} a^{2x_1} \cdots a^{2^{i+1} x_{i+1}}$$

in den Registern. Anschließend müssen die beiden Werte vertauscht werden und das zweite Register wieder auf 0 gesetzt werden. Hierfür wird das Inverse der Multiplikation mit $\left(a^{2^{i+1} x_{i+1}}\right)^{-1} \bmod N$ genutzt. Sei nämlich folgender Zustand gegeben

$$\psi_0 = c \otimes (cd \bmod N).$$

Hier werden die Werte nun vertauscht

$$\psi_1 = S\psi_0 = (cd \bmod N) \otimes c$$

und abschließend wird das Inverse der Multiplikation mit $d^{-1} \bmod N$ angewandt. Dies ergibt

$$\psi_2 = M^{-1}_{N, d^{-1} \bmod N}(\psi_1) = (cd \bmod N) \otimes 0$$

und damit das gewünschte Ergebnis.

Insgesamt erhalten wir am Ende $a^x \bmod N$.

Komplexitätsberechnungen

Da, wie weiter unten erläutert wird, das Potenzieren modulo N wesentlich für das Faktorisieren ist, wollen wir nun die Komplexität in Abhängigkeit der Länge der Eingaben dieser Operation betrachten. Sei also als die Länge der Eingabe n Qubits, dann benötigen wir n Multiplikationen, die wiederum n Additionen ausführen. Jede dieser Additionen benötigt n elementare Operationen, so daß sich insgesamt ein Aufwand von $O(n^3)$ ergibt. Um den Aufwand noch genauer zu bestimmen, wäre es nötig festzulegen, welche Operationen man als elementar betrachtet, dies kann aber je nach tatsächlicher Umsetzung eines Quantencomputers sehr unterschiedlich sein. Dies liefert dann den exakten Aufwand, der sich von einer anderen Umsetzung allerdings nur um einen konstanten Faktor unterscheidet.

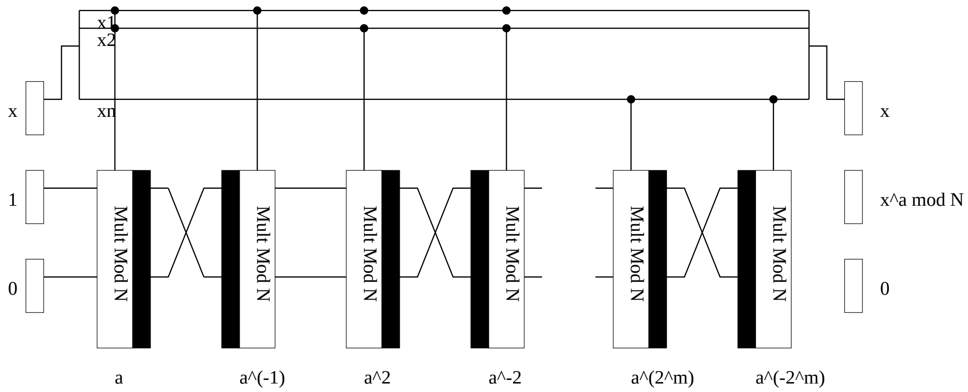

Abbildung 3.9: Potenzierung modulo N. Hier wird jeweils abhängig von x_i der Wert a^{2^i} mit dem Ergebnis multipliziert.

Wichtig bei der Betrachtung der Komplexität ist weiter der benötigte Speicheraufwand. Der Addierer benötigt bei einer Eingabe von n Bits noch zusätzlich $n-1$ Bits, um die Überträge zu berechnen. Da diese nach der Addition wieder auf den Ausgangswert gesetzt werden, können sie für weitere Additionen wieder verwendet werden. Weiter gibt es noch ein zusätzliches Kontrollbit bei der Addition modulo N sowie noch n weitere temporäre Qubits, insgesamt also $2n$ zusätzliche Qubits. Bei der Multiplikation werden wiederum n Qubits temporär genutzt und die Potenzierung benötigt schließlich 3 Eingaben von je n Qubits. Dies ergibt insgesamt einen Speicheraufwand von $6n$ Bits. Bei Shors Algorithmus benötigt man, um eine k Bit lange Zahl zu faktorisieren, $n = 2k$ Qubits, und damit mit Hilfe dieser Umsetzung $12n$ Qubits insgesamt.

Die hier vorgestellten Lösungen sind keineswegs optimal, so können noch weitere Verbesserungen erzielt werden wie in [Ve96] vorgeschlagen. Weitere Informationen zur Optimierung finden sich auch in [Be96].

3.2.5 Quanten-Fourier-Transformation

Für viele Algorithmen wird die Quanten-Fourier-Transformation eine entscheidende Rolle spielen. In diesem Abschnitt soll es dabei nur um folgende Abbildung gehen:

$$F_n : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$$

wobei ein Basisvektor $|\mathbf{k}\rangle \in \mathbb{C}^{2^n}$ abgebildet wird auf

$$F_n(|\mathbf{k}\rangle) = \frac{1}{\sqrt{2^n}} \sum_{u=0}^{2^n-1} \exp\left(\frac{2\pi iuk}{2^n}\right) |\mathbf{u}\rangle$$

Als erstes wollen wir zeigen, daß das Bild jedes Basisvektors ein Tensorprodukt von Teilzuständen ist. Dazu betrachten wir für $k, u \in \{0, ..., 2^n - 1\}$ die

Binärdarstellung $k = \sum_{i=0}^{n-1} k_i 2^i$ und $u = \sum_{i=0}^{n-1} u_i 2^i$. Für die Berechnung von $\exp(2\pi iuk2^{-n})$ reicht es, $2^{-n}uk \bmod 1$ zu bestimmen.

$$\begin{aligned} 2^{-n}uk &\equiv u_{n-1}(0,k_0) + u_{n-2}(0,k_1k_0) \\ &+ ... + u_0(0,k_{n-1}k_{n-2}...k_0) \pmod 1 \end{aligned} \tag{3.4}$$

wobei die Faktoren in Klammern jeweils die Binärdarstellung der Zahlen darstellen. Damit ist dann aber

$$\begin{aligned} F_n(|\mathbf{k}\rangle) &= \frac{1}{\sqrt{2^n}} \sum_{u=0}^{2^n-1} \exp\left(\frac{2\pi iuk}{2^n}\right) |\mathbf{u}\rangle \\ &= \frac{1}{\sqrt{2^n}} (|\mathbf{0}\rangle + e^{2\pi i(0,k_0)}|\mathbf{1}\rangle) \otimes (|\mathbf{0}\rangle + e^{2\pi i(0,k_1k_0)}|\mathbf{1}\rangle) \otimes (|\mathbf{0}\rangle + e^{2\pi i(0,k_2k_1k_0)}|\mathbf{1}\rangle) \\ &\otimes \cdots \otimes (|\mathbf{0}\rangle + e^{2\pi i(0,k_{n-1}...k_0)}|\mathbf{1}\rangle) \end{aligned} \tag{3.5}$$

wie sich durch Ausmultiplizieren überprüfen läßt. An der Darstellung als Produkt läßt sich nun auch leicht sehen, daß F_n eine unitäre Abbildung ist. Für zwei Vektoren $|\mathbf{k}\rangle, |\mathbf{k}'\rangle \in \mathbb{C}^{2^n}$ gibt es, wenn $k \neq k'$, eine erste Stelle $j \in \{0, ..., n-1\}$ in der Binärdarstellung, an der sich beide unterscheiden. Dann ist aber

$$\begin{aligned} < (|\mathbf{0}\rangle + e^{2\pi i(0,k_j...k_0)}|\mathbf{1}\rangle), (|\mathbf{0}\rangle + e^{2\pi i(0,k_j'...k_0')}|\mathbf{1}\rangle) > &= 1 + \exp\left(2\pi i \frac{k_j - k_j'}{2}\right) \\ &= 1 + \exp(\pi i) \\ &= 0 \end{aligned}$$

und mit der Definition 2.4.1 ist damit $< F_n(k), F_n(k') >= 0$ für $k \neq k'$. Es bleibt also noch $||F_n(|\mathbf{k}\rangle)|| = 1$ zu zeigen.

$$\begin{aligned} ||F_n(|\mathbf{k}\rangle)|| &= \left(\frac{1}{\sqrt{2^n}} \sum_{u=0}^{2^n-1} \exp\left(\frac{2\pi iuk}{2^n}\right) |\mathbf{u}\rangle, \frac{1}{\sqrt{2^n}} \sum_{v=0}^{2^n-1} \exp\left(\frac{2\pi ivk}{2^n}\right) |\mathbf{v}\rangle \right) \\ &= \frac{1}{2^n} \sum_{u=0}^{2^n-1} \exp\left(\frac{2\pi ik}{2^n}(u-u)\right) \\ &= 1 \end{aligned}$$

Mit der Darstellung als Tensorprodukt ist auch einleuchtend, daß es eine Implementierung der Fourier-Transformation geben muß, die sich aus einfachen Transformationen zusammensetzt.

Hierfür brauchen wir nun wieder die schon bekannte Hadamard-Transformation R und folgende Abbildung

$$B(k) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi 2^{-k}} \end{pmatrix}$$

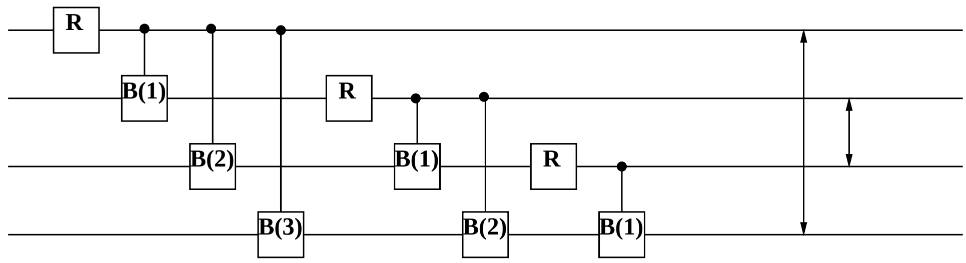

Abbildung 3.10: Die Quanten-Fourier-Transformation F_4.

Das Gatter in Abbildung 3.10 zeigt eine Implementierung von F_4.

Wegen der rekursiven Struktur des Gatters reicht es zu zeigen, daß die erste Anwendung von R und der anschließenden $B(i)$ das richtige Ergebnis liefert. Als erstes bemerken wir, daß

$$R(|\mathbf{k_{n-1}}\rangle) = |\mathbf{0}\rangle + (-1)^{k_{n-1}}|\mathbf{1}\rangle = |\mathbf{0}\rangle + \exp(i\pi k_{n-1})|\mathbf{1}\rangle$$

Die $B(i)$ werden jetzt, durch das erste Qubit kontrolliert, angewandt und liefern den Zustand

$$|\mathbf{0}\rangle + \alpha|\mathbf{1}\rangle$$

mit

$$\begin{aligned} \alpha &= \exp(\pi i k_{n-1}) \exp(\pi i k_{n-2} 2^{-1}) ... \exp(\pi i k_0 2^{n-1}) && (3.6) \\ &= \exp(2\pi i k 2^{-n}) && (3.7) \end{aligned}$$

und damit das gewünschte Ergebnis des letzten Qubits nach 3.5. Insgesamt sind also für die Umsetzung von F_n n Anwendungen von R und $n(n-1)/2$ Anwendungen von $B(i)$ nötig. Zusätzlich werden noch n Vertauschungs-Operationen genutzt, um die Qubits im Ergebnis richtig zu sortieren.

3.2.6 Berechnung von Eigenwerten

Ein weitere Operation, die bei einer zweiten Darstellung von Shors Algorithmus (siehe 4.3.6) eine entscheidende Rolle spielt soll noch besprochen werden. Wir nehmen an, es gebe zu einer unitären Operation U einen Eigenwert $e^{2\pi i\phi}$ mit $\phi \in [0, 1[$ und einen Eigenvektor ψ. Wir kennen weder ψ noch ϕ, und das Ziel ist es nun, eine Approximation von ϕ zu berechnen. Hierzu nehmen wir weiter an, daß wir auf den Zustand ψ zugreifen können, also daß dieser Zustand in so vielen Qubits wie nötig gespeichert ist, und wir auf diese Qubits die Operation U ausführen können. Weiter nehmen wir an, daß wir m Qubits gegeben haben, um den Eigenwert zu approximieren. Das Gatter in Abbildung 3.11 liefert eine Lösung hierfür.

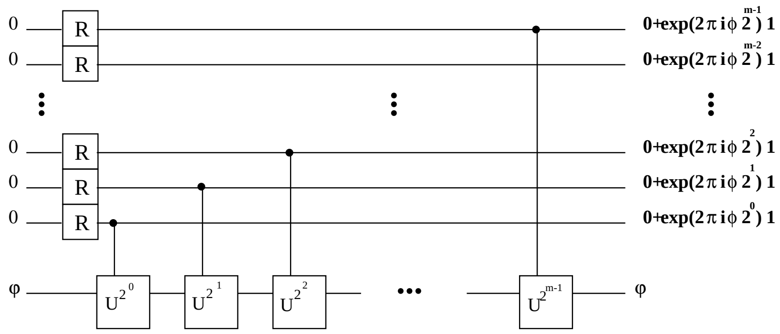

Abbildung 3.11: Ermittlung von Eigenwerten von U.

Hierbei wird durch das i.te Qubit kontrolliert die Operation U^{2^i} auf ψ angewandt. Wenn wir nur den Zustand des i.ten Qubits und des zweiten Registers betrachten, so erhalten wir also:

$$\begin{aligned}
\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) \otimes \psi &= \frac{1}{\sqrt{2}}|\mathbf{0}\rangle \otimes \psi + \frac{1}{\sqrt{2}}|\mathbf{1}\rangle \otimes \psi \\
&\to \frac{1}{\sqrt{2}}|\mathbf{0}\rangle \otimes \psi + \frac{1}{\sqrt{2}}|\mathbf{1}\rangle \otimes \left(e^{2\pi i\phi}\right)^{2^i} \psi \\
&= \frac{1}{\sqrt{2}}|\mathbf{0}\rangle \otimes \psi + \frac{1}{\sqrt{2}} e^{2\pi i 2^i \phi}|\mathbf{1}\rangle \otimes \psi \\
&= \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + e^{2\pi i\phi 2^i}|\mathbf{1}\rangle) \otimes \psi
\end{aligned}$$

Insgesamt liefert diese Transformation den Zustand

$$\begin{aligned}
\rho &= \frac{1}{\sqrt{2^m}}(|\mathbf{0}\rangle + e^{2\pi i\phi 2^{m-1}}|\mathbf{1}\rangle) \otimes (|\mathbf{0}\rangle + e^{2\pi i\phi 2^{m-2}}|\mathbf{1}\rangle) \otimes \ldots \otimes (|\mathbf{0}\rangle + e^{2\pi i\phi}|\mathbf{1}\rangle) \\
&= \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^n-1} \exp(2\pi i\phi y)|\mathbf{y}\rangle \qquad (3.8)
\end{aligned}$$

Wenn nun ϕ eine rationale Zahl ist, deren Binärdarstellung nach m Nachkommastellen abbricht, so liefert eine auf diesen Zustand angewandte Inverse der Fourier-Transformation den Wert ϕ. Denn

$$F_m^{-1}\left(\frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} \exp(2\pi i\phi y)|\mathbf{y}\rangle\right) = |\mathbf{2^m}\phi\rangle$$

folgt sofort auf der Definition von F_m, und wenn wir diesen Zustand messen, erhalten wir mit Wahrscheinlichkeit eins den Eigenwert als Ergebnis.

Wenn die Binärdarstellung von ϕ nun nicht nach m Nachkommastellen abbricht, so liefert das Inverse der Fourier-Transformation mit einer Wahrscheinlichkeit von mehr als $\frac{4}{\pi^2}$ die beste m-Bit Approximation für ϕ, wie im folgenden erläutert werden soll.

Zu ϕ gibt es ein a mit der Binärdarstelllung

$$a = \sum_{i=1}^{m} a_i 2^{m-i}$$

und $\phi = 2^{-m}a + \delta$ wobei $|\delta| \leq 2^{-(m+1)}$. Die inverse Fourier-Transformation, angewandt auf Zustand aus 3.8, liefert

$$\begin{aligned}
\psi &= \frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} \exp\left(\frac{-2\pi ixy}{2^m}\right) \exp(2\pi i\phi y)|\mathbf{x}\rangle && (3.9)\\
&= \frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} \exp\left(\frac{-2\pi ixy}{2^m}\right) \exp\left(2\pi i\left(\frac{a}{2^m} + \delta\right) y\right) |\mathbf{x}\rangle && (3.10)\\
&= \frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} \exp\left(\frac{2\pi i(a-x)y}{2^m}\right) \exp\left(2\pi i\delta y\right) |\mathbf{x}\rangle && (3.11)
\end{aligned}$$

Wenn dieser Zustand nun gemessen wird, so erhalten wir das Ergebnis a mit einer Wahrscheinlichkeit von

$$P(M = a|\psi) = \left|\frac{1}{2^m} \sum_{y=0}^{2^m-1} \exp\left(2\pi i\delta y\right)\right|^2$$

und dies ist eine geometrische Reihe und läßt sich dementsprechend schreiben als

$$P(M = a|\psi) = \left|\frac{1}{2^m} \frac{1 - \exp\left(2\pi i\delta 2^m\right)}{1 - \exp\left(2\pi i\delta\right)}\right|^2 \qquad (3.12)$$

Nun ist aber

$$|1 - \exp(it)|^2 = 4\sin^2\frac{t}{2}$$

und für $t \in [-\pi, \pi]$ ist

$$4\sin^2\frac{t}{2} \geq \frac{4t^2}{\pi^2}.$$

Dazu reicht es für $t \in [0, \pi]$ zu zeigen, daß

$$\sin\frac{t}{2} \geq \frac{1}{\pi}t$$

gilt. Aber $\sin\frac{t}{2}$ ist konkav auf $[0, \pi]$ und so liegt die Gerade durch 0 mit Steigung $\frac{1}{\pi}$ unterhalb von $\sin\frac{t}{2}$.

Bei einer Abschätzung nach oben nutzen wir, daß $\sin t$ auf $[0, \pi]$ unterhalb der Geraden durch 0 mit Steigung 1 liegt. Das bedeutet

$$\sin \frac{t}{2} \leq \frac{t}{2}$$

für $t \in [0, \pi]$. Also

$$\sin^2 \frac{t}{2} \leq \left(\frac{t}{2}\right)^2$$

für $t \in [-\pi, \pi]$, schließlich

$$|1 - \exp(it)|^2 = 4 \sin^2 \frac{t}{2} \leq t^2.$$

Damit läßt sich wegen $|2\pi i \delta 2^m| \leq \pi$ und $|2\pi i \delta| \leq \pi$ 3.12 abschätzen zu

$$P(M = a|\psi) \geq \left(\frac{1}{2^{-m}}\right)^2 \frac{4(2\pi\delta 2^m)^2}{\pi^2 (2\pi\delta)^2} \geq \frac{4}{\pi^2}.$$

Die Wahrscheinlichkeit, die beste m-Bit Approximation zu erhalten, ist somit größer als $0,4$, und durch wiederholtes Anwenden dieser Prozedur ist es möglich, die Approximation von ϕ mit beliebig großer Wahrscheinlichkeit zu berechnen.

Kapitel 4

Die Quantenalgorithmen

Mit Hilfe des in den vorangegangenen Kapiteln erarbeiteten Modells und den Operationen ist es nun möglich, Algorithmen zusammenzusetzen, die auf Quantencomputern abgearbeitet werden könnten. Hierbei steht die Frage im Vordergrund, ob es möglich ist, Probleme effizienter auf Quantencomputern zu lösen als es auf klassischen Rechnern möglich ist.

4.1 Der Algorithmus von D. Deutsch

Der erste Algorithmus, der den möglichen Vorteil von Quantencomputern aufzeigt, stammt von D. Deutsch [De85] und soll hier beschrieben werden. Das Problem, das hierbei gelöst wird, ist allerdings eher theoretischer Natur, zeigt aber, daß es Probleme gibt, die ein Quantencomputer schneller lösen kann als ein klassischer Rechner.

Für die Beschreibung des Problems ist folgender Begriff wichtig:

Definition 4.1.1 *Gegeben sei eine Funktion*

$$f : \mathbb{Z}_2^n \to \mathbb{Z}_2$$

f heißt balanciert, wenn die Anzahlen der Urbilder der 0 und der 1 gleich sind.

□

Gegeben sei nun eine beliebige Funktion $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ von der bekannt ist, daß sie entweder konstant oder balanciert ist. Das Problem ist es, mit Sicherheit festzustellen, ob sie konstant ist. Auf einem klassischen Rechner müßte die Funktion f im ungünstigsten Fall für $2^{n-1} + 1$ Werte berechnet werden, um das Problem mit Sicherheit zu lösen, d.h. die Laufzeit im "worst case" ist exponentiell in n. Deutsch gelang es nun, einen Algorithmus für Quantencomputer anzugeben, der mit nur zwei Auswertungen von f auskommt, um mit Sicherheit zu entscheiden zu welcher Klasse f gehört. Dies geschieht mit Hilfe des folgenden Algorithmus.

Algorithmus 4.1.2 *Für diesen Algorithmus werden 2 Register benötigt, eines mit n Qubits und eines mit einem Qubit.*

1. *Setze den Zustand des Quantencomputers auf* $|\mathbf{0}\rangle = |\mathbf{0}\rangle \otimes \cdots \otimes |\mathbf{0}\rangle$

2. *Wende die Hadamard-Transformation R^n aus 3.1.3 auf die ersten n Qubits an. Dies liefert den Zustand*

$$\psi_0 = R^n(|\mathbf{0}...\mathbf{0}\rangle) \otimes |\mathbf{0}\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\mathbf{k}\rangle \otimes |\mathbf{0}\rangle$$

also eine gleichmäßige Verteilung aller 2^n Basiszustände.

3. *Wende nun die Funktion f als unitäre Matrix an (siehe 3.1.4). Dies ergibt:*

$$\psi_1 = U_f(\psi_0) = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\mathbf{k}\rangle \otimes |\mathbf{f(k)}\rangle$$

*In gewisser Weise wurde mit diesem einen Schritt die Funktion f an allen Stellen **gleichzeitig** ausgewertet, und dies ist eine entscheidende Eigenschaft eines Quantencomputers.*

4. *Als nächster Schritt wird die unitäre Matrix T auf das letzte Qubit angewandt, wobei*

$$T = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Dies hat zur Folge, daß sich folgender Zustand ergibt:

$$\psi_2 = (I \otimes ... \otimes I \otimes T)(\psi_1) = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\mathbf{k}\rangle \otimes (-1)^{f(k)} |\mathbf{f(k)}\rangle$$

5. *Nun wird U_f erneut angewandt und es entsteht*

$$\psi_3 = U_f(\psi_2) = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\mathbf{k}\rangle \otimes (-1)^{f(k)} |\mathbf{0}\rangle$$

6. *Als letzter Schritt vor der Messung wird nun die Hadamard-Transformation erneut auf die ersten n Qubits angewandt:*

$$\begin{aligned} \psi_4 &= (R^n \otimes I)(\psi_3) \\ &= \frac{1}{2^n} \sum_{r=0}^{2^n-1} \sum_{k=0}^{2^n-1} (-1)^{f(k)} (-1)^{(r,k)} |\mathbf{r}\rangle \otimes |\mathbf{0}\rangle \\ &= \frac{1}{2^n} \sum_{r=0}^{2^n-1} |\mathbf{r}\rangle \otimes |\mathbf{0}\rangle \sum_{k=0}^{2^n-1} (-1)^{f(k)} (-1)^{(r,k)} \end{aligned}$$

7. *Nun erfolgt eine Messung des Quantencomputers im Zustand ψ_4. Angenommen, die Funktion f sei konstant, dann wird der Zustand $|\mathbf{0}\rangle = |\mathbf{0}\rangle \otimes ... \otimes |\mathbf{0}\rangle$ mit Wahrscheinlichkeit*

$$P(M = 0|\psi_4) = |\frac{1}{2^n} \sum_{k=0}^{2^n-1} (-1)^{f(0)}|^2 = 1$$

gemessen. Ist nun aber die Funktion f balanciert, dann ergibt die Summation über k für $r = 0$ $\sum_{k=0}^{2^n-1}(-1)^{f(k)} = 0$, und somit läßt sich mit dieser Messung mit Sicherheit unterscheiden, um was für eine Art von Funktion es sich gehandelt hat. □

Dieser Algorithmus liefert also bei einer zweifachen Anwendung von U_f unabhängig von n mit Sicherheit eine Aussage über die Art der Funktion f. Hier wird also schon deutlich, daß es möglich ist, Lösungen für Probleme auf Quantencomputern zu berechnen, die auf klassischen Rechnern deutlich länger bräuchten.

Das hier geschilderte Problem könnte ein klassischer Rechner allerdings auch für jede vorgegebene Fehlerwahrscheinlichkeit in von n unabhängiger konstanter Zeit lösen. Denn wenn auf einem klassischen Rechner die Funktion f an t zufällig gewählten verschiedenen Stellen ausgewertet wurde und für alle t Eingabewerte dasselbe Ergebnis, also entweder t Einser oder t Nullen, ermittelt wurde, dann ist die Wahrscheinlichkeit dafür, daß f balanciert ist nur noch 2^{-t}. Wenn sogar $t > 2^{n-1}$ gilt, dann kann man immer mit Sicherheit entscheiden, ob f balanciert oder konstant ist.

4.2 Simons Algorithmus

Der nächste Algorithmus ist auch deshalb interessant, weil er die Grundlagen für Shors Algorithmus zur Faktorisierung darstellt. In gewisser Weise sind beide Algorithmen sogar gleich und unterscheiden sich nur in der Gruppe, welche der Fourier-Transformation zugrunde liegt. Dies wird in [NC00]erläutert.

Simon [Si97] suchte nach einem Algorithmus, um die Periode einer Funktion f zu bestimmen:

Es sei $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ eine Funktion, wobei $m \geq n$, dann ist der Vektor $s \in \mathbb{Z}_2^n$ zu finden, so daß

$$f(x') = f(x) \Leftrightarrow x' = x \oplus s.$$

Oder es soll festgestellt werden, daß es kein solches s gibt.

Ein wichtiger Unterschied zu dem Algorithmus von Deutsch ist, daß Simons Algorithmus die Lösung nicht mit Sicherheit ermitteln kann, sondern nur mit einer (positiven) Wahrscheinlichkeit Erfolg hat. Damit kann es nötig sein, diesen Algorithmus wiederholt auszuführen.

Algorithmus 4.2.1 *Für diesen Algorithmus werden 2 Register mit einmal n und einmal m Bits benötigt.*

1. *Als erstes wird der Quantencomputer in den Zustand* $\psi_0 = |\mathbf{0}\rangle \otimes |\mathbf{0}\rangle$ *gesetzt.*

2. *Auf die ersten* n *Bits wird die Hadamard-Transformation angewandt.*

$$\psi_1 = (R^n \otimes I^m)(\psi_0) = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\mathbf{k}\rangle \otimes |\mathbf{0}\rangle$$

3. *Jetzt wird die unitäre Matrix* U_f *angewandt, und* f *damit auf allen Werten gleichzeitig berechnet.*

$$\psi_2 = U_f(\psi_1) = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\mathbf{k}\rangle \otimes |\mathbf{f(k)}\rangle$$

4. *Die Hadamard-Transformation wird erneut angewendet und liefert*

$$\begin{aligned}\psi_3 = (R^n \otimes I^m)(\psi_2) &= \frac{1}{2^n} \sum_{r=0}^{2^n-1} \sum_{k=0}^{2^n-1} (-1)^{(r,k)} |\mathbf{r}\rangle \otimes |\mathbf{f(k)}\rangle \\ &= \frac{1}{2^n} \sum_{r=0}^{2^n-1} |\mathbf{r}\rangle \otimes \sum_{k=0}^{2^n-1} (-1)^{(r,k)} |\mathbf{f(k)}\rangle\end{aligned}$$

und mit den Eigenschaften von f *läßt sich dies als Summe über die Nebenklassen von* $\{0, s\}$ *schreiben als*

$$\begin{aligned}\psi_3 &= \frac{1}{2^n} \sum_{r=0}^{2^n-1} |\mathbf{r}\rangle \otimes \sum_m (-1)^{(r,m)} |\mathbf{f(m)}\rangle + (-1)^{(r,m+s)} |\mathbf{f(m+s)}\rangle \\ &= \frac{1}{2^n} \sum_{r=0}^{2^n-1} |\mathbf{r}\rangle \otimes \sum_m (-1)^{(r,m)} (1 + (-1)^{(r,s)}) |\mathbf{f(m)}\rangle\end{aligned}$$

wobei m *ein Repräsentantensystem aller Nebenklassen von* $\{0, s\}$ *durchläuft.*

5. *Nun erfolgt die Messung der ersten* n *Bits. Hier betrachten wir nun zuerst den Fall, daß ein* r_1 *orthogonal zu dem gesuchten* s *liegt, d.h.* $(r_1, s) = 0$. *Die Wahrscheinlichkeit, daß ein solches* r_1 *gemessen wird, ist:*

$$P(M = r_1|\psi_3) = \sum_m |\frac{1}{2^n}(-1)^{(r_1,m)}(1 + (-1)^{(r_1,s)})|^2 = \left(\frac{1}{2^n}\right)^2 \sum_m 4 = \frac{1}{2^{n-1}}$$

Für ein r_2, *welches nicht orthogonal zu* s *liegt, also* $(r_2, s) = 1$, *ergibt sich für die Wahrscheinlichkeit:*

$$P(M = r_2|\psi_3) = \sum_m 0 = 0$$

Die Messung liefert also bei jedem Durchlauf des Algorithmus einen Vektor, der orthogonal zum gesuchten Vektor s steht. Da bei der Messung alle zu s orthogonalen Vektoren mit der gleichen Wahrscheinlichkeit auftreten, erhält man durch wiederholtes Anwenden des Algorithmus nach $O(n)$ Schritten $n-1$ linear unabhängige Vektoren, die alle orthogonal zu s liegen. Mit deren Hilfe läßt sich s nun klassisch berechnen. □

Zu diesem Algorithmus gibt es eine Verbesserung in [Br97], welche die Lösung mit Sicherheit nach n Schritten findet.

4.3 Shors Algorithmus zur Faktorisierung

Den wohl bekanntesten Algorithmus für Quantencomputer entwickelte Peter Shor [Sh97]. Der von Shor dargestellte Algorithmus ist in der Lage, große Zahlen schnell, d.h. mit logarithmischem Aufwand, in Primfaktoren zu zerlegen. Dies hat erhebliche Bedeutung für die Kryptographie, denn das am meisten eingesetzte Verfahren der Public-Key-Verschlüsselung, RSA, beruht auf der Annahme, daß es unmöglich ist, zu einer hinreichend großen Zahl $n = pq$ in effizienter Weise die Primfaktoren p und q zu bestimmen. Bevor der Algorithmus von Shor vorgestellt werden kann, benötigen wir zuerst einige Vorbemerkungen.

4.3.1 Einige Bemerkungen

Sei nun $n \in \mathbb{N}$ gegeben. Shors Algorithmus berechnet die Faktoren von n über die Ordnung eines geeigneten Elementes $y \in \mathbb{Z}_n^*$. Wenn wir die Ordnung r von y kennen, wissen wir

$$y^r = 1 \pmod{n},$$

und wenn r eine gerade Zahl ist, mit $r = 2k$, so gilt weiter

$$(y^k - 1)(y^k + 1) = 0 \pmod{n}.$$

Nehmen wir nun weiter an, daß $y^k \neq -1 \pmod{n}$, so können wir einen nicht trivialen Faktor von n bestimmen:

$$\gcd(y^k - 1, n) | n.$$

Wie wir gesehen haben, müssen wir an y Bedingungen stellen, damit es für die Faktorisierung geeignet ist. In 4.3.2 wird eine Möglichkeit vorgestellt, diese Bedingungen teilweise schon vorher abzufangen. Für den Moment können wir allerdings annehmen, daß in der Praxis einfach ein zufälliges Element aus $\mathbb{Z}_n^*$ gewählt wird, dessen Ordnung bestimmt wird, und daß,falls es nicht den Bedingungen genügt, der Vorgang mit einem neuen Element wiederholt wird. Um abzuschätzen, wie groß die Erfolgswahrscheinlichkeit dieser Methode ist, benötigen wir folgendes Lemma:

Lemma 4.3.1 *Sei $n = pq$ mit p, q prim gegeben. Dann erfüllen mindestens die Hälfte aller $y \in \mathbb{Z}_n^*$ die folgenden Bedingungen:*

1. *Die Ordnung r von y ist gerade, d.h. $\exists k$ mit $r = 2k$*

2. *Es gilt $y^k \neq -1 \pmod n$*

Für den Beweis benötigen wir folgendes Lemma:

Lemma 4.3.2 *Sei p prim mit $p-1 = 2^m x$, x ungerade und $b \in \mathbb{Z}_p^*$ gegeben. Weiter sei g einen Erzeuger von $\mathbb{Z}_p^*$*

1. *Die Ordnung t von b hat genau dann die Form $t = 2^k w$ mit $k \in \{1, ..., m\}$ und w ungerade, wenn $b = g^{2^{(m-k)}v}$ für ungerades v gilt. Insbesondere gibt es genau $2^{k-1}x$ Elemente dieser Form.*

2. *b hat eine ungerade Ordnung genau dann, wenn $b = g^{2^m w}$ mit $1 \leq w \leq x$ in $\mathbb{Z}_p^*$ gilt. Es gibt genau x Elemente mit ungerader Ordnung.*

Beweis:

1. Sei also g ein Erzeuger von $\mathbb{Z}_p^*$ und sei $b = g^s$. Die Ordnung von b bezeichnen wir mit t. Das ist gleichbedeutend mit $st = 0 \bmod (p-1)$ und t minimal. Das heißt,
$$t = \frac{p-1}{(p-1, s)}$$
Wenn nun t die Form $2^k w$ (w ungerade) haben soll, so muß 2^{m-k} s teilen, 2^{m-k+1} aber nicht, und dies beweist die Behauptung.

2. Die Ordnung t ist nun genau dann ungerade, wenn s von 2^m geteilt wird, und dies bedeutet, daß s von der Form $s = 2^m w$ mit $1 \leq w \leq x$ ist.

□

Beweis von 4.3.1: Wir bezeichnen die Ordnung von y in $\mathbb{Z}_p^*$ und $\mathbb{Z}_q^*$ mit $s = 2^i v$ und $t = 2^j w$, wobei v, w ungerade seien. Dann gilt $r = \operatorname{lcm}(t, s) = 2^{\max\{i,j\}} \operatorname{lcm}(v, w)$. Das heißt insbesondere, daß y nur dann ungerade Ordnung haben kann, wenn s und t ungerade sind. Wenn wir nun annehmen, das $r = 2k$ und $y^k = -1 \bmod n$ gilt, dann folgt $y^k = -1 \bmod q$ und $y^k = -1 \bmod p$. Das heißt aber insbesondere, daß weder s noch t den Wert k teilen darf (denn sonst wäre zum Beispiel $y^k = y^{cs} = 1 \bmod p$), und daraus folgt, daß $i = j$ sein muß. Nun betrachten wir den Anteil der Elemente, die bezüglich p die Bedingungen erfüllen. Sei dazu $\varphi(p) = p-1 = 2^m x$ und x ungerade. Nun können wir das Lemma 4.3.2 anwenden, und wir erhalten, daß nur x Elemente ungerade Ordnung haben. Weiter gibt es für jedes k genau $2^{k-1}x$ Elemente, die eine Ordnung der Form $2^k l$ mit ungeradem

l haben. Analoges gilt für $q-1=2^n z$. Für Elemente in $\mathbb{Z}_{pq}^*$ bedeutet das mit Obigem, daß die Anzahl der Elemente, die die Bedingungen nicht erfüllen, sich berechnet zu

$$xz + \sum_{k=1}^{min\{m,n\}} \left(2^{k-1}x\right)\left(2^{k-1}z\right) = xz\left(1 + \sum_{j=0}^{min\{m,n\}-1} 4^j\right)$$

Da $\varphi(pq) = (p-1)(q-1) = 2^{(m+n)}xz$ gilt, ist die Wahrscheinlichkeit, daß ein zufällig gewähltes y die Bedingungen nicht erfüllt, gegeben durch

$$2^{-(m+n)}\left(1 + \sum_{j=0}^{min\{m,n\}-1} 4^j\right) \leq \frac{1}{2}$$

Die Unabhängigkeit der Werte y mod p und y mod q folgt mit dem Chinesischen Restesatz. □

4.3.2 Vorberechnungen zu Shors Algorithmus

Es wird sich zeigen, daß wir schon im vorhinein ein y mit gerader Ordnung wählen können und sich so die Wahrscheinlichkeit, daß wir den Algorithmus mit einem weiteren Element durchlaufen müssen verkleinert. Das folgende Lemma soll bei der Prüfung der Bedingungen an y helfen

Lemma 4.3.3 *Sei eine Primzahl p und ein Nichtquadrat a in $\mathbb{Z}_p^*$ gegeben. Dann ist die Ordnung von a gerade.*

□

Beweis: Sei g eine Primitivwurzel in $\mathbb{Z}_p^*$ und $a = g^s$. Da a nun kein Quadrat ist, folgt, daß s ungerade ist, denn sonst wäre $s = 2t$ und weiter mit $b = g^t$ wäre $b^2 = a$ ein Widerspruch. Für die Ordnung von a gilt sicher $\operatorname{ord}_n(a)s = 0 \bmod p-1$, also $\operatorname{ord}_n(a)s = k(p-1)$, und da 2 beide Seiten der Gleichung aber nicht s teilt, folgt die Behauptung. □

Nun sei wieder $n = pq$, p und q prim. Ein Element y in $\mathbb{Z}_n^*$ hat gerade Ordnung immer dann, wenn y in $\mathbb{Z}_p^*$ oder in $\mathbb{Z}_q^*$ gerade Ordnung hat. Damit ergibt sich mit Hilfe des Jacobi-Symbols folgendes

Korollar 4.3.4 *Sei y in $\mathbb{Z}_n^*$ gegeben. Dann gilt*

$$\left(\frac{y}{n}\right) = -1 \Rightarrow \operatorname{ord}_n(y) = 2k$$

□

Da sich aber das Jacobi-Symbol effizient berechnen läßt, haben wir nun ein hinreichendes und brauchbares Kriterium dafür gefunden, ob ein Element gerade Ordnung hat.

Insbesondere gilt folgendes Lemma:

Lemma 4.3.5 *Die Wahrscheinlichkeit, daß ein zufälliges $y \in \mathbb{Z}_n^*$ mit $\left(\frac{y}{n}\right) = -1$ die beiden Bedingungen aus Lemma 4.3.1 erfüllt, ist größer als $\frac{3}{4}$.*

Beweis: Wir wollen die Elemente zählen, die die Bedingung nicht erfüllen. Wir suchen also $y \in \mathbb{Z}_n^*$ mit $\left(\frac{y}{n}\right) = -1$; wegen obigem Lemma können wir schließen, daß die Ordnung von y in $\mathbb{Z}_n^*$ gerade ist: $\operatorname{ord}_n(y) = 2k$. Weiter bezeichnen wir $\operatorname{ord}_p(y) = 2^i v$ und $\operatorname{ord}_q(y) = 2^j w$ mit ungeraden v, w. Es gilt insbesondere $2k = 2^{\max\{i,j\}} \operatorname{lcm}(v, w)$. Weiter sei also $y^k = -1 \bmod n \Leftrightarrow y^k = -1 \bmod p$ und $y^k = -1 \bmod q$. Dies ist aber gleichbedeutend mit $i = j$, wie wir bei dem Beweis von 4.3.1 gesehen haben.

$\left(\frac{y}{n}\right) = -1$ ist äquivalent dazu, daß entweder $\left(\frac{y}{p}\right) = -1$ und $\left(\frac{y}{q}\right) = 1$ oder $\left(\frac{y}{p}\right) = 1$ und $\left(\frac{y}{q}\right) = -1$ gilt. Wir betrachten zunächst den ersten Fall:

Sei $p - 1 = 2^{m_1}x_1$ und g_p ein Erzeuger von $\mathbb{Z}_p^*$, dann ist $\left(\frac{y}{p}\right) = -1$ gleichbedeutend damit, daß $y = g_p^{t_1}$ für ungerades t_1. Um die Elemente zu zählen, welche die Bedingungen aus 4.3.1 nicht erfüllen, können wir nun für festes i die Anzahl der Elemente berechnen und dann über alle möglichen i summieren. Die Frage ist also, wieviel ungerade t_1 gibt es, so daß $y = g_p^{t_1}$ eine Ordnung der Form $2^i v$ für ungerades v hat. Aus Lemma 4.3.2 folgt, daß nur für $i = m_1$ der Wert von t_1 ungerade gewählt werden kann. Dann ergeben aber alle möglichen ungeraden Werte zwischen 1 und $p - 1$ ein passendes Element.

Nun müssen wir dieselben Überlegungen bezüglich q betrachten. Sei hierzu $q - 1 = 2^{m_2}x_2$ und g_q ein Erzeuger von $\mathbb{Z}_q^*$. Gesucht sind hier also nun alle möglichen geraden Werte von t_2, so daß g^{t_2} eine Ordnung der Form $2^{m_1}w$ hat. Ist nun $m_1 > m_2$, so gibt es keine Lösungen, denn die Ordnung von g^{t_2} muß $q - 1 = 2^{m_2}x_2$ teilen. Gilt $m_1 = m_2$ so gibt es nur ungerade Lösungen für t_2 und somit auch hier keine passenden. Es bleibt also nur der Fall $m_1 < m_2$. Nach Lemma 4.3.2 sind die Lösungen genau die t_2 von der Form $2^{m_2 - m_1}u$ für ungerades u. Hier kann nun u alle ungeraden Werte von 1 bis $2^{m_1}x_2 - 1$ annehmen, und dies liefert genau $2^{m_1 - 1}x_2$ Lösungen.

Für den zweiten Fall, also $\left(\frac{y}{p}\right) = 1$ und $\left(\frac{y}{q}\right) = -1$ ergeben sich die gleichen Überlegungen, und wir können für den Fall, daß $m_1 \neq m_2$ O.E $m_1 < m_2$ annehmen.

Insgesamt ergibt sich also bei $m_1 \neq m_2$ für die Anzahl der Elemente, die die Bedingung nicht erfüllen:

$$A(n) = \frac{p-1}{2} 2^{(m_1-1)} x_2 = \frac{p-1}{2} \frac{2^{m_2} x_2}{2^{(m_2 - m_1 + 1)}}$$

$$\begin{aligned} &= \frac{1}{4}(p-1)(q-1)\frac{1}{2^{(m_2-m_1)}} \\ &= \frac{1}{4}\varphi(n)\frac{1}{2^{(m_2-m_1)}} \end{aligned}$$

Die Anzahl der Elemente mit $\left(\frac{y}{n}\right) = -1$ ist $\frac{1}{2}\varphi(n)$, und damit ergibt sich für die gesuchte Wahrscheinlichkeit

$$P(n) = 1 - \frac{1}{2^{(m_2-m_1+1)}} \geq \frac{3}{4}.$$

Für den Fall $m_1 = m_2$, ergibt sich die gesuchte Wahrscheinlichkeit sogar zu 1, das heißt, aus $\left(\frac{y}{n}\right) = -1$ folgen beide Bedingungen. □

4.3.3 Weitere Möglichkeiten, einen Faktor zu bestimmen

Bei der oben vorgeschlagenen Methode, mittels der Ordnung einen Faktor zu bestimmen, wird nach einer nicht-trivialen quadratischen Wurzel aus Eins gesucht. Dies ist nur eine andere Beschreibung der Bedinungen in 4.3.1. Statt einer nicht-trivialen quadratischen Wurzel können wir dasselbe auch mit höheren Wurzeln der Eins durchführen.

Sei dazu für $y \in \mathbb{Z}_n$ die Ordnung r gegeben. Wenn wir nun in der Lage sind, einen Primfaktor s von r zu finden, also $r = st$, können wir das Element $x = y^{r/t}$ bestimmen. Das Element x hat also die prime Ordnung s in $\mathbb{Z}_n$. Damit können wir wegen $s = \operatorname{lcm}(\operatorname{ord}_p(x), \operatorname{ord}_q(x))$ O.E.d.A. annehmen, daß $s = \operatorname{ord}_p(x)$ gilt. Für die Ordnung von x in $\mathbb{Z}_q$ bleiben damit nur die beiden Fälle $\operatorname{ord}_q(x) = s$ oder $\operatorname{ord}_q(x) = 1$. Der erste Fall ist der für unsere Zwecke ungünstige, der zweite aber ist geeignet, um einen Faktor von n zu finden. Es gilt nämlich hier

$$x \neq 1 \bmod p \text{ und } x = 1 \bmod q.$$

Damit ist aber

$$\gcd(x-1, n) = q$$

und somit ist n faktorisiert. Der Erfolg dieser Methode hängt mit der Möglichkeit zusammen, Primfaktoren von r abzuspalten. Dies kann in speziellen Fällen unmöglich sein, so zum Beispiel, wenn r aus zwei in etwa gleich großen Primzahlen besteht. In den meisen Fällen sollte es allerdings möglich sein, zumindest einige Faktoren von r zu bestimmen, und mit diesen obige Berechnungen durchzuführen.

Wir wollen nun noch genauer die Wahrscheinlichkeit dafür untersuchen, daß, wenn wir einen Primfaktor s gefunden haben, der für die Faktorisierung ungünstige Fall $x \neq 1 \bmod q$ eintritt.

Dazu nehmen wir an, daß $q-1$ eine zufällig gewählte gerade Zahl sei. Der Fall $x \neq 1 \bmod q$ kann nur dann auftreten, wenn $s^k | q-1$ für ein $k \in \mathbb{N}$, denn nur dann kann es ein Element der Ordnung s in $\mathbb{Z}_q^*$ geben. Unter der Annahme, daß $q-1$ eine zufällige Zahl ist, beträgt die Wahrscheinlichkeit hierfür s^{-k}. Dann hat aber ein zufällig gewähltes y mit Wahrscheinlichkeit $(s^k - 1)/s^k$ eine durch s teilbare Ordnung in $\mathbb{Z}_q^*$. Damit können wir die Wahrscheinlichkeit P für den ungünstigen Fall nach oben abschätzen:

$$\begin{aligned} P &= \sum_{k \in \mathbb{N}} \frac{1}{s^k} \left(\frac{s^k - 1}{s^k} \right) = \sum_{k \in \mathbb{N}} \left(\frac{1}{s} \right)^k - \sum_{k \in \mathbb{N}} \left(\frac{1}{s^2} \right)^k \\ &= \frac{s}{s-1} - \frac{s^2}{s^2-1} = \frac{s}{s^2-1} \\ &\approx \frac{1}{s} \end{aligned}$$

Die Wahrscheinlichkeit für den günstigen Fall ist also umso größer, je größer der Primfaktor ist, den wir von der Ordnung r abspalten können.

Damit sind auch solche y aus $\mathbb{Z}_n$ für eine Faktorisierung nützlich, für die die Bedinungen in 4.3.1 nicht zutreffen.

Dies verbessert somit weiter die Wahrscheinlichkeit, daß der Algorithmus auf dem Quantencomputer nur einmal durchlaufen werden muss.

4.3.4 Der Algorithmus

Damit können wir nun den Algorithmus beschreiben und die Erfolgswahrscheinlichkeit bestimmen.

Algorithmus 4.3.6 *Für den Algorithmus von Shor werden zwei Register benötigt. Die Länge t des ersten Registers wird so gewählt, daß $n^2 < 2^t < 2n^2$ gilt. Die Begründung hierfür findet sich im Anschluß an die Beschreibung des Algorithmus. Das zweite Register muß so gewählt sein, daß alle Werte in $\mathbb{Z}_n$ aufgenommen werden können. Das bedeutet, um eine s Bit lange Zahl zu faktorisieren, benötigt man ungefähr $2s$ Qubits im ersten und s Qubits im zweiten Register.*

1. *Wende die Hadamard-Transformation auf die ersten t Qubits an, um den Zustand*
$$\psi_0 = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |\mathbf{k}\rangle \otimes |\mathbf{0}\rangle$$
zu erzeugen.

2. *Berechne nun die Funktion $f(x) = y^x \bmod n$ mittels U_f; es ergibt sich*
$$\psi_1 = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |\mathbf{k}\rangle \otimes |\mathbf{f(k)}\rangle$$
Die Umsetzung von f als unitäre Transformation ist in 3.2.4 beschrieben. Bei der Analyse der Laufzeit ist dieser Schritt der entscheidende.

3. *Wende nun die Quanten-Fourier-Transformation auf die ersten t Qubits an. Dies ist in 3.2.5 beschrieben und liefert*

$$\psi_2 = \frac{1}{2^t}\sum_{k=0}^{2^t-1}\sum_{u=0}^{2^t-1}\exp\left(\frac{2\pi iuk}{2^t}\right)|\mathbf{u}\rangle\otimes|\mathbf{f}(\mathbf{k})\rangle$$

Diesen Zustand können wir nun umschreiben, indem wir ausnutzen, daß $f(x)=f(x+r)$ gilt, wenn r die Ordnung von y ist. Dazu setzen wir $X=[\frac{2^t-1}{r}]$ und schreiben $k=mr+j$. Wir erhalten so

$$\psi_2 = \frac{1}{2^t}\sum_{u=0}^{2^t-1}\sum_{m=0}^{X-1}\sum_{j=0}^{r-1}\exp\left(\frac{2\pi iu(mr+j)}{2^t}\right)|\mathbf{u}\rangle\otimes|\mathbf{f}(\mathbf{mr}+\mathbf{j})\rangle$$

wobei wir den kleinen Fehler ignorieren, wenn $mr+j\geq 2^t$ ist.

$$\psi_2 = \frac{1}{2^t}\sum_{u=0}^{2^t-1}\sum_{j=0}^{r-1}|\mathbf{u}\rangle\otimes\exp\left(\frac{2\pi iuj}{2^t}\right)|\mathbf{f}(\mathbf{j})\rangle\sum_{m=0}^{X-1}\exp\left(\frac{2\pi iumr}{2^t}\right)$$

Die Summe über m ist eine geometrische Reihe, und wir können sie deshalb noch vereinfachen zu

$$s_u = \frac{1-\exp\left(\frac{2\pi iuXr}{2^t}\right)}{1-\exp\left(\frac{2\pi iur}{2^t}\right)}.$$

Wenn wir nun die ersten t Qubits messen, erhalten wir das Ergebnis u mit einer Wahrscheinlichkeit von

$$<\psi_2, P_u\psi_2> = \frac{r}{2^{2t}}|s_u|^2$$

$\square$

Damit ist der Teil des Algorithmus der auf Quantencomputern ablaufen muß, komplett beschrieben. Es ist aber noch nicht klar, wie aus dem Ergebnis die Ordnung von y bestimmt werden kann. Dazu sind noch einige Überlegungen nötig, die nun erläutert werden sollen.

Angenommen wir messen ein u, für das folgende Ungleichung gilt

$$-\frac{r}{2}\leq ur-k2^t\leq\frac{r}{2}. \tag{4.1}$$

Dies können wir wegen der Wahl von t umformulieren zu

$$\left|\frac{u}{2^t}-\frac{k}{r}\right|\leq 2^{-(t+1)}\leq\frac{1}{2n^2}\leq\frac{1}{2r^2} \tag{4.2}$$

Dies bedeutet, daß der uns bekannte Bruch $\frac{u}{2^t}$ durch den unbekannten Bruch $\frac{k}{r}$ approximiert wird. Da aber $r\leq n$ gilt, kann es nur einen Bruch mit Nenner kleiner als n geben, der dies Erfüllt. Um den unbekannten Bruch $\frac{k}{r}$ zu berechnen, ist es also nötig, den bekannten Bruch bis auf die gegebene Genauigkeit durch eine Bruch mit Nenner kleiner n zu approximieren. Bei der Ermittlung von $\frac{k}{r}$ helfen uns nun die Kettenbrüche, wie weiter unten (4.3.5) erläutert wird.

Wenn nun k und r teilerfremd sind, liefert diese Methode den Wert von r, andernfalls nur ein Teiler von r. Für den Fall, daß nur ein Teiler von r ermittelt werden kann, kann der Algorithmus mit gleichem y erneut durchlaufen werden. Wenn man nach mehreren Durchläufen nur Teiler $t_1, ..., t_i$ von r erhält, so bietet es sich an, auch $\prod t_i$ als Ordnung von y zu testen.

Die Überprüfung, ob die Ordnung korrekt ermittelt wurde, ist schnell durchführbar. Hierzu muß nur $y^r = 1$ geprüft werden, und dies läßt sich in $O(\log r)$ Schritten berechnen.

Um die Erfolgswahrscheinlichkeit für die Bestimmung der Ordnung zu ermitteln, müssen wir noch die Wahrscheinlichkeit berechnen, daß ein gemessenes u die Bedingung 4.1 erfüllt. Dazu bemerken wir zuerst, daß

$$|1 - e^{i\theta}|^2 = (1 - e^{i\theta})(1 - e^{-i\theta}) = (ie^{i\frac{\theta}{2}} - ie^{-i\frac{\theta}{2}})^2 = 4\sin^2\frac{\theta}{2}$$

und damit ist

$$rs_u^2 = 2^{-2t} r \frac{\sin^2(\pi urX/2^t)}{\sin^2(\pi ur/2^t)}$$

Nun setzen wir $\tau = 2^{-t}ur - k$. Dann gilt sicher $|\tau| \leq \frac{1}{2}$ und weiter

$$\sin^2(\pi ur 2^{-t}) = \sin^2(\pi\tau) \leq (\pi\tau)^2$$

da $\sin(\theta) \leq \theta$ für $\theta \in [0, \frac{\pi}{2}]$. Für den Zähler ist die Abschätzung etwas aufwendiger

Wir schätzen nun $X\tau$ ab. Wegen der Voraussetzung gilt

$$|\tau| = |2^{-t}ur - k| \leq 2^{1-t}r$$

und damit

$$|X\tau| \leq 2^{1-t}rX$$

und aus der Definition von X folgt $X - 1 \leq 2^t/r \leq X$ und weiter $Xr2^{-t} \leq 1 + 1/(X-1)$. Damit ist

$$|X\tau| \leq \frac{1}{2}\left(1 + \frac{1}{X-1}\right)$$

und wegen $X \geq \frac{2^t}{r} \geq \frac{n^2}{r} \geq n$ folgt

$$|X\tau| \leq \frac{1}{2}\left(1 + \frac{1}{n-1}\right)$$

Wir betrachten nun die folgende Hilfsfunktion $g(x) := \frac{\sin(x)}{x}$. Diese Funktion ist auf dem Intervall $[0, \pi]$ monoton fallend, und so erhalten wir:

$$\begin{aligned}
\sin^2(\pi urX/2^t) &= \sin^2(\pi X\tau) \\
&= (\pi X\tau)^2 g^2(\pi X\tau) \\
&\geq (\pi X\tau)^2 g^2\left(\frac{\pi}{2}\left(1+\frac{1}{n-1}\right)\right)
\end{aligned}$$

Insgesamt können wir nun die Wahrscheinlichkeit abschätzen

$$\begin{aligned}
rs_u^2 &= 2^{-2t} r \frac{\sin^2(\pi urX/2^t)}{\sin^2(\pi ur/2^t)} \\
&\geq 2^{-2t} r \frac{(\pi X\tau)^2 g^2\left(\frac{\pi}{2}\left(1+\frac{1}{n-1}\right)\right)}{(\pi\tau)^2} \\
&= (2^{-2t} r X^2) \frac{4}{\pi^2}\left(\frac{n-1}{n}\right)^2 \sin^2\left(\frac{\pi}{2}\left(1+\frac{1}{n-1}\right)\right) \\
&\geq \frac{1}{r}\frac{4}{\pi^2}\left(\frac{n-1}{n}\right)^2
\end{aligned}$$

Jetzt gibt es aber zu jedem $k \in \{1, ..., r\}$ genau ein u, das die Bedingung 4.1 erfüllt, und somit gibt es insgesamt r Lösungen, die 4.1 erfüllen. Damit ist die Wahrscheinlichkeit, daß ein geeignetes u gemessen wird, insgesamt größer als

$$\frac{4}{\pi^2}\left(\frac{n-1}{n}\right)^2 \geq 0.4$$

Algorithmus 4.3.7 *Der gesamte Ablauf, um eine Zahl n zu faktorisieren, sieht also wie folgt aus:*

1. *Wähle zufällig eine Zahl $y \in \mathbb{Z}_n^*$.*
2. *Wende Algorithmus 4.3.6 an und messe ein u.*
3. *Wenn u nicht zu Berechnung der Ordung r hilfreich ist, gehe zu Schritt 2.*
4. *Wenn r ungerade oder $y^{\frac{r}{2}} = -1$ ist, gehe zu Schritt 1.*
5. *Berechne $\gcd(y^{\frac{r}{2}} - 1, n)$, um einen nicht trivialen Faktor von n zu bestimmen.* □

Die Erfolgswahrscheinlichkeit ist größer als:

$$\frac{1}{2}\frac{\varphi(r)}{r}0.4$$

4.3.5 Kettenbrüche

In diesem Abschnitt soll kurz erläutert werden, wie mit Hilfe der Kettenbruch-Methode der Wert $\frac{k}{r}$ aus 4.2 berechnet werden kann. Die Begründung hierfür liefert das folgende Lemma:

Lemma 4.3.8 *(Approximation mit Kettenbrüchen) Sei x eine rationale Zahl und $p, q \in \mathbb{N}$ mit $\gcd(p, q) = 1$, so daß*

$$|\frac{p}{q} - x| \leq \frac{1}{2q^2}$$

Dann ist $\frac{p}{q}$ ein Näherungsbruch der Kettenbruchentwicklung von x.

Der Beweis lehnt sich an den Beweis in [NC00] p. 637 an, dort sind allerdings einige kleine Ungenauigkeiten enthalten.

Für den Beweis benötigen wir einige allgemein bekannte Eigenschaften von Kettenbrüchen, die hier nur zitiert werden sollen. Für weitere Details siehe [Fo96].

Ein Kettenbruch hat die Gestalt

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}$$

hierfür schreiben wir kurz

$$x = \operatorname{cfrac}(a_0, ..., a_n)$$

Ein Kettenbruch heißt regulär, falls $a_i \in \mathbb{Z}$ für $i \in \{0, ...n\}$ und $a_i \geq 1$ für $i \in \{1, ...n\}$.

Jedes $x \in \mathbb{R}$ kann mit Hilfe regulärer Kettenbruche approximiert werden. Dazu geht man wie folgt vor:

Setze $a_0 = [x]$ und falls $a_0 = x$ bricht die Prozedur ab. Die Kettenbruchentwicklung ist in diesem Fall x selbst. Für $x \neq a_0$ setze $\xi_1 = \frac{1}{x - a_0} \geq 1$. Es gilt $x = \operatorname{cfrac}(a_0, \xi_1)$. Die gleiche Prozedur kann jetzt mit ξ_1 anstelle von x fortgesetzt werden. Dabei heißt ξ_i die i.te Restzahl bei der Entwicklung von x.

Lemma 4.3.9 *Jede rationale Zahl x hat eine endliche Kettenbruchentwicklung*

$$x = \operatorname{cfrac}(a_0, ..., a_n).$$

Diese ist nicht eindeutig, denn für $a_n \neq 1$ gilt

$$x = \operatorname{cfrac}(a_0, ..., a_n) = \operatorname{cfrac}(a_0, ..., a_n - 1, 1)$$

und für $a_n = 1$ gilt

$$x = \operatorname{cfrac}(a_0, ..., a_n) = \operatorname{cfrac}(a_0, ..., a_{n-1} + 1)$$

Dies sind die einzigen Mehrdeutigkeiten, die Auftreten können. □

Lemma 4.3.10 *Es sei $x \in \mathbb{R}$ gegeben. Für die gekürzten Näherungsbrüche* $\frac{p_i}{q_i} = \operatorname{cfrac}(a_0, ..., a_i)$ *gilt*

$$p_{j+1} = a_{j+1} p_j + p_{j-1}$$

und

$$q_{j+1} = a_{j+1} q_j + q_{j-1}.$$

Weiter gilt:

$$q_j p_{j-1} - p_j q_{j-1} = (-1)^j$$

□

Korollar 4.3.11 *Es sei $x \in \mathbb{R}$ mit $x = \operatorname{cfrac}(a_0, ..., a_n, \xi_{n+1})$ gegeben, wobei $\xi_{n+1} \geq 1$ die $n+1$ Restzahl sei. Weiter seien die gekürzten Naherungsbrüche von x gegeben durch $\frac{p_i}{q_i} = \operatorname{cfrac}(a_0, ..., a_i)$.Dann gilt*

$$x = \frac{p_n \xi_{n+1} + p_{n-1}}{q_n \xi_{n+1} + q_{n-1}}$$

□

Beweis von 4.3.8: Als erstes definieren wir δ durch folgende Gleichung:

$$x = \frac{p}{q} + \frac{\delta}{2q^2}. \tag{4.3}$$

Damit ist $|\delta| < 1$. Nun hat $\frac{p}{q}$ als rationale Zahl eine abbrechende Kettenbruchentwicklung

$$\frac{p}{q} = \operatorname{cfrac}(a_0, a_1, ..., a_n)$$

mit Näherungsbrüchen $\frac{p_i}{q_i} = \operatorname{cfrac}(a_0, ..., a_i)$ für $i \in \{0, ..., n\}$, insbesondere ist $\frac{p_n}{q_n} = \frac{p}{q}$. Nun ist die Darstellung von $\frac{p}{q}$ durch die Koeffizienten a_i nach 4.3.9 nicht eindeutig, und wir können o.B.d.A. $(-1)^n \delta > 0$ fordern.

Nun definieren wir weiter λ durch:

$$x = \frac{\lambda p_n + p_{n-1}}{\lambda q_n + q_{n-1}} \tag{4.4}$$

Es bleibt also zu zeigen, daß $\lambda > 1$ ist, denn damit ist λ eine gültige Restzahl der Kettenbruchentwicklung von x; also $\frac{p_n}{q_n}$ ein Näherungsbruch von x.

Auflösen von 4.4 nach λ, und Einsetzen von 4.3 liefert:

$$\lambda = 2\left(\frac{q_n p_{n-1} - p_n q_{n-1}}{\delta}\right) - \frac{q_{n-1}}{q_n}.$$

Nach Lemma 4.3.10 und wegen der Wahl von n folgt weiter:

$$\lambda = \frac{2(-1)^n}{\delta} - \frac{q_{n-1}}{q_n} = \frac{2}{|\delta|} - \frac{q_{n-1}}{q_n} \geq 2 - 1 = 1,$$

denn die q_i sind streng monoton steigend. $\square$

Damit ist $\frac{p}{q}$ ein Näherungsbruch von x. Wenn nun $\lambda = \mathrm{cfrac}(b_0, ..., b_m)$ ist, hat x insgesamt eine Darstellung als Kettenbruch $x = \mathrm{cfrac}(a_0, ..., a_n, \lambda) = \mathrm{cfrac}(a_0, ..., a_n, b_0, ..., b_m)$.

Das Lemma 4.3.8 liefert uns nun wegen 4.2 den Bruch $\frac{k}{r}$ mit Hilfe der Kettenbrüche.

4.3.6 Eigenwerte in Quantenalgorithmen

In diesem Abschnitt soll eine andere Sicht auf die Lösung des Faktorisierungs-Problem dargestellt werden, die als erstes von Kitaev [Ki95] beschrieben wurde. Die hier gegebene Darstellung lehnt sich an [Cl98] an. In 3.2.6 haben wir gesehen, wie wir den Eigenwert einer unitären Matrix bestimmen können, wenn wir einen passenden Eigenvektor gegeben haben. Wie uns diese Methode helfen kann, Zahlen zu faktorisieren, soll hier beschrieben werden.

Auch diese Methode faktorisiert eine Zahl n nicht direkt, sondern ist, wie Shors Algorithmus auch, in der Lage, zu einem beliebigen $a \in \mathbb{Z}_n^*$ die Ordnung zu bestimmen. Angenommen, wir hätten für ein a mit $\mathrm{ord}_n(a) = r$ den Zustand ψ_1 konstruiert mit

$$\psi_1 = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{-2\pi i j}{r}} |\mathbf{a^j} \bmod \mathbf{n}\rangle$$

Wir wollen nun die Funkton f als unitäre Abbildung anwenden, wobei

$$f : \mathbb{Z}_n \to \mathbb{Z}_n$$

$$f(x) = ax \bmod n$$

Dazu setzen wir:

$$\begin{aligned} U_f : \mathbb{C}^{2^l} &\to \mathbb{C}^{2^l} \\ U_f(|\mathbf{y}\rangle &= \begin{cases} |\mathbf{ay}\rangle & : \quad y < n \\ |\mathbf{y}\rangle & : \quad y \geq n \end{cases} \end{aligned}$$

wobei l so gewählt ist, das $2^{l-1} < n \leq 2^l$ gilt. Diese Abbildung ist sicher unitär, denn für $y \in \mathbb{Z}_n$ ist die Multiplikation mit a bijektiv und auf $y \geq N$ operiert die Abbildung trivial.

Dann erhalten wir

$$\begin{aligned} U_f(\psi_1) &= U_f\left(\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{-2\pi i j}{r}} |\mathbf{a^j} \bmod \mathbf{n}\rangle\right) && (4.5) \\ &= \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{-2\pi i j}{r}} |\mathbf{a^{j+1}} \bmod \mathbf{n}\rangle && (4.6) \end{aligned}$$

$$= e^{\frac{2\pi i}{r}} \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{-2\pi i (j+1)}{r}} |\mathbf{a^{j+1}} \bmod \mathbf{n}\rangle \quad (4.7)$$

$$= e^{\frac{2\pi i}{r}} \psi_1 \quad (4.8)$$

ψ_1 ist also ein Eigenvektor von U_f mit Eigenwert $e^{\frac{2\pi i}{r}}$. Wenn wir nun das Gatter aus 3.2.6 anwenden mit einem ersten Register der Länge m, liefert uns das die beste Approximation von $\frac{1}{r}$ mit einer Wahrscheinlichkeit von mehr als $\frac{4}{\pi^2}$. Wir erhalten also ein rationales δ, dessen Binärdarstellung nach m Stellen abbricht, so daß

$$\left|\delta - \frac{1}{r}\right| \leq 2^{-(m+1)}$$

gilt.

Nun müssen wir noch garantieren, daß sich mit dieser Information r berechnen läßt. Dazu kann, wie bei Shors Algorithmus auch, m so gewählt werden, daß $n^2 \leq 2^m \leq 2n^2$ erfüllt ist, denn somit kann $\frac{1}{r}$ mit Hilfe von Lemma 4.3.8 ermittelt werden.

Das einzige Problem, das nun noch bleibt, ist die Erzeugung des Zustandes ψ_1. Leider ist hierfür keine direkte Lösung bekannt. Mit Hilfe der folgenden Überlegung werden wir aber sehen, daß es gar nicht nötig ist, ψ_1 zu erzeugen. Hierzu betrachten wir nun für ein $k \in \{1, ..., r\}$ den Zustand

$$\psi_k = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{-2\pi i k j}{r}} |\mathbf{a^j} \bmod \mathbf{n}\rangle.$$

Man erkennt analog zu den Überlegungen zu ψ_1, daß jeder dieser Zustände ein Eigenvektor von U_f mit Eigenwert $e^{\frac{2\pi i k}{r}}$ ist. Die gleiche Überlegung wie oben liefert also auch hier eine Approximation von $\frac{k}{r}$ mit Hilfe der Methode aus 3.2.6. Aus dieser Approximation läßt sich nun der Wert $\frac{k}{r}$ ermitteln, r allerdings nur, wenn $\gcd(r, k) = 1$ gilt. Damit sind auch hier genau wie bei Shor eventuell erneute Durchläufe nötig, um r zu ermitteln.

Die Zustände ψ_k sind auch nicht einfacher zu konstruieren als ψ_1, aber die entscheidende Bemerkung ist:

$$\frac{1}{\sqrt{r}} \sum_{k=0}^{r} \psi_k = \frac{1}{r} \sum_{k=1}^{r} \sum_{j=0}^{r-1} e^{\frac{-2\pi i k j}{r}} |\mathbf{a^j} \bmod \mathbf{n}\rangle \quad (4.9)$$

$$= \frac{1}{r} \sum_{k=1}^{r} e^0 |\mathbf{a^0} \bmod \mathbf{n}\rangle + \frac{1}{r} \sum_{j=1}^{r-1} \sum_{k=1}^{r} \left(e^{\frac{-2\pi i j}{r}}\right)^k |\mathbf{a^j} \bmod \mathbf{n}\rangle \quad (4.10)$$

$$= |\mathbf{1}\rangle + \sum_{j=1}^{r-1} 0 |\mathbf{a^j} \bmod \mathbf{n}\rangle \quad (4.11)$$

$$= |\mathbf{1}\rangle \quad (4.12)$$

und der Zustand $|\mathbf{1}\rangle$ ist leicht konstruierbar. Damit führt man den Algorithmus zur Ermittlung des Eigenwertes nicht mit einem der ψ_k aus, sondern mit $|\mathbf{1}\rangle$, und erhält so eine Approximation von $\frac{k}{r}$ für ein gleich verteiltes $k \in \{1, .., r\}$. Im folgenden wird dieses Verfahren noch einmal zusammengefaßt.

Algorithmus 4.3.12 *Es werden 2 Register benötigt, mit Größen analog zu Shors Algorithmus.*

1. *Als erstes wir der Quantencomputer in den Zustand*
$$\phi_0 = |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle = |\mathbf{0}\rangle \otimes \frac{1}{\sqrt{r}} \sum_{k=0}^{r} \psi_k$$
gebracht.

2. *Wende nun die Transformation aus 3.2.6 an. Dies liefert den Zustand*
$$\phi_3 = \frac{1}{\sqrt{r}} \sum_{k=0}^{r} \left(a_k |[\mathbf{2^m \frac{k}{r}}]\rangle + b_k \rho_k \right) \otimes \psi_k$$
wobei in $b_k \rho_k$ die Zustände zusammengefasst sind, die nicht die beste m-Bit-Approximation von $2^m \frac{k}{r}$ liefern. Für alle a_k gilt $|a_k|^2 > 0,4$.

3. *Nun wird der Zustand gemessen und wir erhalten mit einer Wahrscheinlichkeit größer als $0,4$ eine Approximation von $\frac{k}{r}$ für ein zufälliges k.* □

Dieser Algorithmus hat somit die gleichen Eigenschaften wie Shors Algorithmus, liefert aber einen anderen Blickwinkel auf den Ablauf zur Berechnung der Ordnung eines Elementes. Auch hier können die Vorberechnungen aus 4.3.2 genutzt werden, um die Wahrscheinlichkeit, daß das Element a die zum Faktorisieren nötigen Bedingungen erfüllt, zu erhöhen.

4.3.7 Zwei Versionen des gleichen Algorithmus

In den Abschnitten 4.3.4 und 4.3.6 werden zwei Algorithmen für die Ordungsbestimmung eines Elementes $y \in \mathbb{Z}_n^*$ beschrieben. Hier soll nun erläutert werden, daß diese auf den ersten Blick unterschiedlichen Algorithmen, genau das selbe Ergebnis liefern. Dazu stellen wir zuerst fest, daß die Zustände der Quantencomputerer nach dem zweiten Schritt der Algorithmen identisch sind. Wir bezeichnen die Länge des ersten Registers mit t und die des Zweiten mit s.

Bei dem Algorithmus 4.3.12, wird auf den Zustand $|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle$ das Gatter 3.11 angewandt, wobei hier die Operation U der Multiplikation mit y entspricht. Hierbei wird nun auf jedes der Qubits des ersten Registers zuerst die Hadamard-Transformation angewandt. Dies ergibt den Zustand
$$\psi = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |\mathbf{k}\rangle \otimes |\mathbf{1}\rangle$$

Nun folgen bedingte Multiplikationen mit Potenzen von y, und dies liefert:

$$\frac{1}{\sqrt{2^t}}\sum_{x=0}^{2^t-1}|\mathbf{x}\rangle\otimes|\mathbf{1}\rangle = \frac{1}{\sqrt{2^t}}\left(\sum_{x_{t-1}=0}^{1}\sum_{x_{t-2}=0}^{1}\cdots\sum_{x_0=0}^{1}|\mathbf{x_{t-1}}\rangle\otimes|\mathbf{x_{t-2}}\rangle\cdots\otimes|\mathbf{x_0}\rangle\right)\otimes|\mathbf{1}\rangle$$
$$\to \frac{1}{\sqrt{2^t}}\left(\sum_{x_{t-1}=0}^{1}\cdots\sum_{x_0=0}^{1}|\mathbf{x_{t-1}}\rangle\otimes|\mathbf{x_{t-2}}\rangle\cdots\otimes|\mathbf{x_0}\rangle\right)\otimes|\mathbf{y^{x_0}y^{2x_1}\cdots y^{2^{t-1}x_{t-1}}} \bmod \mathbf{n}\rangle$$
$$= \frac{1}{\sqrt{2^t}}\sum_{x=0}^{2^t-1}|\mathbf{x}\rangle\otimes|\mathbf{y^x} \bmod \mathbf{n}\rangle = \psi_2$$

Dies ist aber genau der Zustand nach dem zweiten Schritt von Shors Algorithmus 4.3.6.

Der einzige Unterschied der jetzt noch bleibt ist, daß bei Algorithmus 4.3.6 auf diesen Zustand die Quanten-Fourier-Transformation angewandt wird, und bei Algorithmus 4.3.12 das Inverse der Quanten-Fourier-Transformation. Nun gilt aber $\overline{\psi_2} = \psi_2$, da dieser Vektor nur reelle Komponenten hat, und weiter ist $F_t^{-1} = \overline{F_t}$. Damit folgt $F_t^{-1}\psi_2 = \overline{F_t}\psi_2 = \overline{F_t\psi_2}$. Dies bedeutet, daß die Ergebnise der beiden Algorithmen durch Konjugieren ineinander übergehen. Für die Messung eines Zustandes ist dies aber kein Unterschied, da es hierbei nur auf die Beträge der Komponenten ankommt.

4.3.8 Das RSA-Public-Key-Verfahren

Um eine RSA-verschlüsselte Nachricht $c = x^e \bmod pq$ zu knacken, genügt es, ein d zu finden, so das $ed = 1 \bmod \operatorname{ord}_n(x)$, wobei $\operatorname{ord}_n(x)$ die Ordnung von x in $\mathbb{Z}_n^*$ bezeichne. Da aber $gcd(\phi(n), e) = 1$, folgt $\operatorname{ord}_n(x) = \operatorname{ord}_n(c)$. Das heißt, wir berechnen mit Shors Algorithmus die Ordnung von c und mit Hilfe des Euklidischen Algorithmus das multiplikative Inverse d von $e \bmod \operatorname{ord}_n(c)$. Damit ist es für das Brechen einer bestimmten, mit RSA verschlüsselten Nachricht nicht nötig, daß die berechnete Ordnung gerade ist, denn n muß nicht faktorisiert werden, um die Nachricht zu entschlüsseln.

4.4 Grovers Suchalgorithmus

4.4.1 Das Problem und die Lösung

Dieser von Lov Grover 1996 entwickelte Algorithmus [Gr97] ist, neben Shors Algorithmen, der für die Kryptographie bedeutendste. Es handelt sich hierbei um einen Suchalgorithmus für unsortierte Mengen. Genauer sei

$$f : \mathbb{Z}_2^n \to \mathbb{Z}_2$$

eine Funktion mit

$$f(x) = \begin{cases} 1 & : \quad x = x_0 \\ 0 & : \quad x \neq x_0 \end{cases}$$

Das Problem besteht nun darin, x_0 zu bestimmen, ohne spezielle Eigenschaften der Funktion f auszunutzen. Man kann sich vielleicht am besten eine Blackbox vorstellen, auf deren innere Funktionsweise kein Einblick möglich ist. Ein klassischer Rechner müßte die Funktion an 2^{n-1} Stellen berechnen, um mit einer Wahrscheinlichkeit von $\frac{1}{2}$ den Wert von x_0 bestimmen zu können, denn es bleibt nichts anderes übrig, als für zufällige Werte t den Funktionswert $f(t)$ zu bestimmen, und dies solange, bis für ein t die Funktion f den Wert 1 annimmt und somit $x_0 = t$ gefunden wurde.

Grover entwickelte nun einen Algorithmus, bei dem schon $2^{\frac{n}{2}}$ Ausführungen von f für eine Erfolgswahrscheinlichkeit von über $\frac{1}{2}$ ausreichen; also eine enorme Verbesserung.

Die Bedeutung für die Kryptographie liegt nun darin, daß sich mit Hilfe von Grovers Algorithmus Brute-Force-Angriffe deutlich verbessern lassen. Wir betrachten einen symmetrischen Chiffrier-Algorithmus

$$s : \mathbb{Z}_2^m \times \mathbb{Z}_2^n \to \mathbb{Z}_2^m \tag{4.13}$$

$$s(M, K) = C \tag{4.14}$$

wobei der Klartext M mit dem Schlüssel K auf den Chiffretext C abgebildet wird. Hierbei nehmen wir an, daß die Funktion $s(M, \cdot) : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ eine Injektion ist, daß also jedes Klartext/Chiffretext-Paar (M, C) den Schlüssel eindeutig bestimmt.

Sei nun ein Klartext/Chiffretext Paar (M_0, C_0) mit $C_0 = s(M_0, K_0)$ gegeben. Wenn wir nun die Funktion f definieren als

$$f : \mathbb{Z}_2^n \to \mathbb{Z}_2$$

$$f(K) = \begin{cases} 1 & : \quad s(M_0, K) = C_0 \\ 0 & : \quad s(M_0, K) \neq C_0 \end{cases}$$

und Grovers Suchalgorithmus anwenden, dann liefert dieser uns wegen $f(K) = 0 \Leftrightarrow K = K_0$ nach $2^{\frac{n}{2}}$ Schritten den richtigen Schlüssel K_0 mit einer Wahrscheinlichkeit größer als $\frac{1}{2}$. Dies entspricht effektiv einer Halbierung der Schlüssellänge. Ein symmetrischer Algorithmus mit einer Schlüssellänge von 128 Bit, der mit klassischen Rechnern vor einem Brute-Force-Angriff nahezu sicher ist, könnte so mit Hilfe von Quantencomputern geknackt werden, und das, ohne eine Schwäche des Algorithmus auszunutzen. Hierbei haben wir implizit immer vorausgesetzt, daß sich die Funktion f und damit auch die Funktion s effektiv auf Quantencomputern umsetzten lassen, was theoretisch für jeden Chiffrier-Algorithmus der Fall ist, aber in der praktischen Umsetzung durchaus Probleme bereiten könnte. Darauf soll hier aber nicht weiter eingegangen werden.

Wir beschreiben nun als erstes den Algorithmus, und anschließend werden die Details erläutert.

Algorithmus 4.4.1 *Für diesen Algorithmus wird ein Register mit n Qubits und ein Register mit einem Qubit benötigt.*

1. *Als erstes wird der Quantencomputer in den Zustand* $|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle$ *gesetzt und die Hadamard-Transformation* R^{n+1} *auf alle* $n+1$ *Qubits angewandt.*

$$\psi_0 = R^{n+1}(|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle) = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\mathbf{k}\rangle \otimes \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$$

Diese etwas unübliche Darstellung hat den Vorteil, daß die Bedeutung von Schritt 2a) deutlicher wird.

2. *Jetzt folgen zwei Schritte, die in einer Schleife* $m(n)$ *mal durchlaufen werden. Der Wert* $m(n)$ *wird weiter unten genau bestimmt. Wir bezeichnen den Zustand des Quantencomputers vor der i.ten Iteration mit*

$$\psi_i = \sum_{k=0}^{2^n-1} \alpha(k,i)|\mathbf{k}\rangle \otimes \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$$

(a) *Die unitäre Martix* U_f *wird angewandt und liefert*

$$\psi_i' = U_f(\psi_i) = \sum_{k=0}^{2^n-1} \alpha(k,i)|\mathbf{k}\rangle \otimes (-1)^{f(k)} \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$$

Dieser Schritt kommt auch in Deutschs Algorithmus zum Tragen.

(b) *Auf den Quantencomputer im Zustand* ψ_i' *wird nun die unitäre Transformation* D *angewandt, wobei*

$$D = 2P - I \text{ und } P = (p_{ij})_{i,j=0,\dots,2^n-1} \text{ mit } p_{ij} = \frac{1}{2^n}$$

Der Beweis, daß D *eine unitäre Transformation darstellt, erfolgt unten. Dies liefert einen Zustand.*

$$\psi_{i+1} = (D \otimes I)(\psi_i') = \sum_{k=0}^{2^n-1} \alpha(k,i+1)|\mathbf{k}\rangle \otimes \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$$

3. *Nach den nötigen Iterationen des Schrittes 2 folgt nun abschließend eine Messung der ersten n Qubits. Dabei ist die Wahrscheinlichkeit, daß ein bestimmtes* $|\mathbf{k}\rangle$ *gemessen wird, gegeben durch* $|\alpha(k, m(n))|^2$*, und diese Wahrscheinlichkeit wird weiter unten zu bestimmen sein. Hierbei wird sich zeigen, daß* $m(n)$ *so gewählt werden kann, daß* $\alpha(k_0, m(n))$ *für das gesuchte* k_0 *betragsmäßig deutlich größer ist als die übrigen.* □

Dieser Algorithmus scheint auf den ersten Blick sehr seltsam und wenig intuitiv. Eine etwas andere Beschreibung des Schrittes 2 wird dies allerdings ändern.

4.4.2 Analyse des Algorithmus

Der Schritt 2(a) lässt sich auch als bedingte Anweisung schreiben:

$$\text{if } (f(k) \neq 0) \text{ then } \alpha(k,i) = (-1)\alpha(k,i) \text{ else } \alpha(k,i) = \alpha(k,i)$$

Wichtig hierbei ist, daß diese bedingte Anweisung ohne Messung erfolgt; es wird also nicht erst der Zustand gemessen und je nach Ergebnis reagiert, sondern diese bedingte Anweisung läßt sich als unitäre Matrix darstellen und geht somit nicht mit einer Zerstörung des Zustandes einher. Der Schritt 2(b) läßt sich als eine "Spiegelung am arithmetischen Mittel" interpretieren. Dazu bemerken wir zuerst, daß die Abbildung P die orthogonale Projektion auf den von $(1, ..., 1)$ erzeugten Untervektoraum ist und somit $P^2 = P$ gilt. Damit ist D die Spiegelung an dieser Geraden. Weiter bildet P jeden Vektor $w = \sum_k \alpha_k |\mathbf{k}\rangle$ auf den Vektor, der in jeder Komponente das arithmetische Mittel der Komponenten von w hat, ab.

$$Pw = P \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2^n-1} \end{pmatrix} = \frac{1}{2^n} \sum_i \alpha_i \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Damit ist auch klar, daß D eine unitäre Matrix ist, denn man sieht sofort, daß $D^* = D$ ist, und es bleibt zu zeigen $DD^* = D^2 = I$. Es ist aber

$$D^2 = (2P - I)^2 = 4P^2 - 4P + I = I$$

Für einen Vektor $w = \sum_k \alpha_k |\mathbf{k}\rangle$ und das zugehörige arithmetische Mittel $\alpha_m = \frac{1}{2^n} \sum_k \alpha_k$ ergibt sich dann durch die Transformation D für die j.te Komponente des Bildes:

$$(Dw)_j = ((2P - I)w)_j = 2\alpha_m - \alpha_j = \alpha_m - (\alpha_j - \alpha_m) \tag{4.15}$$

und dies ist genau die Spiegelung am arithmetischen Mittel. Der Schritt 2 läßt sich insgesamt am besten mit folgendem Bild beschreiben:

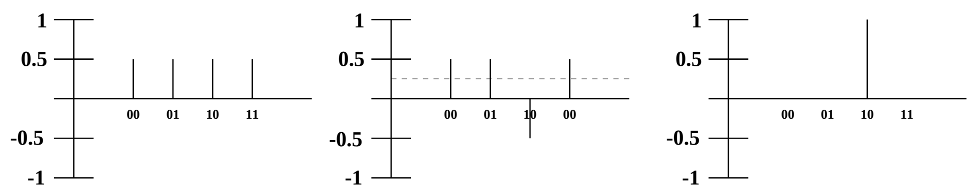

Abbildung 4.1: Die Wirkung des zweiten Schrittes bei 2 Qubits, die gestrichelte Linie entspricht dem arithmetischen Mittel. In diesem Fall ist schon nach einem Schritt das richtige Ergebnis (hier 10) gefunden.

Damit ist verständlich, warum eine Iteration des Schrittes 2 die Möglichkeit bietet, die Wahrscheinlichkeit der richtigen Lösung zu erhöhen. Wichtig ist aber die richtige Anzahl der Iterationen, und diese soll im folgenden bestimmt werden.

Wir betrachten die Entwicklung der Amplituden der gewünschten Lösung $\alpha(k_0, i)$ und der anderen Werte $\alpha(k, i)$ für $k \neq k_0$. Hierfür bemerken wir zuerst, daß für $k_1, k_2 \neq k_0$

$$\alpha(k_1, i) = \alpha(k_2, i) \Rightarrow \alpha(k_1, i+1) = \alpha(k_2, i+1)$$

wie sofort aus Gleichung 4.15 folgt. Da nun zu Anfang alle $\alpha(k, i)$ gleich sind, reicht es also, die Entwicklung eines $\alpha(k', i)$ für $k' \neq k_0$ zu betrachten, da die anderen während des gesamten Algorithmus mit diesem übereinstimmen. Dies vereinfacht die Analyse.

Der Schritt 2 hat dann wegen 4.15 folgende Wirkung

$$\begin{pmatrix} \alpha(k', i+1) \\ \alpha(k_0, i+1) \end{pmatrix} = T \begin{pmatrix} \alpha(k', i) \\ \alpha(k_0, i) \end{pmatrix}$$

wobei

$$T = \begin{pmatrix} 1 - \frac{1}{2^{n-1}} & -\frac{1}{2^{n-1}} \\ 2 - \frac{1}{2^{n-1}} & 1 - \frac{1}{2^{n-1}} \end{pmatrix}$$

Nun wissen wir weiterhin, daß $\alpha(k_0, i)^2 + (2^n - 1)\alpha(k', i)^2 = 1$ für alle i, und wenn wir nun $a_1(i) := \sqrt{2^n - 1}\alpha(k', i)$ und $a_2(i) := \alpha(k_0, i)$ setzen, erhalten wir einen Vektor $a(i)$ der Länge eins, dessen Koordinaten die Entwicklung der Wahrscheinlichkeit des richtigen bzw. falschen Ergebnisses widerspiegeln. Wenn wir nun die Matrix betrachten, die $a(i)$ in $a(i+1)$ überführt, so erhalten wir:

$$a(i+1) = \begin{pmatrix} a_1(i+1) \\ a_2(i+1) \end{pmatrix} = T' a(i)$$

mit

$$\begin{aligned} T' &= \begin{pmatrix} (2^n - 1)^{\frac{1}{2}} & 0 \\ 0 & 1 \end{pmatrix} T \begin{pmatrix} (2^n - 1)^{-\frac{1}{2}} & 0 \\ 0 & 1 \end{pmatrix} && (4.16) \\ &= \begin{pmatrix} 1 - 2^{-n+1} & -2^{(-n+1)}(2^n - 1)^{\frac{1}{2}} \\ 2^{(-n+1)}(2^n - 1)^{\frac{1}{2}} & 1 - 2^{-n+1} \end{pmatrix} && (4.17) \end{aligned}$$

Die Matrix T' ist aber eine Rotation um einen Winkel θ, und da der Vektor $a(i)$ immer reell ist, reicht es, sich die Wirkung von T' auf dem Einheitskreis im $\mathbb{R}^2$ zu veranschaulichen.

Der Vektor $a(0)$ entsteht aus einer Drehung des Vektors $\binom{1}{0}$ um den Winkel $\frac{\theta}{2}$, denn

$$a(0) = 2^{\frac{n}{2}} \begin{pmatrix} \sqrt{2^n - 1} \\ 1 \end{pmatrix} = T_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ mit } T_0 = 2^{\frac{n}{2}} \begin{pmatrix} \sqrt{2^n - 1} & -1 \\ 1 & \sqrt{2^n - 1} \end{pmatrix}$$

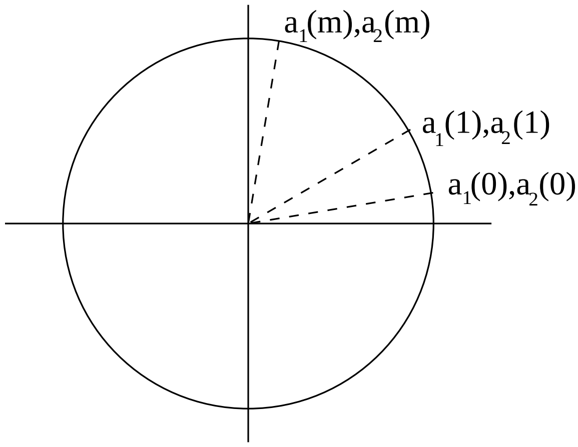

Abbildung 4.2: Die Änderungen der Wahrscheinlichkeiten durch die Abbildung T'.

und es gilt $T_0^2 = T'$. Die Wahrscheinlichkeit für das richtige Ergebnis ist dann am größten, wenn der gesamte Rotationswinkel nach $m(n)$ Iterationen nahe bei $\frac{\pi}{2}$ liegt. Das bedeutet

$$\frac{\pi}{2} = \theta \left(m(n) + \frac{1}{2} \right)$$

und damit

$$m(n) = \frac{\pi}{2\theta} - \frac{1}{2} \tag{4.18}$$

Wir haben aber $\cos\theta = 1 - 2^{-n+1}$, und dies ist für große n nahe bei 1 und θ damit nahe bei 0. Dann ist aber $\cos\theta \approx 1 - \frac{\theta^2}{2}$ und weiter $\theta \approx \frac{2}{\sqrt{2^n}}$. Damit ergibt sich die Anzahl der Iterationen mit 4.18 zu

$$m(n) \approx 2^{\frac{n-4}{2}}\pi - \frac{1}{2}$$

und damit ist gezeigt, daß die Wahrscheinlichkeit der richtigen Lösung in $O(\sqrt{2^n})$ Schritten auf über $\frac{1}{2}$ steigt.

Die Frage, ob es nicht noch effizientere Methoden für das Suchproblem auf Quantencomputern geben kann, kann mit nein beantwortet werden, wie weiter unten diskutiert werden soll. Zur vollständigen Beschreibung des Algorithmus fehlt noch folgendes

Lemma 4.4.2 *Die unitäre Transformation $D = 2P - I$ ist effizient implementierbar.*

Beweis: Die Transformation $D = 2P - I$ stimmt mit $R^n S R^n$ überein, wobei

$$S_{ij} = \begin{cases} 1 & : \; i = j = 0 \\ -1 & : \; i = j \neq 0 \\ 0 & : \; \textit{sonst} \end{cases}$$

und R^n wie immer die Hadamard-Transformation darstellt. Denn $S = S_2 - I$ mit

$$S_2 = \begin{pmatrix} 2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

und damit $R^n S R^n = R^n(S_2 - I)R^n = R^n S_2 R^n - I$, und es bleibt zu zeigen, daß $R^n S_2 R^n = 2P$ gilt:

$$\begin{aligned} R^n S_2 R^n(|\mathbf{k}\rangle) &= \frac{1}{\sqrt{2^n}} R^n S_2 \left(\sum_{u=0}^{2^n-1} (-1)^{(k,u)} |\mathbf{u}\rangle \right) \\ &= \frac{2}{\sqrt{2^n}} R^n(|\mathbf{0}\rangle) \\ &= \frac{2}{2^n} \sum_{x=0}^{2^n-1} |\mathbf{x}\rangle \\ &= 2P(|\mathbf{k}\rangle) \end{aligned}$$

Um also Grovers Algorithmus zu implementieren, ist neben U_f, welches wir als effizient implementierbar vorausgesetzt haben, und der Hadamard-Transformation R nur die unitäre Operation S nötig. Für diese Operation müssen wir nun noch eine Zerlegung in elementare Teilumformungen finden. Dazu betrachten wir zuerst die Operation $-S$.

Die Operation $-S$ bildet $(|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle \cdots \otimes |\mathbf{0}\rangle)$ auf $(-1)(|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle \cdots \otimes |\mathbf{0}\rangle)$ und alle anderen Basisvektoren auf sich selbst ab. Hierfür wird zuerst auf alle n Qubits die NOT-Operation angewandt, das bedeutet insbesondere: $(|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle \cdots \otimes |\mathbf{0}\rangle)$ wird auf $(|\mathbf{1}\rangle \otimes |\mathbf{1}\rangle \cdots \otimes |\mathbf{1}\rangle)$ abgebildet. Danach wird mit einem n-XOR ein Hilfs-Qubit genau dann auf $|\mathbf{1}\rangle$ gesetzt, wenn der Wert nach dem NOT $(|\mathbf{1}\rangle \otimes |\mathbf{1}\rangle \cdots \otimes |\mathbf{1}\rangle)$ war. Anschließend wird auf ein beliebiges der ersten n Qubits ein $-I$ angewandt, wenn das Hilfs-Qubit $|\mathbf{1}\rangle$ war. Um das Hilfs-Qubit nun wieder auf seinen Ausgangswert zu setzen, wird erneut ein n-XOR auf das Hilfs-Qubit angewandt. Abschließend wird wieder die NOT Operation auf jedem Qubit angewandt, und insgesamt ergibt sich so $-S$.

Um nun mit Hilfe von $-S$ die Operation S umzusetzten reicht es auf ein beliebiges Qubit $-I$ anzuwenden.

Damit ist das Lemma bewiesen. $\square$

Ein ausführlicheres Beispiel ist in den Abildungen 4.3 bis 4.6 gegeben. Dieses Beispiel sucht in 32 Werten den richtigen (hier 0010). Auf der CDRom zu dieser Diplomarbeit gibt es auch eine Simulation von Grovers Algorithmus, in der die Arbeitsweise des Algorithmus und auch die Umsetztun von D mit Hilfe von Lemma 4.4.2 verdeutlicht werden.

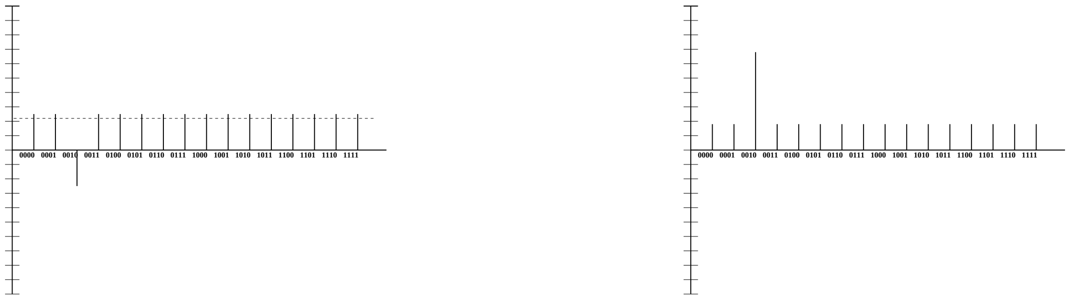

Abbildung 4.3: erste Iteration

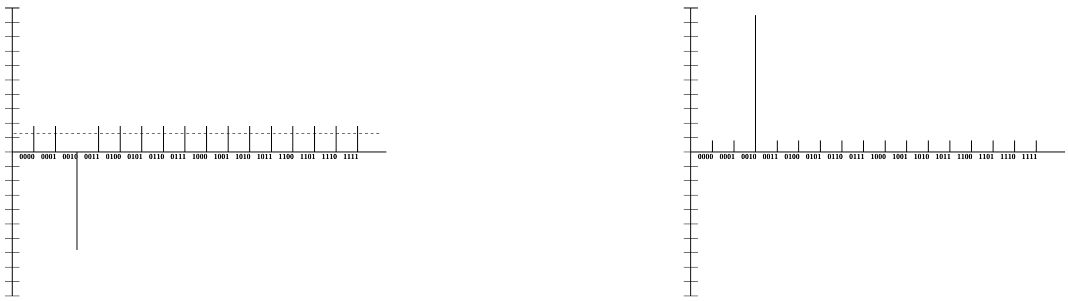

Abbildung 4.4: zweite Iteration

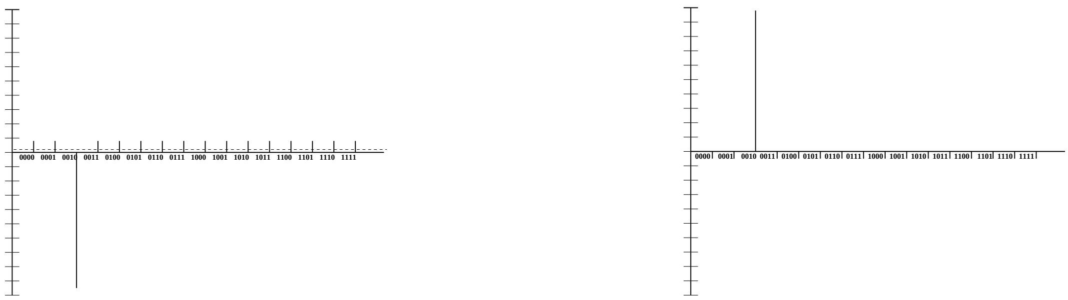

Abbildung 4.5: dritte Iteration

Abbildung 4.6: vierte Iteration

4.4.3 Untere Grenzen für Suchalgorithmen

In diesem Abschnitt soll gezeigt werden, daß Grovers Algorithmus das beschriebene Problem asymptotisch optimal löst. Der Beweis hierzu ist aus [NC00] entnommen. Hierzu betrachten wir für jedes x aus $\mathbb{Z}_2^n$ die Funktion

$$f_x : \mathbb{Z}_2^n \to \mathbb{Z}_2$$

$$f_x(a) = \begin{cases} 1 & : \quad a = x \\ 0 & : \quad a \neq x \end{cases} .$$

Die Umsetzung als unitäre Operation, bezeichnen wir mit U_x, wobei

$$U_x(|\mathbf{k}\rangle) = \begin{cases} |\mathbf{k}\rangle & : \quad k = x \\ -|\mathbf{k}\rangle & : \quad k \neq x \end{cases}$$

Dies entspricht der Anwendung von U_f in Grovers Algorithmus. Wenn wir mit P_x die Projektion auf den von $|\mathbf{x}\rangle$ aufgespannten Untervektorraum bezeichnen gilt $U_x = I - 2P_x$.

Von einen generischen Algorithmus können wir nun annehmen, daß er aus einigen Anwendungen von U_x und zusätzlichen unitären Operationen A_i besteht. Wir bezeichnen den Anfangszustand mit ψ und den Zustand nach k Anwendungen von U_x mit ψ_k^x. Es ist also

$$\psi_k^x = A_k U_x A_{k-1} U_x \cdots A_2 U_x A_1 U_x \psi$$

Weiter definieren wir:

$$\psi_k = A_k A_{k-1} \cdots A_2 A_1 \psi$$

Dieser Zustand unterscheidet sich von ψ_k^x also nur durch Anwendungen von U_x. Die Idee ist nun eine Schranke für die Abweichung von ψ_k von allen ψ_k^x zu finden, und dies mit der Erfolgswahrscheinlichkeit in Verbindung zu setzten. Dazu setzen wir

$$D_k = \sum_{x=0}^{2^n-1} \|\psi_k^x - \psi_k\|^2$$

Wir beweisen nun folgendes Lemma:

Lemma 4.4.3 *Seien die Bezeichnungen wir oben, dann gilt:*

$$D_k = \sum_{x=0}^{2^n-1} \|\psi_k^x - \psi_k\|^2 \leq 4k^2$$

Beweis: Wir beweisen die Behauptung über Induktion nach k.

Für $k = 0$ ist $D_k = 0 \leq 4k^2$, es bleibt also die Untersuchung von D_{k+1}. Es gilt:

$$\begin{aligned}
D_{k+1} &= \sum_{x=0}^{2^n-1} \|\psi_{k+1}^x - \psi_{k+1}\|^2 \\
&= \sum_{x=0}^{2^n-1} \|A_{k+1}(U_x\psi_k^x - \psi_k)\|^2 \\
&= \sum_{x=0}^{2^n-1} \|U_x\psi_k^x - \psi_k\|^2 \\
&= \sum_{x=0}^{2^n-1} \|U_x(\psi_k^x - \psi_k) + (U_x - I)\psi_k\|^2
\end{aligned}$$

Es gilt für $a, b \in \mathbb{C}^{2^n}$ die Ungleichung $\|a+b\|^2 \leq \|a\|^2 + 2\|a\|\|b\| + \|b\|^2$ und mit $a = U_x(\psi_k^x - \psi_k)$ und $b = (U_x - I)\psi_k$ folgt:

$$D_{k+1} \leq \sum_{x=0}^{2^n-1} \|\psi_k^x - \psi_k\|^2 + 4\|\psi_k^x - \psi_k\|\|P_x\psi_k\| + 4\|P_x\psi_k\|^2 \tag{4.19}$$

$$= D_k + 4\sum_{x=0}^{2^n-1} \|\psi_k^x - \psi_k\|\|P_x\psi_k\| + 4\sum_{x=0}^{2^n-1} \|P_x\psi_k\|^2 \tag{4.20}$$

nun gilt $\sum_{x=0}^{2^n-1} \|P_x\psi_k\|^2 = \|\psi_k\| = 1$, und auf den mittleren Term können wir die Cauchy-Schwarz-Ungleichung anwenden. Dies liefert:

$$D_{k+1} \leq D_k + 4\left(\sum_{x=0}^{2^n-1} \|\psi_k^x - \psi_k\|^2\right)^{\frac{1}{2}} \left(\sum_{y=0}^{2^n-1} \|P_y\psi_k\|^2\right)^{\frac{1}{2}} + 4 \tag{4.21}$$

$$= D_k + 4\sqrt{D_k} + 4 \tag{4.22}$$

und mit der Induktionsannahme $D_k \leq 4k^2$ folgt

$$D_{k+1} \leq 4k^2 + 8k + 1 = 4(k+1)^2$$

Damit ist das Lemma bewiesen. □

Jetzt müssen wir diese Ergebnis noch mit der Wahrscheinlichkeit den richtigen Wert zu messen in Zusammenhang setzen. Dazu nehmen wir an, daß die Wahrscheinlichkeit den richtigen Wert zu messen größer als $\frac{1}{2}$ ist, also $\|P_x\psi_k^x\| = | < |\mathbf{x}\rangle, \psi_k^x > | \geq \frac{1}{2}$. Aus dieser Annahme können wir eine asymptotische untere Schranke für den Wert k, also die Anzahl der Schritte, herleiten. Wir können $|x\rangle$ durch $e^{i\tau}|x\rangle$ ersetzen, ohne das sich die Erfolgswahrscheinlichkeit ändert, und so o.E.d.A. $| < |\mathbf{x}\rangle, \psi_k^x > | =< |\mathbf{x}\rangle, \psi_k^x >$ annehmen. Damit ist

$$\|\psi_k^x - |\mathbf{x}\rangle\|^2 = 2 - 2| < |\mathbf{x}\rangle, \psi_k^x > | \leq 2 - \sqrt{2}$$

Wir definieren weiter $E_k = \sum_{x=0}^{2^n-1} \|\psi_k^x - |\mathbf{x}\rangle\|^2$ und es gilt $E_k \leq (2-\sqrt{2})2^n$. Mit $F_k = \sum_{x=0}^{2^n-1} \||\mathbf{x}\rangle - \psi_k\|^2$ gilt:

$$\begin{aligned} D_k &= \sum_{x=0}^{2^n-1} \|(\psi_k^x - |\mathbf{x}\rangle) + (|\mathbf{x}\rangle - \psi_k)\|^2 \\ &\geq \sum_{x=0}^{2^n-1} \|\psi_k^x - |\mathbf{x}\rangle\|^2 - 2\sum_{x=0}^{2^n-1} \|\psi_k^x - |\mathbf{x}\rangle\|\||\mathbf{x}\rangle - \psi_k\| + \sum_{x=0}^{2^n-1} \||\mathbf{x}\rangle - \psi_k\|^2 \\ &= E_k + F_k - 2\sum_{x=0}^{2^n-1} \|\psi_k^x - |\mathbf{x}\rangle\|\||\mathbf{x}\rangle - \psi_k\| \end{aligned}$$

wobei wir für $a, b \in \mathbb{C}^{2^n}$ die Ungleichung $\|a+b\|^2 \geq \|a\|^2 - 2\|a\|\|b\| + \|b\|^2$ ausgenutzt haben. Aus dem letzten Term erhalten wir mit Hilfe der Cauchy-Schwarz-Ungleichung $\sum_{x=0}^{2^n-1} \|\psi_k^x - |\mathbf{x}\rangle\|\||\mathbf{x}\rangle - \psi_k\| \leq \sqrt{E_k F_k}$, und es folgt

$$D_k \geq \left(\sqrt{F_k} - \sqrt{E_k}\right)^2$$

Mit Lemma 4.4.4 und der Abschätzung für E_k erhalten wir somit für hinreichend großes n und einer geeignetet Konstanten c

$$D_k \geq c2^n$$

Mit Hilfe von 4.4.3 ergibt sich:

$$4k^2 \geq D_k \geq c2^n$$

und also

$$k \geq \sqrt{\frac{c2^n}{4}}$$

Um also eine Erfolgswahrscheinlichkeit von mehr als $\frac{1}{2}$ zu erhalten muß ein beliebiger Algorithmus die Transformation U_x asymtotisch $O(\sqrt{2^n})$ anwenden, und dies zeigt, das Grovers Lösung asymptotisch optimal ist.

Lemma 4.4.4 *Mit obigen Bezeichnungen gilt:*

$$F_k \geq 22^n - 2\sqrt{2^n}$$

Beweis: Es ist

$$\begin{aligned} F_k &= \sum_{x=0}^{2^n-1} \||\mathbf{x}\rangle - \psi_k\|^2 \\ &= \sum_{x=0}^{2^n-1} < |\mathbf{x}\rangle - \psi_k, |\mathbf{x}\rangle - \psi_k > \end{aligned}$$

$$
\begin{aligned}
&= \sum_{x=0}^{2^n-1} (\||\mathbf{x}\rangle\| + \|\psi_k\| - 2Re(<|\mathbf{x}\rangle, \psi_k>)) \\
&= 22^n - 2\sum_{x=0}^{2^n-1} Re(<|\mathbf{x}\rangle, \psi_k>) \\
&\geq 22^n - 2\sum_{x=0}^{2^n-1} |<|\mathbf{x}\rangle, \psi_k>|1
\end{aligned}
$$

Die letzte Summe können wir mit der Cauchy-Schwarz-Ungleichung abschätzen

$$
\begin{aligned}
F_k &\geq 22^n - 2\left(\sum_{x=0}^{2^n-1} |<|\mathbf{x}\rangle, \psi_k>|^2\right)^{\frac{1}{2}} \left(\sum_{y=0}^{2^n-1} 1\right)^{\frac{1}{2}} \\
&= 22^n - 2\sqrt{2^n}
\end{aligned}
$$

$\square$

Kapitel 5

Simulationen mit jaQuzzi

In diesem Kapitel möchte ich die Möglichkeiten zur Simulation von Quantencomputern auf klassischen Rechnern vorstellen. Hierbei beziehe ich mich hauptsächlich auf den Simulator jaQuzzi von Felix Schürmann. Eine Übersicht über weitere Simulatoren und auch Programmiersprachen für simulierte Quantencomputer findet sich unter "www.desex.ac.qc.uk/ jwallace/simtable.htm".

Die Simulation von Quantencomputern auf klassischen Rechnern kann sich immer nur auf sehr wenige Qubits beschränken, da der Aufwand der Simulation im Normalfall exponentiell mit der Anzahl der Qubits wächst. Weiter ist die Behandlung von Zuständen als Dichtematrizen noch deutlich komplexer und so begnügen sich alle mir bekannten Simulatoren mit der Simulation von reinen Zuständen. Dies stellt aber für die Anwendungen aus dieser Arbeit kein Problem dar, denn alle hier besprochenen Algorithmen bauen auf einem Modell mit reinen Zuständen auf.

Ein Simulator für Quantencomputer muß für n Qubits 2^n komplexe Zahlen, also die Komponenten des Zustandsvektors in $\mathbb{C}^{2^n}$, verwalten. Dies geschieht natürlich nur auf eine durch den Computer bestimmete Rechengenauigkeit. Die Rundungsfehler, die hierbei entstehen, spielen in den Beispielen allerdings keine Rolle.

5.1 Die CDRom

Auf der beiligenden CDRom sind der Simulator jaQuzzi, sowie die nötigen JAVA Komponenten enthalten. Zusätlich gibt es noch eine Dokumentation zu jaQuzzi, und die folgenden, von mir mit jaQuzzi erzeugten Beispielsimulationen. Es gibt eine Webseite, die durch die Anwendungen führt. Diese ist auf der CDRom unter "/index.html" zu finden.

5.2 jaQuzzi

Der in JAVA programmierte Simulator von Felix Schürmann hat sich bei den von mir gewählten Beispielen als sehr einfach zu handhaben und gut bei der Darstellung der Zustände während des Ablaufes eines Algorithmus gezeigt. Weitere Vorteile diese Simulators sind die durch JAVA gegebene Plattformunabhänigkeit und die Tatsache, daß auch der Quellcode des Programmes zur Verfügung steht.

Als JAVA-Programm ist jaQuzzi leider nicht so schnell wie es wünschenswert wäre, und es kommt auch hin und wieder zu Programmabstürzen. Leider erlaubt es jaQuzzi auch nicht, selbst erzeugte Gatter in späteren Gattern als fertige Teilstücke einzufügen. Dadurch wird zum Beispiel der Aufbau eines Multiplizierers zu einer reinen Fleißaufgabe, so das ich mich hier mit einem einfachen Addierer begnügt habe.

5.2.1 ein 3-Qubit-Addierer

Als erstes Beispiel soll ein einfacher 3-Qubit-Addierer dienen, wie er unter 3.2.1 beschrieben ist. Da, wie oben erwähnt, das Bündeln von Teiloperationen mit jaQuzzi nicht möglich ist, gibt es hier keine expliziten CARRY- und SUM-Operationen, sondern diese stecken im Schaltbild selbst.

An der linken Seite des Schaltplans sind die einzelnen Qubits gekennzeichnet. Die Schaltung addiert dann den Wert von $4a_2 + 2a_1 + a_0$ zu $4b_2 + 2b_1 + b_0$ und speichert das Ergebnis in den Qubits b_0, b_1, b_2 und in dem zusätzlichen Qubit c, welches einen Überlauf anzeigt. Die Qubits c_0, c_1 und c_2 dienen, wie unter 3.2.1 erläutert, als Speicher für Zwischenergebnisse und werden am Ende des Addierers wieder auf ihre Ursprungswerte gesetzt.

Bei dieser Auflösung der CARRY- und SUM-Operationen erkennt man auch, daß die XOR-Operationen in der Mitte des Addierers nur den letzten Teil der vorangegangenen CARRY-Operation rückgänig macht. Diese beiden XOR-Operationen können natürlich entfallen, sind aber wegen des Bezuges auf 3.2.1 beibehalten worden.

Um diesen Addierer zu testen, kann man nun einfach die Werte a_i und b_i beliebig festlegen und beobachten, was passiert.

Da dieser Addierer so noch nicht mehr leistet als jeder normale Addierer auf einem klasischen Rechner, gibt es noch ein weiteres Beispiel, welches die Stärke der Quantencomputer besser verdeutlicht. Bei diesem Schaltdiagramm wurde auch die Reihnfolge der Qubits geändert, um die Anfangswerte einfacher setzen und vor allem das Ergebnis einfacher ablesen zu können. Dies geschiet einfach durch Vertauschen der Qubits, bevor diese an den Addierer weitergeleitet werden und einem inversen Vertauschen danach.

Um den Vorteil eines Quantencomputers gegenüber klasischen Rechnern deutlich zu machen, wurden zwei Hadamardtranformationen auf die Qubits a_0 und a_1 eingefühgt. Dies bewirkt, daß nun bei jedem Durchlauf die Addition nicht nur auf einem einzigen Wert, also Basisvektor, durchgeführt wird, sondern auf vier Werten gleichzeitig.

5.2.2 Grovers Algorithmus für 5 Qubits

Als weiteres Beispiel eines Algorithmus habe ich Grovers Suchalgorithmus (siehe 4.4) gewählt. Die Funktion, die untersucht werden soll, ist:

$$f : \mathbb{Z}_2^5 \to \mathbb{Z}_2^5$$

$$f(x) = \begin{cases} 1 & : \quad x = 01101 \\ 0 & : \quad sonst \end{cases}$$

Leider ist es mit jaQuzzi nicht möglich, den Anfangszustand der Qubits mit abzuspeichern. Wenn ein Algorithmus neu geladen wird, befinden sich alle Qubits am Anfang standardmäßig im Zustand 0. Für Grovers Algorithmus muß das Ergebnis-Qubit von U_f allerdings mit 1 belegt sein, um die Berechnungen richtig durchzuführen. Aus diesem Grund wird vor Begin des eigentlichen Algorithmus auf dieses Qubit eine NOT-Operation angewandt.

Der Hauptteil von Grovers Algorithmus ist die Gruppe von Operationen, die hier mit "Iteration" bezeichnet wird. Diese Operationen entsprechen der Schleife aus 4.4.1.

Die unitäre Matrix U_f wird umgesetzt, indem zuerst der Vektor $|\mathbf{01101}\rangle$ auf $|\mathbf{11111}\rangle$ abgebildet wird und auf das Ausgabe-Qubit ein, durch alle fünf Eingabe-Qubits kontrolliertes, XOR angewand wird.

Die "Spiegelung am arithmetischen Mittel" wird in diesem Schaltbild gemäß Lemma 4.4.2 umgesetzt. Das hierbei benutzte Hilfs-Qubit wird wie gefordert nach jedem Durchlauf wieder auf den Ausgangszustand gesetzt.

Interesant ist nun zu sehen, wie sich die Warscheinlichkeiten der einzelnen Ergebnisse je nach Anzahl der Iterationen der Schleife verändern. Hier bietet jaQuzzi eine gute Darstellung der Wahrscheinlichkeitsverteilung an, die man erhält, wenn man vor dem Klicken auf das Symbol zur graphischen Anzeige des Zustandes nur die gewünschten Qubits markiert. Dieses Markieren kann in einem beliebigen Gatter erfolgen. Daraufhin berücksichtigt jaQuzzi nur noch die markierten Qubits bei der Berechnung der Wahrscheinlichkeit.

Für Grovers Algorithmus ist es nun sinnvoll, sich nur die Wahrscheinlichkeitsverteilung der ersten 5 Qubits anzusehen, denn nur hier steht der gesuchte Wert. Die größte Wahrscheinlichkeit, den gesuchten Wert zu messen, erhält man nach 4 Iterationen, und dies ist auch die im Gatter voreingestellte Anzahl.

Wenn man sich nur einen Teil der Qubits ansieht, ist jaQuzzi leider nicht immer in der Lage, auch hier die entsprechenden Real- und Imaginärteile der Zustände anzugeben. Dies liegt daran, daß sobald nur ein Teilzustand betrachtet wird, es sich hierbei nicht mehr um einen reinen Zustand handeln muß. Wenn es sich um keinen reinen Zustand handelt, lassen sich zwar immer noch die Wahrscheinlichkeiten bei einer Messung leicht ermitteln, aber Real- und Imaginärteile der Zustände können nicht mehr angegeben werden. jaQuzzi kann nun, sobald sich mehrere Qubits in Überlagerungszuständen befinden, nicht entscheiden, ob dieser Zustand noch ein reiner Zustand ist oder nicht, und bietet somit keine Anzeige mehr für Teilzustände außer der Wahrscheinlichkeitsverteilung. Deshalb ist die "Spiegelung am arithmetischen Mittel", der eigentliche Kern des Algorithmus, bei der Betrachtung von Teilzuständen nicht zu beobachten.

Literaturverzeichnis

[Ah98] Aharonov, D., “Quantum Computation”, quant-ph/9812037

[Ba95] Barenco, A., Bennett, C.H., Cleve, R., et al., “Elementary Gates for Quantum Computation”, Physical Review A, 1995

[Be96] Beckman, David, Preskill, John, “Efficient networks for quantum factoring”, Phys. Rev. A, vol. 54, no. 2, 3457-3467, 1996

[Bo96] Boyer, Michel, “Tight bounds on quantum searching”, Proc. Phys Comp 1996 (quant-h/9605034)

[Br97] Brassard, G., Hoyer, P., “An Exact Quantum Polynomial-Time Algorithm for Simon’s Problem”, quant-ph/9704027, 1997

[Cl97] Cleve, R., Ekert, A., Macchiavello, C., Mosca, M., “Quantum Algorithms Revisited”, quant-ph/9708016

[Cl98] Cleve, R., Ekert, A., Macchiavello, C., Mosca, M., ”Quantum Algorithms Revisited”, Proc. Roy. Soc. London A, Volume 454, 339-354, 1998

[De85] Deutsch, David, “Quantum theory, the Church-Turing principle and the universal quantum computer”, Proc. Roy. Soc. London A, Volume 400, 97-117, 1985

[Fo96] Foster, Otto, “Algorithmische Zahlentheorie”, Vieweg, 1996

[Gr97] Grover, Lov, “Quantum Mechanics helps in searching for a needle in a haystack”, Phys. Rev. Letters, Volume 79, issue2, pp. 325-328, 1997 (quant-ph/9706033)

[Jo99] Jozsa, R., “Searching in Grover’s Algorithm”, quant-ph/9901021

[Ki95] Kitaev, A., “Quantum Measurments and the Abelian Stabiliser Problem”, e-print quant-ph/9511026, 1995

[Lo00] Lomonaco, Samuel J., “A Lecture on Shor’s Quantum Factoring Algorithm”, quant-ph/0010034

[ME99] MOSCA, Michele, EKERT, Artur, “The Hidden Subgroup Problem and Eigenvalue Estimation on a Quantum Computer”, quant-ph/9903071

[NC00] NIELSEN, M. A., CHUANG, I. L., “Quantum Computation and Quantum Information”, Cambrige Univ. Press, 2000

[Pi00] PITTENGER, Arthur O., “An Introduction to Quantum Computing Algorithms”, Birkhäuser, 2000

[Sc] SCHÜRMANN, Jens, “jaQuzzi”, “www.desex.ac.qc.uk/~jwallace/simtable.htm”

[Sh94] SHANKAR, R., “Principles of Quantum Mechanics”, Plenum Press, 1994

[Sh95] SCHUMACHER, Benjamin, “Quantum Coding”, Phys. Rev. A, vol. 51, 2738-2747, April 1995

[Sh97] SHOR, Peter W., “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”, SIAM Journal on Computing, Volume 26, Number 5, pp. 1484-1509

[Si97] SIMON, Daniel R., “On the Power of Quantum Computation”, SIAM J. Comput, Volume 26, No. 5, pp. 1474-1483, 1997

[Su86] SUDBERY, Anthony, “Quantum Mechanics and the Particles of Nature”, Cambrige Univ. Press, 1986

[Ve96] VEDRAL, V., BARENCO, A., ECKERT, A., “Quantum Networks for elemenatry arithmetic operations”, Phys. Rev. A, vol. 54, 1996