5. Quantencomputer

5.1. Elemente der klassischen Theorie des "computings"

Wesentliche Elemente moderner Rechenmaschinen gehen auf Charles Babbage (1791-1871) zurück. As Vater der formalen Theorie des klassischen "computings" gilt Alan Turing (1912-1954)

Rechnung: systematische Erzeugung eines endlichen Satzes von (binären) Symbolen mit abstrakter Bedeutung aus anderen endlichen Satz von (binären) Symbolen

Netzwerkmodell klassischer computer

Rechnung ist äquivalent zur Auswertung von

(1) $$f: \{0,1\}^n \longrightarrow \{0,1\}^m$$

für bestimmtes n-bit Argument (1) läßt sich reduzieren auf m Funktionen

(2) $$f: \{0,1\}^n \longrightarrow \{0,1\}$$

Die Gesamtzahl der möglichen Funktionen f ist 2^{2^n} (2 outputs und 2^n inputs). Jede dieser 2^{2^n} Funktionen läßt sich durch eine Sequenz elementarer logischer Operationen darstellen. Dazu zerlegen wir die Menge der inputs

(3) $$x = x_1 x_2 x_3 \cdots x_n$$

in zwei Klassen je nachdem ob $f = 1$ oder $f = 0$ ist.
$x^{(a)}$ beschreibe alle x mit $f(x) = 1$, $f^{(a)}(x)$ sei

(4) $$f^{(a)}(x) = \begin{cases} 1 & \text{für } x = x^{(a)} \\ 0 & \text{sonst} \end{cases}$$

Dann gilt offensichtlich

(5) $$f(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee f^{(3)}(x) \vee \cdots$$

($\vee$ = ODER). Betrachten wir die Funktionen $f^{(a)}(x)$.
Für den Fall, daß $x^{(a)} = 111 \cdots 1$ können wir schreiben

(6) $$f^{(a)}(x) = x_1 \wedge x_2 \wedge x_3 \wedge \cdots$$

($\wedge$ = UND). Falls $x^{(a)}$ an bestimmten Stellen 0 enthält kann dies durch (6) und die Negation $\neg x$ ($\neg$ = NICHT) dargestellt werden, z.B.,

(7) $$f^{(a)}(x) = (\neg x_1) \wedge x_2 \wedge x_3 \wedge (\neg x_4) \wedge \cdots$$

Um $f(x)$ gemäß Gl. (5) auswerten zu können benötigen wir ein weiteres Element, nämlich

(8) $$\text{COPY}: \quad x \longrightarrow xx$$

da jeder der Funktionen $f^{(a)}(x)$ einen separaten Input

benötigt. Wir sehen, daß die Berechnung einer beliebigen Funktion $f: \{0,1\}^n \to \{0,1\}$ durch eine endliche Anzahl der logischen Operationen

(9) $\underbrace{\text{COPY, NICHT}}_{\text{1-bit}}$, $\underbrace{\text{UND, ODER}}_{\text{2-bit}}$

darstellbar ist. Man kann zeigen, daß COPY und NUND (NICHT UND) ebenfalls hinreichend sind. Wenn man darüberhinaus in der Lage ist ein bit in einem bestimmten Zustand zu präparieren ($x=0$ oder $x=1$), reicht die Kombination einer 2-bit Operation U und NICHT als 1-bit Operation aus:

(10) $$(x,y) \xrightarrow{U} (1-x, 1-xy)$$

ist äquivalent zu NUND, falls wir das erste output bit ignorieren und ist für $y=1$ in Kombination mit einem bitweisen NICHT äquivalent zu COPY. Man sagt U ist ein <u>universelles Gatter</u>

$\Rightarrow$ **Netzwerkmodell eines klass. Computers**

- enthält wenige Basiselemente: 1-bit und 2-bit Gatter
- Präparation von bits in bes. Zuständen möglich

Rechnung = endliche Abfolge von Basisoperationen (azyklisch!)

Wir haben gesehen, daß die Berechnung einer Funktion f bei einem gegebenen input $x : \{0,1\}^n$ durch eine bestimmte endliche Zahl elementarer Operationen möglich ist. Eine interessante und wichtige Frage ist nun wie sich diese Zahl elementarer Operationen verhält, wenn wir f auf beliebige x mit zunehmender Größe (n) anwenden. Das führt uns auf den Begriff der <u>Komplexität einer Rechnung</u>

Komplexitätsklassen klassischer Rechnungen

Eine Funktion f der Art von Gleichung (2) kodiert ein Entscheidungsproblem: $0 \hat{=}$ nein $1 \hat{=}$ ja. Die meisten mathematischen Probleme lassen sich in Entscheidungsprobleme umformen.

Wir wollen die Berechnung einer Funktion $f: \{0,1\}^n \to \{0,1\}$ mit variablen n <u>einfach</u> nennen, falls die erforderliche Zahl von elementaren Gattern eine polynomiale Funktion von n ist. Wir werden später sehen, daß die Entscheidung ob dies der Fall ist oder nicht unabhängig von der konkreten Realisierung von f in einer Rechenmaschine ist. Alle Funktionen f (Probleme) dieser Art sind Elemente der <u>Komplexitätsklasse</u>

P = {Entscheidungsprobleme die mit poly(n) Elementen lösbar sind}

Probleme die eine Anzahl von Basiselementen erfordern, die stärker als ein Polynom mit n anwächst, wollen wir schwer nennen.

Eine weitere Komplexitätsklasse sind die Probleme bei denen die Verifizierung einer gegebenen Lösung einfach ist, das Auffinden der Lösung aber sowohl einfach als auch schwer sein kann.

$NP = \{$ Entscheidungsprobleme für die die Verifizierung einer gegebenen Lösung poly(n) Elemente erfordert $\}$

(11) Offensichtlich gilt $P \subseteq NP$

(12) Vermutung $P \neq NP$

Ein wichtiges Beispiel aus der Klasse NP ist die CIRCUIT-SAT Funktion (Funktional)

(13) $$C[f(x)] = \begin{cases} 1 & \text{falls } X \text{ existiert mit } f(x)=1 \\ 0 & \text{sonst} \end{cases}$$

Theorem von Cook (1971):

Jedes Problem aus der Klasse NP ist mit polynomialen Aufwand auf CIRCUIT-SAT reduzierbar

Falls CIRCUIT-SAT $\in P \Rightarrow NP = P$, daher ist die

Vermutung, daß CIRCUIT-SAT $\in NP$ aber $\notin P$.

Es existieren noch weitere Probleme aus NP die dieselbe Eigenschaft wie CIRCUIT-SAT haben, daß sich alle Probleme aus NP auf sie mit polynomialen Aufwand reduzieren lassen. Diese Klasse von Problemen heißt <u>NP-vollständig</u> oder <u>NPC</u> (NP-complete). Falls wir für nur ein Element aus NPC zeigen können, daß es zu P gehört, folgt $NP = P$.

(14) falls $NP \neq P \quad \Longrightarrow \quad NPC \cap P = \emptyset$

(15) falls $NP \neq P \quad \Longrightarrow \quad NPC \cup P \neq NP$ (o.B.)

d.h. es gibt Probleme aus der Klasse NP die weder P noch NPC sind. Ein Problem von dem angenommen wird, daß es zu dieser Klasse (<u>NPI</u> genannt) gehört, ist die <u>Faktorisierung</u>.

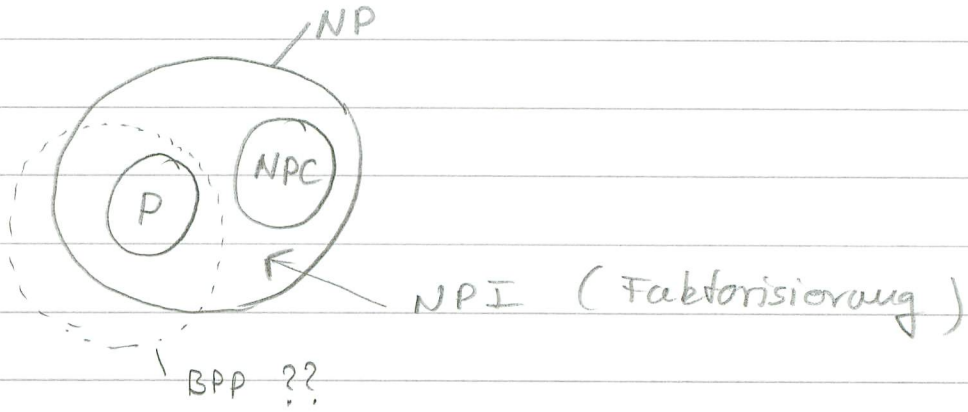
Bisher haben wir nur <u>deterministische</u> Rechnungen betrachtet. Es stellt sich jedoch heraus, daß es u.U. effizienter sein kann in einem Rechenpfad bestimmte Gatteroperationen durch einen Zufallsgenerator festlegen zu lassen. Ein solche <u>probabilistische</u> Berechnung eines Entscheidungsproblems führt nur mit endlicher Wahrscheinlichkeit zum richtigen Resultat. Falls die richtige Entscheidung für jeden Input jedoch mit Wahrscheinlichkeit $\frac{1}{2} + \delta$, $\delta > 0$ gefunden wird, kann das Problem durch Wiederholung der Rechnung und Majoritätsentscheidung mit beliebig

großer Wahrscheinlichkeit korrekt gelöst werden. Die Klasse von Problemen die sich mittels probabilistischen Rechnungen mit polynomialen Aufwand lösen lassen heißt BPP (bounded-error probabilistic polynomial). Es gilt offensichtlich

(16) $$P \subseteq BPP$$

Die Beziehung zwischen NP und BPP ist unbekannt.

wahrscheinliche Struktur der Komplexitätsklassen



Bei der Diskussion von Komplexitätsklassen waren wir stets davon ausgegangen, daß das Entscheidungsproblem eine Lösung besitzt. Tatsächlich haben Turing und Gödel gezeigt, daß es Entscheidungsprobleme gibt die nicht entscheidbar, d.h. nicht lösbar sind.* Ein Beispiel für ein solches Problem ist das Halteproblem

* mit endlichem Aufwand

universeller klassischer Computer: die Turing Maschine

Ein konzeptionell einfaches und wie wir sehen werden sehr wichtiges Modell einer endlichen Rechenmaschine ist von Alan Turing vorgeschlagen worden:

Turing Maschine

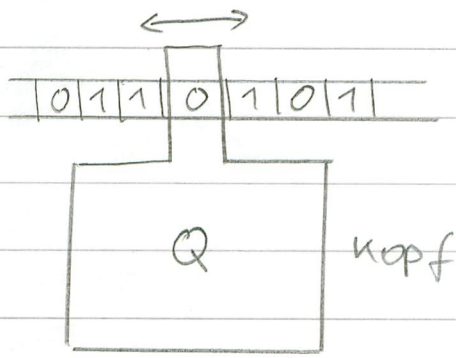
Band: diskrete Speicherplätze mit Symbolen s aus Alphabet Σ

Schreib/Lesekopf: interner Zustand $q \in Q = \{q_0, \ldots, q_n\}$
q_0 Anfangszustand
q_n Haltezustand

input: endlicher Satz von Symbolen auf Band im Anfangszustand (Programm + Daten)

output: endlicher Satz von Symbolen auf Band nachdem (und falls) Maschine anhält

Rechenschritt: $q, s \longrightarrow q', s'$ und Kopf rückt auf Band vor/zurück oder bleibt stehen



Eine Turing Maschine $T(x)$ mit binärem Input x kann durch Reaktion auf Input 0 oder 1 für alle möglichen (endlich vielen) Zustände q_i vollständig charakterisiert werden. Diese Spezifikation kann selber wieder als binäre Zahl $d[T]$ geschrieben werden.

Turing:

> $\exists$ universelle Turing Maschine $U(d[T], x)$ die jede Turing Maschine $T(x)$ mit Spezifikation $d[T(x)]$ mit nur polynomial größeren Aufwand simulieren kann.

Folgerung: Ein Problem daß auf U zu einer bestimmten Komplexitätsklasse gehört, gehört dies auf allen Turing Maschinen. Man kann zeigen, daß derselbe Schluß auch für alle Netzwerkcomputer gilt.

Mit Hilfe der Konstruktion der Turing-Maschine kann man zeigen, daß es unberechenbare Probleme gibt. Ein solches Problem ist das Halteproblem: Wir konstruieren eine Turing Maschine T_H die folgendes Problem löst:

gegeben Input x und $d[T]$, würde die Turing Maschine $T(x)$ zum Halten kommen?

$\Rightarrow$ $T_H(d[T])$ hält $\Leftrightarrow$ T hält nicht

aber:

$T_H(d[T_H])$ hält $\Leftrightarrow$ T_H hält nicht ↯

$\Rightarrow \nexists$ Turing Maschine, die das Halteproblem löst.

Eine Funktion $f: \Sigma \to \Sigma$ heißt berechenbar mit einer Turing Maschine, wenn für alle $x \in \Sigma$ als input die Turing Maschine anhält und als output $y \in \Sigma$ liefert

Church-Turing These:

Die Klasse der mit einer Turing Maschine berechenbaren Funktionen entspricht exakt der Klasse von Funktionen die man als durch einen Algorithmus als berechenbar ansehen würde.

reversibles computing

Um einen Quantencomputer zu entwerfen werden wir das Netzwerkmodell klassischer Rechner verallgemeinern und quanten-Gatter einführen. Diese werden unitären Operationen entsprechen. Unitäre operationen sind aber reversibel. Daher stellt sich die Frage nach universellen reversiblen klassischen logischen Basiselementen.

Ein zweiter Grund reversible Logik zu betrachten, ist die Tatsache, daß irreversible Elemente Information vernichten und somit Entropie erhöhen und zu Wärme produktion führen.

reversibles Rechnen $\hat{=}$ invertierbare Funktion

(17) $$f: \{0,1\}^n \longrightarrow \{0,1\}^n$$

irreversible Rechnungen können durch Hinzunahme von Ancilla bits zu reversiblen Rechnungen eingebettet werden. z.B.

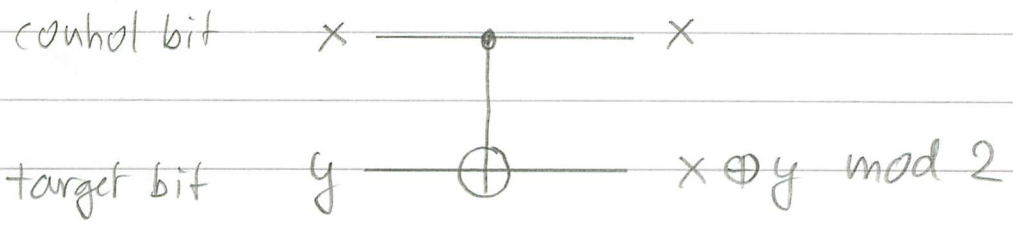
(18) $f: f(x) \longrightarrow f: \{0,1\}^n \longrightarrow \{0,1\}^m$

kann überführt werden in $\tilde{f}: \{0,1\}^{n+m} \longrightarrow \{0,1\}^{n+m}$

(19) $$\tilde{f}(x; 0^{(m)}) = (x; f(x))$$

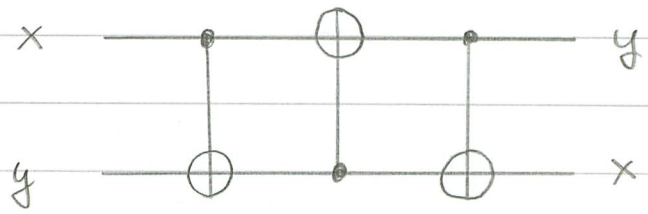
Von den mögl. 1-bit Gattern sind zwei reversibel: IDENTITY und NOT, Von den 256 mögl. 2-bit Gattern sind 24 reversibel, z.B. das CNOT

$$CNOT: (x,y) \longrightarrow (x, x \oplus y \bmod 2)$$



CNOT $\hat{=}$ NOT für $x=1$ und CNOT $\hat{=}$ COPY für $y=0$.

Hintereinanderschalten von 3 CNOT entspricht einem bit-swapping



Reversible 1-bit und 2-bit Gatter sind nicht universell, denn sie sind alle linear

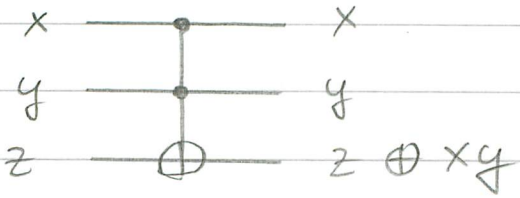
$$(20) \qquad \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}$$

wobei $\begin{pmatrix} a \\ b \end{pmatrix} \in \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$ und $M \in$

$$\left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \underbrace{\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}}_{\text{CNOT}}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \right\}$$

(insgesamt $6 \times 4 = 24$). Die Nacheinanderausführung der linearen Operationen ist ebenfalls linear und damit können in Computerschaltkreis aus reversiblen 1 und 2-bit Gattern keine nichtlineare Funktion berechnen. Für $n \geq 3$ existieren jedoch reversible Gatter die einer nichtlinearen Abbildung entsprechen.

<u>Toffoli-Gatter</u> controlled-controlled-NOT



Das Toffoli gatter ist universell falls die Präparation von Inputbits in bestimmten Zuständen (0 oder 1) möglich ist.

Durch die Ersetzung irreversibler durch reversible Gatter kann der Energieverbrauch, d.h. die Wärmeerzeugung bei einzelnen Gatteroperationen vermieden werden. Dennoch ist jede Rechnung mit einem unvermeidbaren Energieverlust verbunden da mindestens der output einer vorangegangenen Rechnung irreversibel gelöscht werden muß um den Ausgangszustand wieder herzustellen.

5.2. Quantennetzwerkcomputer

← 21.06.06

Ein Quantencomputer ist ein klassischer Computer dessen Ressourcen (im Fall des Turing computers: Band, Kopf, im Falle eines Netzwerkcomputers: Register und Gatter) den Gesetzen der Quantenmechanik genügen.

bits ⟶ qubits

input ⟶ präparierter faktorisierter Zustand eines Satzes von qubits in Rechenbasis $|0\rangle, |1\rangle$

z.B.

$10101 \longrightarrow |1\rangle|1\rangle|0\rangle|1\rangle|0\rangle$

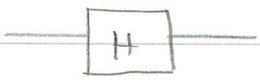
output ⟶ von Neumann Messung eines bestimmten Satzes von qubits in Rechenbasis $|0\rangle, |1\rangle$:
liefert klass. Satz von bits
011001

Gatter ⟶ unitäre Operationen

Da unitäre Operationen kontinuierlich sind können mit einer endlichen Anzahl von elementaren Gattern unitäre Operationen maximal approximativ realisiert werden.

Bem: Statt unitäre Operationen hätte man auch allg. lokale Operationen (siehe Kap. 1), statt von-Neumann Messungen auch unvollständige projektive Messungen zulassen können. Das benutzte Modell ist aber hinreichend allgemein

wichtige 1-qubit Gatter

Hadamard 

$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

$\pi/8$ 

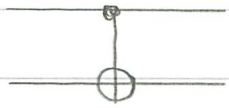
$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$

Mit Hilfe des Hadamard und $\pi/8$ Gatters kann

jede unitäre Operation auf einem qubit beliebig genau approximiert werden.

2-qubit Gatter

CNOT

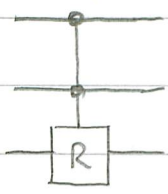


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Es gilt: Jedes 2-qubit Gatter kann aus einem CNOT und einem 1-qubit Gatter zusammengesetzt werden.

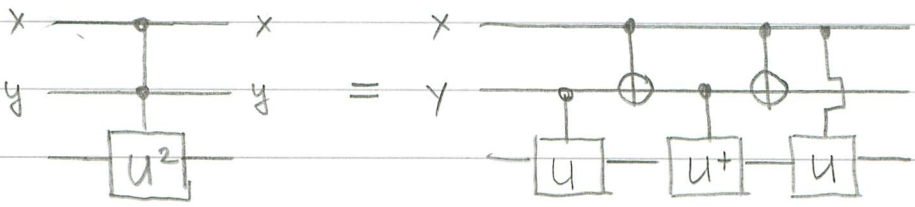
3-qubit Gatter

David Deutsch hat 1989 gezeigt, daß in Analogie zum klassischen Toffoli gatter ein universelles 3-qubit Quanten-gatter existiert



wobei $R = -i \exp\left[i \frac{\theta}{2} \sigma_x\right]$ falls erstes und zweites qubit im Zstd. $|1\rangle$ sind.

Ein solches 3-qubit Gatter läßt sich jedoch aus 2-qubit Gattern zusammensetzen:



Die Wirkung ist

$$U^y U^{+(x\oplus y)} U^x = U^{y-x\oplus y+x}$$

(21)

$$= U^{y-(x+y-2xy)+x} = U^{2xy}$$

d.h. für Implementierung des Deutsch Gatters

(22) $$U^2 = R(\theta) \Rightarrow U = e^{-i\frac{\pi}{4}} R\left(\frac{\theta}{2}\right)$$

Theorem von Deutsch (1985)

> Sei U eine beliebige d-dimensionale unitäre Matrix. Dann kann U als Produkt aus $2d^2-d$ unitären 2×2 Matrizen geschrieben werden.

Da die 2×2 Matrizen im Theorem von Deutsch i.A. auch Zustände verschiedener qubits verknüpfen benötigt man zur Darstellung einer beliebigen unitären Matrix U poly(d) 1-qubit Gatter die nur auf den 2-dim Unterräumen $\mathcal{H}_i$ einzelner qubits wirken, sowie poly(d) 2-qubit Gatter die auf 4-dim. Unterräumen $\mathcal{H}_i \otimes \mathcal{H}_j$ zwischen beliebigen qubits i und j wirken.

Netzwerkmodell eines Quantencomputers

5.3. Quantenalgorithmen

Wir haben bereits gesehen, daß Rechnungen auf einem Quantencomputer i.A. von probabilistischer Natur sein werden. Wir werden daher die Komplexitätsklasse BQP, die Klasse aller Probleme die sich mit polynomialen Aufwand und beliebig hoher Wahrscheinlichkeit auf einem Quantencomputer berechnen lassen einführen. Sicher gilt

(23) $$P \subseteq BPP \subseteq BQP$$

Außerdem werden wir sehen daß Quantenalgorithmen existieren, die die Vermutung nahelegen daß

(24) $$BQP \neq BPP$$

Der Nachweis von (24) sowie die Aufklärung der Verhältnisse zwischen BQP, BPP und NP sind einige der wichtigen ungelösten Probleme der Komplexitätstheorie.

Klar ist, daß ein mittels klassischer Rechner unentscheidbares Problem auch durch einen Quantencomputer nicht lösbar ist, denn jede unitäre Evolution kann mittels klassischer Rechner beschrieben werden. Es gilt sogar daß mittels stochastischer Simulationen (z.B.) unitäre Operationen mit polynomialen Aufwand an physikalischen Gattern und Speicherplätzen jedoch mit nichtpolyno-

minden Aufwand an Zeit klassisch berechenbar sind. Die Klasse von Problemen, die mit polynomialen Aufwand an Platz aber nicht notwendig polynomialen Aufwand an Rechenzeit mit klass. Computern lösbar ist, heißt PSPACE. Es gilt

(25) $BQP, BPP, NP \subseteq PSPACE$

Wir werden 3 Klassen von Quantenalgorithmen kennenlernen die sich von klass. Algorithmen unterscheiden durch

(i) nichtexponentiellen speedup (Grovers Suchalgorithm.)
(ii) exponentieller speedup bzgl. eines Orakels
(Ein Quantenorakel ist eine unbekannte unitäre Transformation. Wir werden sehen daß es Algorithmen gibt, bei denen der Input des Orakels geschickt präpariert wird so daß die unitäre Transformation mit polynomialen Aufwand bestimmbar ist (Simons Algorithm.)
(iii) exponentieller speedup schwerer Probleme
oder besser scheinbar schwerer Probleme
(Shor's Faktorisierung)

(A) Deutsch und Deutsch-Josza Problem

gehört zu keiner der oben angegebenen Klassen, ist aber instruktiv:

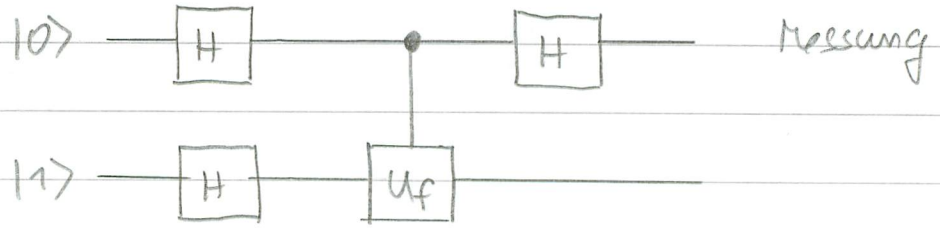
geg: $f: \{0,1\} \longrightarrow \{0,1\}$

$f(0)=0$	$f(0)=1$	f: konstant
$f(1)=0$	$f(1)=1$	

(26)

$f(0)=0$	$f(0)=1$	f: balanciert
$f(1)=1$	$f(1)=0$	

Um klassisch zu bestimmen ob f konstant oder f balanciert sind 2 Anfragen nötig. Quantenmechanisch ist dies durch 1 Anfrage möglich. Dazu verwendet man (Deutsch)



wobei

(27) $$U_f: \quad |x\rangle|y\rangle \longrightarrow |x\rangle|y \oplus f(x)\rangle$$

Wirkung:

(28) $$|0\rangle|1\rangle \xrightarrow{\text{H-Gatter}} \frac{1}{2}\left(|0\rangle+|1\rangle\right)\left(|0\rangle-|1\rangle\right) = \frac{1}{2}\sum_{x=1}^{2}|x\rangle\left(|0\rangle-|1\rangle\right)$$

es gilt

$$U_f\,|x\rangle\left(|0\rangle-|1\rangle\right) = \begin{cases} |x\rangle\left(|0\rangle-|1\rangle\right) & f(x)=0 \\ |x\rangle\left(|1\rangle-|0\rangle\right) & f(x)=1 \end{cases}$$

(29) $$= (-1)^{f(x)}\,|x\rangle\left(|0\rangle-|1\rangle\right)$$

$$\frac{1}{2}\sum_{x=1}^{2}|x\rangle(|0\rangle-|1\rangle) \xrightarrow{U_f} \frac{1}{2}\sum_{x=1}^{2}(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$$

(30)
$$= \begin{cases} \pm\frac{1}{2}(|0\rangle+|1\rangle)(|0\rangle-|1\rangle) & f: \text{konst.} \\ \pm\frac{1}{2}(|0\rangle-|1\rangle)(|0\rangle-|1\rangle) & f: \text{balanc.} \end{cases}$$

$$\xrightarrow{\text{Hadamard}} = \begin{cases} \pm\frac{1}{2}|0\rangle(|0\rangle-|1\rangle) & f: \text{konst.} \\ \pm\frac{1}{2}|1\rangle(|0\rangle-|1\rangle) & f: \text{balanc.} \end{cases}$$

Messung des 1. qubits liefert "0" → f = konstant, "1" → f = balanciert.

Die Essenz des "speedups" des Quantenalgorithmus gegenüber einem klassischen liegt hier in der Verwendung der Superposition $|0\rangle+|1\rangle$ als Input für f.

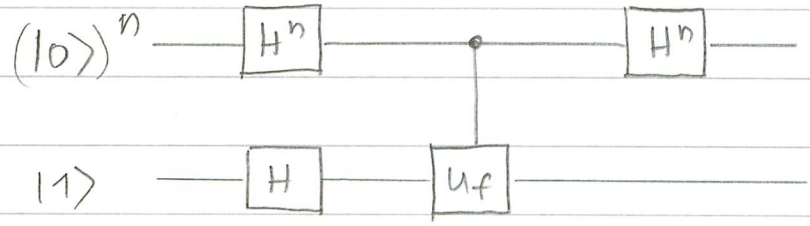
Speedup durch Superposition: Quantenparallelismus

Der Effekt wird dramatischer wenn wir eine Funktion

(31) $$f: \{0,1\}^n \longrightarrow \{0,1\}$$

betrachten (Deutsch-Josza) von der wir wissen, daß f entweder konstant ist ($f(x)\equiv 0$ oder $f(x)\equiv 1$ für alle x) oder balanciert ist (f(x) nimmt Werte 0 und 1 gleich häufig an). (Eine beliebige Funktion f der

Strahlsatz (31) wird weder das eine noch das andere sein.)
Um klassisch zu entscheiden, ob f konstant oder balanciert ist, sind $2^{n-1}+1$ Anfragen nötig, da selbst wenn 2^{n-1} Anfragen dasselbe Resultat liefern f immer noch balanciert sein kann. Quantenmechanisch kann dies mit einer Anfrage erledigt werden:



wobei $H^n = H \otimes H \otimes \cdots \otimes H$ bitweise H bedeutet

$$(32) \quad H^n |x\rangle = \prod_{i=1}^{n} \frac{1}{\sqrt{2}} \sum_{y_i=0}^{1} (-1)^{x_i y_i} |y_i\rangle$$

$$= \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle$$

wobei $x \cdot y$ bitweises AND bzw. Skalarprodukt modulo 2 ist

Gatterwirkung:

$$(|0\rangle)^n |1\rangle \xrightarrow{H^n H} \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

$$(33) \quad \xrightarrow{U_f} \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

$$\xrightarrow{H^n} \frac{1}{2^n} \left(\sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y} |y\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

betrachten $\sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y} = S(y)$

f: konstant

(34) $S(y) = \left(\pm \sum_{x=0}^{2^n-1} (-1)^{x \cdot y}\right) = \pm \delta_{y,0} \cdot 2^n$

(falls $y \neq 0$ enthält Summe gleiche Anzahl von "+" wie "−")
D.h. falls f: konstant erhalten wir den Ausgangszustand $|y=0\rangle = |0^n\rangle$ mit Wahrscheinlichkeit 1.

f: balanciert

(35) $S(y=0) = \pm \sum_{x=0}^{2^n-1} (-1)^{f(x)} = 0$

D.h. Wahrscheinlichkeit des Ausgangszustandes $|y=0\rangle$ ist 0. Projektion auf $|y=0\rangle$ erlaubt Entscheidung ob f konstant oder balanciert.

Das Deutsch-Josza Problem ist jedoch in der Klasse BPP: Nehmen wir an wir wollen das richtige Ergebnis nur mit Wahrscheinlichkeit

$$P \geq 1 - \varepsilon$$

richtig raten. Wenn f tatsächlich balanciert ist, ist die Wahrscheinlichkeit in k zufälligen Anfragen dasselbe Resultat zu bekommen $p = 2^{-(k-1)}$. Wenn wir bei k gleichen sukzessiven Ausgängen raten, daß f balanciert ist, ist die Wahrscheinlichkeit daß wir richtig raten $\frac{1}{2^{k-1}+1}$.

Sinnvollerweise raten wir natürlich $f = \text{konstant}$. Das hat dann die Fehlerwahrscheinlichkeit

$$1 - P = \frac{1}{2^{k+1}(2^{k-1}+1)} \tag{36}$$

D.h. $P = 1 - \varepsilon$ kann mit $k \sim \frac{1}{2}\log\left(\frac{1}{\varepsilon}\right)$ Anfragen erreicht werden.

← 28.06.06

(B) Simon Problem

Gegeben

$$f : \{0,1\}^n \longrightarrow \{0,1\}^n \tag{37}$$

mit "Periode" a (n-bit Wort), i.e.

$$f(x) = f(y) \iff y = x \oplus a \tag{38}$$

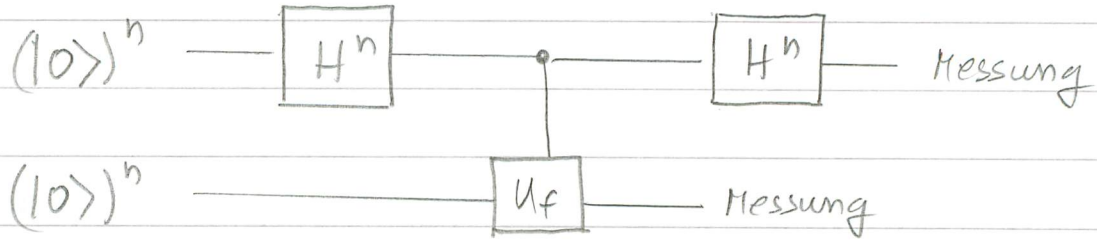
wobei $\oplus$ wieder bitweise Addition modulo 2 (XOR) bedeutet. Die Aufgabe besteht darin a zu finden. Klassisch ist dieses Problem schwer. Wir müssen solange Paare x, y testen bis $x \oplus y = a$. Nehmen wir an wir untersuchen $2^{n/2}$ Paare (x,y). Für jedes Paar ist die Wahrscheinlichkeit, daß $x \oplus y = a$ 2^{-n}, d.h. die Erfolgswahrscheinlichkeit nach $2^{n/2}$ Paaren ist nur exponentiell klein

$$2^{-n}\, 2^{n/2} = 2^{-n/2} \tag{39}$$

Um $2^{n/2}$ Paare $f(x), f(y)$ zu berechnen müssen mindestens

$2^{n/4}$ Auswertungen von f durchgeführt werden.

Quantenmechanisch ist das Problem leicht:



Notation $|0\rangle \equiv |0\rangle^n$,
Gatteroperation

$$|0\rangle|0\rangle \xrightarrow{H^n} \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle|0\rangle$$

(40)

$$\xrightarrow{U_f} \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle$$

Nun können wir das zweite Register messen. Das Messergebnis wird zufällig einen der 2^n möglichen Werte von f liefern. Nehmen wir an die Messung liefert $f(x_0)$. Da sowohl x_0 als auch $x_0 \oplus a$ denselben Wert $f(x_0) = f(x_0 \oplus a)$ geben, hat nach dieser Messung das erste Register den Zustand

(41) $$\frac{1}{\sqrt{2}} \left(|x_0\rangle + |x_0 \oplus a\rangle\right)$$

Um Informationen über a zu bekommen, führen wir ein bitweise Hadamard Transformation des ersten Gatters aus.

$$H^{(n)} \frac{1}{\sqrt{2}} \left(|x_0\rangle + |x_0 \oplus a\rangle \right)$$

$$= \frac{1}{2^{(n+1)/2}} \sum_{y=0}^{2^n-1} \left[(-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus a)\cdot y} \right] |y\rangle$$

$$= \frac{1}{2^{(n+1)/2}} \sum_{y=0}^{2^n-1} (-1)^{x_0 \cdot y} \left[1 + (-1)^{a \cdot y} \right] |y\rangle \qquad (42)$$

$$= \frac{1}{2^{(n-1)/2}} \sum_{a \cdot y = 0} (-1)^{x_0 \cdot y} |y\rangle$$

Messung des $|y\rangle$-Registers projiziert auf einen der 2^{n-1} Zustände mit $a \cdot y = 0$.

Man wird den Algorithmus wiederholt bis wir n linear unabhängige Werte y_i gefunden haben mit

$$y_i \cdot a = 0 \qquad i = 1, 2, \ldots, n \qquad (43)$$

Dann können wir das System aus n Gleichungen eindeutig nach dem n-bit Wort a lösen. (Man beachte $y \cdot a$ ist bitwise UND bzw. Skalarprodukt <u>modulo 2</u>)

Um n linear unabhängige Werte y_i zu finden muß der Algorithmus nur $O(n)$ oft wiederholt werden, d.h. das Problem ist quantenmechanisch mit polynomialen Aufwand lösbar.

D.h. wir haben anhand des Simon'schen Algorithmus gezeigt

> Es gibt ein Orakel f bzgl. welchem $BQP \neq BPP$

(a) Grover Suchalgorithmus

Gegeben eine Ungeordnete Liste mit N Einträgen, Man finde einen bestimmten Eintrag. Beispiel: Man finde den Namen einer Person die eine bestimmte vorgegeben Telephonnummer hat durch Suche im Telephonbuch. Klassisch sind dazu im Mittel $N/2$ Anfragen nötig.

Bem: Suche in einer geordneten Liste führt noch $O(\log N)$ Schritten zum Erfolg (Bisektionsverfahren)

Von Lov Grover stammt ein Algorithmus der es im Mittel in $\sqrt{N}$ Schritten erlaubt den gesuchten Listeneintrag (Name) zu einer gegebenen Variable (Telephonnummer) zu finden.

Das "Telephonbuch" wird dazu mit Hilfe eines Orakels f kodiert. Für eine gegebene Telephonnummer w sei der zugehörige Name x_0. Ist x irgendein Name so liefert das Orakel

$$f_w(x) = \begin{cases} 1 & \text{für } x = x_0 \\ 0 & \text{für } x \neq x_0 \end{cases} \tag{44}$$

f kann nun durch eine quantenmechanisch unitäre Transformation ersetzt werden (quantum oracle)

(45) $$U_\omega: \quad |x\rangle|y\rangle \longrightarrow |x\rangle|y \oplus f_\omega(x)\rangle$$

Wie wir in Abschnitt (A) und (B) gesehen haben gilt

(46) $$U_\omega: |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \longrightarrow (-1)^{f_\omega(x)} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Wenn wir das zweite Register ignorieren ist die Wirkung von U_ω

(47) $$U_\omega |x\rangle \longrightarrow (-1)^{f_\omega(x)} |x\rangle$$

D.h. U_ω spiegelt den Zustandsvektor an der Hyperfläche orthogonal zu $|x_0\rangle$ dadurch das Vorzeichen der Komponente von $|x_0\rangle$ umdreht, während alle anderen unbeeinflußt bleiben, D.h.

(48) $$U_{\omega_0} = \mathbb{1} - 2|x_0\rangle\langle x_0|$$

Wie beim Deutsch-Josza und Simon Algorithmus nutzt auch der Grover Algorithmus das Superpositionsprinzip der Quantenmechanik. Es wird ein Anfangszustand

(50) $$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

präpariert. In $|s\rangle$ ist der unbekannte Zustand $|x_0\rangle$ mit Wahrscheinlichkeitsgewicht $1/N$ enthalten. Grovers

Algorithmus besteht nun in einer iterativen Anwendung der unitären Operation

(51) $$R = U_s U_w$$

wobei

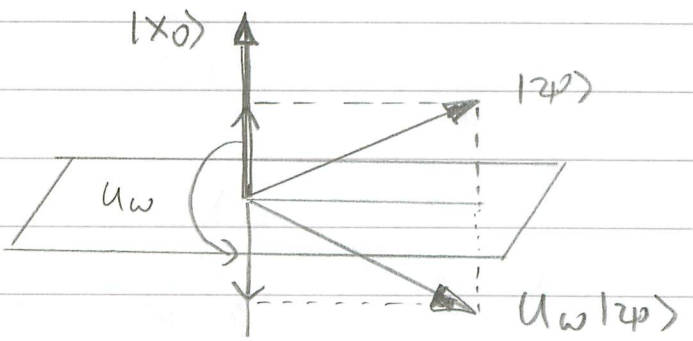
(52) $$U_s = 2|s\rangle\langle s| - 1 = -(1 - 2|s\rangle\langle s|)$$

Die Wirkung von U_s wird auch als "reflection about the mean" bezeichnet.

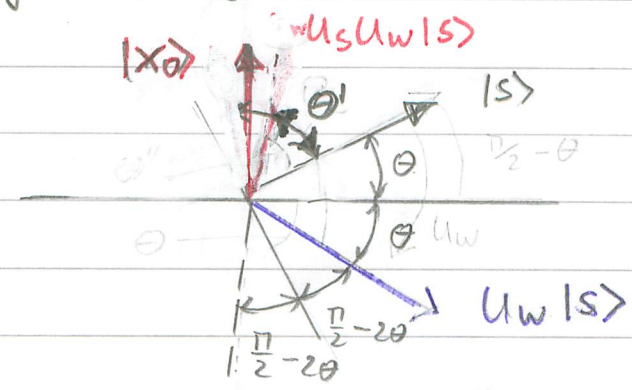
Geometrische Veranschaulichung

(53) $$|\langle s|x_0\rangle| = \frac{1}{\sqrt{N}} \equiv \sin\theta \approx \theta$$

Wirkung von U_w



Wirkung von $U_s U_w$



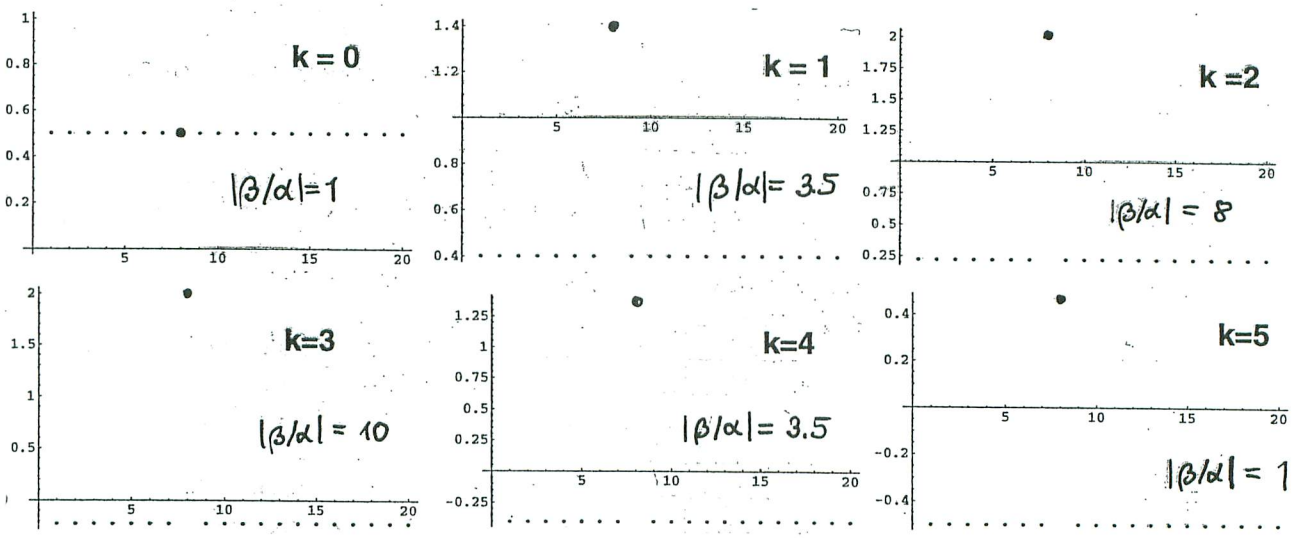
Wahrscheinlichkeit extrahiert werden.

Suche in unsortierter Liste mit $\sqrt{N}$ Schritten erfolgreich

- Grovers Algorithmus ändert nicht die Komplexitätsklasse!

Beispiel

x = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20



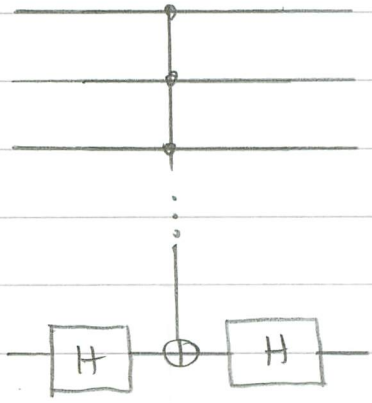
$\sqrt{20} = 4.47$

Implementierung von U_s

(60) $\quad |s\rangle = H^{(n)} |0\rangle$

d.h.

(61) $\quad U_s = 2|s\rangle\langle s| - 1 = H^{(n)} \left(2|0\rangle\langle 0| - 1\right) H^{(n)}$



Implementierung von
$-U_s$

$|111\ldots1\rangle \rightarrow -|111\ldots1\rangle$

alle anderen Zustände
bleiben unberührt

Man kann zeigen, daß dieses Gatter durch $\mathcal{O}(n)$ elementare Gatter konstruierbar ist, d.h. Grovers Algorithmus erfordert von der Ordnung $\sqrt{N}$ poly $(n = \log N)$ Operationen elementarer Gatter.

Man kann zeigen, daß Grovers Algorithmus optimal ist.

(D) Quantenfouriertransformation und Shors Faktorisierungsalgorithmus

Gegeben eine Funktion f

(62) $$f: \{0,1\}^n \longrightarrow \{0,1\}^m$$

mit einer unbekannten Periode r, $1 \ll r \ll 2^n$, d.h.

(63) $$f(x) = f(x+mr) \qquad r, m \in \mathbb{N}$$
$$x, x+mr \in \{0,1,2,3,\ldots,2^n-1\}$$

Die Bestimmung von r ist klassisch ein schweres Problem,

selbst dann wenn die Berechnung von $f(x)$ in $poly(n)$ Schritten machbar ist.

Die zentrale Idee für einen Quantenalgorithmus zur Periodenbestimmung ist die <u>Quanten Fouriertransformation.</u>

(64) $$QFT: |x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i \frac{xy}{N}} |y\rangle$$

wobei $N = 2^n$. Nehmen wir zunächst an, daß wir QFT effizient implementieren können. Wie können wir damit die Periode r einer unbekannten Funktion f bestimmen? Analog zum Simon Problem

(65)
$$|0\rangle^n |0\rangle^n \xrightarrow{H^{(n)}} \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |0\rangle^n$$
$$\xrightarrow{U_f} \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle |f(x)\rangle$$

Messung des zweiten Registers mit Ergebnis $f(x_0)$ mit $0 \leq x_0 < r$ projiziert erstes Register auf

(66) $$\frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle \qquad A \in \{q\,1,2,3,\ldots,2^n-1\}$$

wobei

(67) $$N - r \leq x_0 + (A-1)r < N$$

oder

(68) $$A - 1 < \frac{N}{r} < A + 1$$

Falls N/r natürliche Zahl $\Rightarrow$ $A = N/r$.

Problem: Bestimmung von r aus (66). Direkte Projektion von (66) auf computational basis liefert keine Information über r. Stattdessen wenden wir zunächst QFT an

$$\text{QFT}\ \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle = \frac{1}{\sqrt{NA}} \sum_{y=0}^{N-1} e^{2\pi i \frac{x_0 y}{N}}$$

(69)
$$\cdot \sum_{j=0}^{A-1} e^{2\pi i \frac{jry}{N}} |y\rangle$$

Messung in computational basis liefert nun y mit Wahrscheinlichkeit (die unbekannte x_0 taucht nicht mehr auf!)

(70)
$$p(y) = \frac{A}{N} \left| \frac{1}{A} \sum_{j=0}^{A-1} e^{2\pi i \frac{jry}{N}} \right|^2$$

$p(y)$ groß falls $\frac{ry}{N}$ natürliche Zahl.

z.B. falls N/r natürliche Zahl, d.h. $N/r = A$

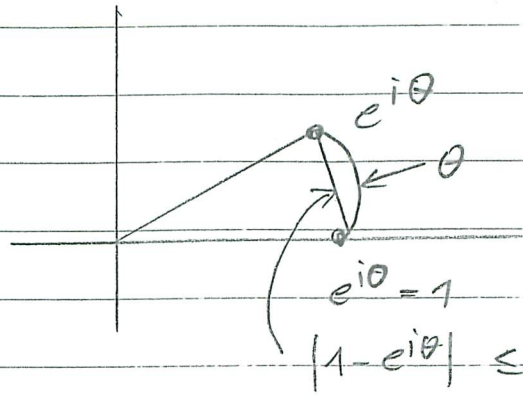
$\Rightarrow$

(71)
$$p(y) = \frac{1}{r} \left| \frac{1}{A} \sum_{j=0}^{A-1} e^{2\pi i \frac{jy}{A}} \right|^2 = \begin{cases} \frac{1}{r} & y = mA \\ 0 & \text{sonst} \end{cases}$$

Allgemein hat man

(72)
$$\sum_{j=0}^{A-1} e^{ij\theta_y} = \frac{e^{iA\theta_y} - 1}{e^{i\theta_y} - 1}$$

wobei



$|1 - e^{i\theta}| \leq |\theta|$

(73) $$\theta_y = \frac{2\pi r y \pmod N}{N} \qquad \left(A = \frac{N}{r}\right)$$

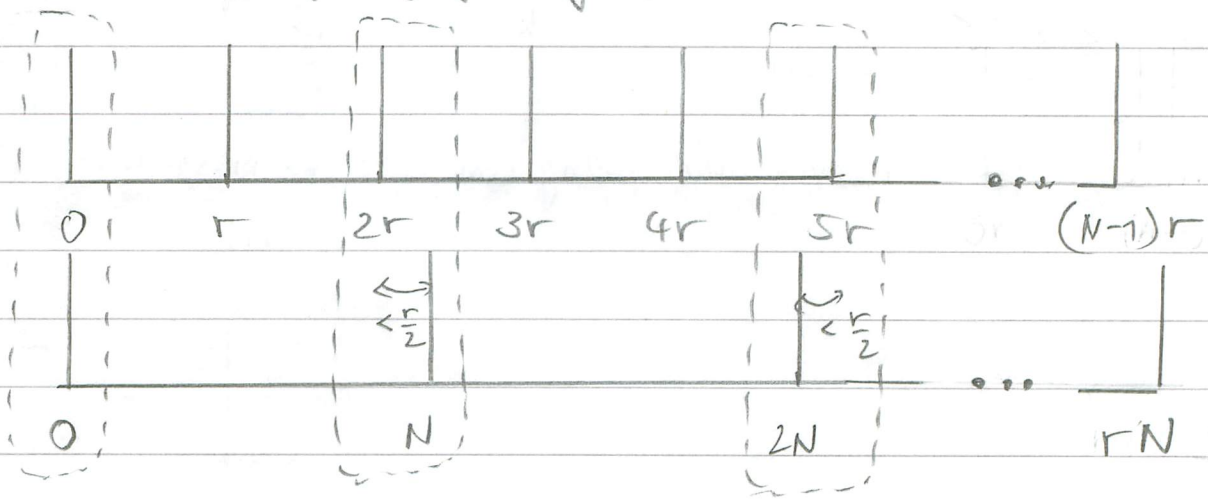
bedeutet. Für welche Werte y gibt die Summe (72) einen großen Beitrag? Da $A - 1 < \frac{N}{r}$ liegen alle komplexen Summanden $e^{ij\theta_y}$ in (72) auf der selben Halbebene wenn

(74) $$-\pi \frac{r}{N} \leq \theta_y \leq \pi \frac{r}{N}$$

In diesem Fall wird die Summe extremal. (74) ist gleichbedeutend mit

(75) $$-\frac{r}{2} \leq yr \pmod N \leq \frac{r}{2}$$

Es gibt genau r Werte y im Bereich $\{0, 1, \ldots, N-1\}$ die dieser Bedingung genügen



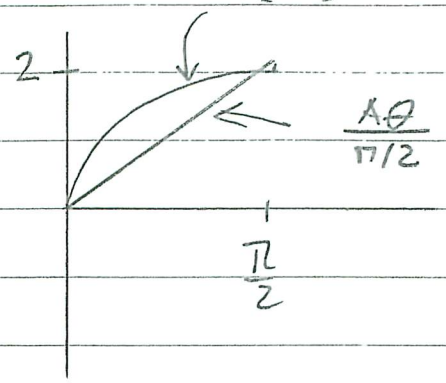
$y=0, y=2, y=5, \ldots$ erfüllen (75) $\Rightarrow \exists$ r Werte y

Nun gilt

(76) $$|1 - e^{i\theta}| \leq |\theta|$$

$$|1 - e^{i\Delta\theta}| = |e^{i\theta\Delta/2}| \, 2\sin(\Delta\theta)$$

$$= 2\sin(\Delta\theta)$$



$$y - \frac{1}{2} \leq k\frac{N}{r} \leq y + \frac{1}{2}$$

$$\frac{y}{N} - \frac{1}{2N} \leq \frac{k}{r} \leq \frac{y}{N} + \frac{1}{2N} \qquad k = 0, 1, \ldots, r-1$$

$\Rightarrow$ r Versuche nötig um $\frac{k}{r}$ zu bestimmen

Außerdem ist für $A|\theta| \leq \pi$

$$|1 - e^{iA\theta}| \geq \frac{2A|\theta|}{\pi} \tag{77}$$

Das liefert die folgende Abschätzung

$$\left|\frac{e^{iA\theta}-1}{e^{i\theta}-1}\right| \geq \frac{2A|\theta|}{\pi}\frac{1}{|\theta|} = \frac{2A}{\pi} \tag{78}$$

$\Rightarrow$

$$p(y) \geq \frac{A}{N}\frac{4}{\pi^2} \geq \frac{4}{\pi^2}\left(\frac{1}{r} - \frac{1}{N}\right) > \frac{4}{\pi^2}\frac{1}{r} \tag{79}$$

für jeden Wert y für den (75) erfüllt ist. D.h. mit Wahrscheinlichkeit $> 4/\pi^2$ liefert die Messung einen der y-Werte die Gl. (75) erfüllen, d.h. für die

$$k\frac{N}{r} - \frac{1}{2} \leq y \leq k\frac{N}{r} + \frac{1}{2} \tag{80}$$

wobei k eine natürliche Zahl aus $\{0, 1, \ldots, r-1\}$ bedeutet. Aus der Kenntnis von y können wir k/r bestimmen

$$y \longrightarrow k/r \tag{81}$$

Mit hoher Wahrscheinlichkeit sind k und r teilerfremd, so daß wir r bestimmen können. Durch Anwenden des Orakels U_f kann dann leicht überprüft werden ob

$$f(x) = f(x+r) \tag{82}$$

Falls (82) nicht erfüllt ist, können wir benachbarte Werte von y testen. Falls k und r nicht teilerfremd waren (d.h. $f(x) \neq f(x+r)$) verwerfen wir die Rechnung und starten den Algorithmus erneut.

Implementierung der Quantenfouriertransformation

Um zu sehen von welcher Komplexitätsklasse die Periodenbestimmung mittels QFT ist, müssen wir uns fragen mit wievielen elementaren Gatteroperationen sie implementierbar ist. F-Trafo:

$$(83) \qquad \sum_{x=0}^{N-1} f(x)|x\rangle \longrightarrow \sum_{y=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi i \frac{xy}{N}} f(x)|y\rangle$$

entspricht Multiplikation mit $N \times N$ Matrix $\left(e^{\frac{2\pi i}{N}}\right)^{xy}$.
Naiv würde man meinen, daß dies $O(N^2)$ d.h. $O(2^{2n})$ elementare Operationen erfordert. Dies ist jedoch nicht der Fall. Auf klassischer Ebene existiert das Verfahren der FFT das $O(N \log N)$ Schritte benötigt. Mit $N = 2^n$

$$(84) \qquad x = x_{n-1} \cdot 2^{n-1} + x_{n-2} 2^{n-2} + \cdots + x_1 \cdot 2 + x_0$$
$$y = y_{n-1} \cdot 2^{n-1} + y_{n-2} 2^{n-2} + \cdots + y_1 \cdot 2 + y_0$$

Binärdarstellung von x, y. Zur Berechnung von $\left(e^{\frac{2\pi i}{N}}\right)^{xy}$ können alle Terme in $x \cdot y$ die Vielfache von 2^n sind ignoriert werden. D.h. für die Auswertung von $\left(e^{\frac{2\pi i}{N}}\right)^{x \cdot y}$ gilt die Äquivalenz

$$\frac{x \cdot y}{N} = \frac{x \cdot y}{2^n} \;\hat{=}\; y_{n-1}\frac{x_0}{2} + y_{n-2}\left(\frac{x_1}{2^2} + \frac{x_0}{2^2}\right)$$

(85)

$$+ \cdots + y_0\left(\frac{x_{n-1}}{2} + \frac{x_{n-2}}{2^2} + \cdots \frac{x_0}{2^n}\right)$$

Die Berechnung dieses Ausdrucks erfordert $\mathcal{O}(n)$ Schritte. D.h. die Fast-Fourier Transformation

(86)
$$\tilde{f}(x) = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N} f(y)$$

kann für jeden der N Werte von x in $\mathcal{O}(n = \log N)$ Schritten berechnet werden. Leider ändert die FFT nicht die Komplexitätsklasse des Problems da $N = 2^n$ Werte von x ausgewertet werden müssen.

Mit Hilfe des Quantenparallelismus ist dies wesentlich effizienter implementierbar. Gemäß (85) soll $(N = 2^n)$

$$QFT : |x\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{y=1}^{N-1} e^{\frac{2\pi i x y}{N}} |y\rangle$$

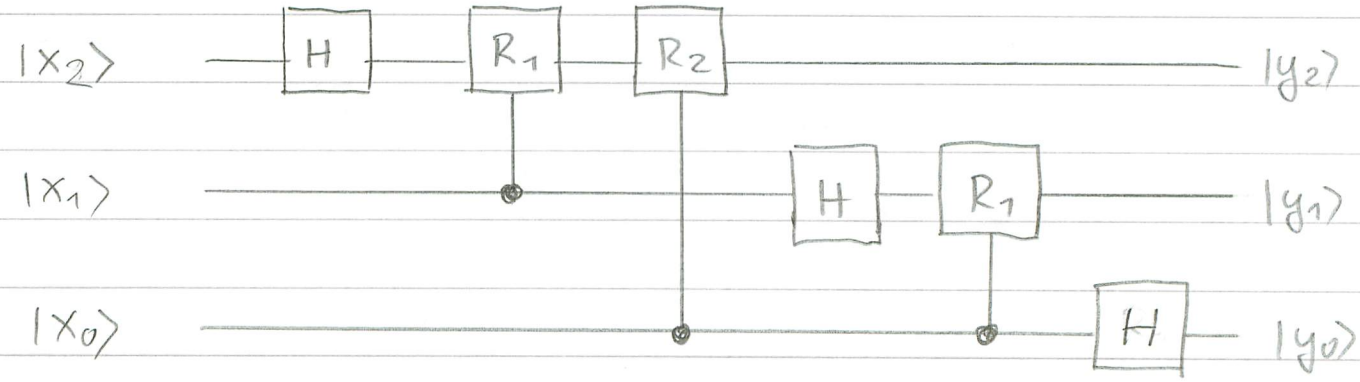
(87)
$$= \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i [x_0]} |1\rangle\right)\left(|0\rangle + e^{2\pi i [x_0 x_1]} |1\rangle\right)$$
$$\cdots \left(|0\rangle + e^{2\pi i [x_0 x_1 \cdots x_{n-1}]} |1\rangle\right)$$

wobei

(88)
$$[x_0 x_1 \cdots x_k] = \frac{x_k}{2} + \frac{x_{k-1}}{2^2} + \cdots \frac{x_0}{2^{k+1}}$$

(87) kann effizient implementiert werden.

Bspl: $n = 3$



(89) $$H|x_0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i [x_0]}|1\rangle\right)$$

(90) $$R_d = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2^d} \end{pmatrix}$$

Hier werden je Input bit ein Hadamard und $n(n-1)/2$ 2-Gatter benötigt. Allgemein ist der Aufwand zur Implementierung der QFT von der Ordnung n^2 d.h. $\mathcal{O}((\log N)^2)$.

> Ein Quantenrechner kann die Periode einer Funktion $f: \mathbb{Z}_N \to \mathbb{Z}_N$ in $\mathcal{O}((\log N)^2)$ Schritten bestimmen. Ein klassischer Algorithmus benötigt $\mathcal{O}(N \log N)$ Schritte.

exponentielle Beschleunigung!

Periodenbestimmung und Faktorisierung

Wir wollen nun argumentieren, daß eine effiziente Periodenbestimmung zum Auffinden von Primfaktoren großer Zahlen genutzt werden kann.

Gegeben N, gesucht $k \neq 1, N$ mit $N = k \cdot r$ (N, k, r natürliche Zahlen). Aufwand des besten bekannten klass. Zahlensiebes: $\mathcal{O}\left[\exp\left((\log N)^{\frac{1}{3}} (\log\log N)^{\frac{2}{3}}\right)\right]$
$\Rightarrow$ "P" Problem.

(i) Man wähle $a < N$ zufällig aus

(ii) Man bestimme den größten gemeinsamen Teiler von a und N, z.B. mit dem Euklid Algorithmus mit Aufwand $\text{poly}(\log N)$. Falls ein solcher existiert ist man fertig. Viel Wahrscheinlicher ist jedoch, daß ein solcher nicht existiert.

(iii) Falls a und N teilerfremd folgt nach dem Theorem von Euler: $\exists$ eine Zahl r mit

$$a^r = 1 \quad (\text{modulo } N) \tag{91}$$

(iv) Der Wert r aus (iii) ist die Periode der Funktion

$$f_{N,a}(x) = a^x \quad (\text{modulo } N) \tag{92}$$

Diese Periode kann effizient mittels QFT bestimmt werden mit Aufwand $O((\log N)^2)$

(v) Falls r aus (iv) gerade $\Rightarrow$

(93) $$\underbrace{(a^{r/2}-1)}_{=\alpha}\underbrace{(a^{r/2}+1)}_{=\beta} = a^r - 1 = 0 \quad \text{modulo } N$$

D.h. N hat einen gemeinsamen Faktor mit $\alpha\beta$. Bestimmung des größten gemeinsamen Teilers von α und N bzw. von β und N liefert einen Faktor von N. Dies ist mittels des Euklidschen Verfahrens effizient durchführbar und entweder $\{\alpha, N\}$ oder $\{\beta, N\}$ haben einen größten gemeinsamen Teiler.

Falls r aus (iv) ungerade $\Rightarrow$ Wiederholung des Algorithmus mit anderer Zahl a.

Die Bestimmung der ganzzahligen Faktoren einer ganzen Zahl N ist mittels eines Quantenalgorithmusses in $\text{poly}(\log N)$ Schritten möglich!

Faktorisierung $\in$ BQP

Der RSA Kode der klassischen Kryptographie ist nicht sicher!