

Quantencomputing

Modul 7271

Martin Rehberg

DB Systel GmbH / Hochschule RheinMain

Inhaltsverzeichnis

1 Grundlagen

- Einleitung & Ein-Qubit Systeme
- n -Qubit Systeme & Grundlegende Schaltkreise
- Verschränkung, Deutsch-Jozsa, Bernstein-Vazirani & Simon

2 Quantensuche

- Algorithmus von Grover
- Varianten der Quantensuche

3 Fouriertransformation & Shor

- RSA & periodische Funktionen
- Diskrete- & Quanten-Fouriertransformation
- Algorithmus von Shor

4 Ausblick: QEC & QKD

Inhaltsverzeichnis

1 Grundlagen

- Einleitung & Ein-Qubit Systeme
- n -Qubit Systeme & Grundlegende Schaltkreise
- Verschränkung, Deutsch-Jozsa, Bernstein-Vazirani & Simon

2 Quantensuche

- Algorithmus von Grover
- Varianten der Quantensuche

3 Fouriertransformation & Shor

- RSA & periodische Funktionen
- Diskrete- & Quanten-Fouriertransformation
- Algorithmus von Shor

4 Ausblick: QEC & QKD

Einleitung

Themengebiet Quantencomputing:

- Quantenalgorithmen entwerfen
- Quantenalgorithmen untersuchen (insb. Laufzeit)
- Quantenkomplexitätstheorie

Einleitung

Themengebiet Quantencomputing:

- Quantenalgorithmen entwerfen
- Quantenalgorithmen untersuchen (insb. Laufzeit)
- Quantenkomplexitätstheorie

Ziel der Vorlesung:

- grundlegende Algorithmen auf Quantencomputern verstehen
 - physikalischen Grundlagen als *gegeben* annehmen
 - Mathematik werden wir nach Bedarf erarbeiten
- Anwendungen, insb. mit Blick auf Verschlüsselungsverfahren
- Ausblick: Quantenfehlerkorrektur, Quantum Key Distribution

Einleitung

Literatur:¹

- Matthias Hanneister - Quantencomputing verstehen (**Hauptquelle**), 5. Auflage, Springer, 2018.
- Artuhr Pittenger - An Introduction to Quantum Computing Algorithms, Birkhäuser, 2001.
- Michael Nielsen, Isaac Chuang - Quantum Computation and Quantum Information, 10. Auflage, Cambridge University Press, 2010.
- Wolfgang Scherer - Mathematics of Quantum Computing, 2. Auflage, Springer, 2019.

¹ verwendete Grafiken sind allesamt dem Buch von M. Hohmeister oder Wikipedia (public domain) entnommen

Einleitung

Klassische Welt

- mechanische Rechenmaschinen
 - Difference Engine (Auswertung von Polynomen), Analytical Engine (Gleichungssysteme lösen) - Charles Babbage
 - Schachmaschine - Leonardo Quevedo
- elektromechanische Rechenmaschinen
 - Z3 (Gleitkommaarithmetik), Z4 - Konrad Zuse
 - Kryptoanalyse (Lorenz-Schlüsselmaschine) - Colossus
- *moderne* Rechenmaschinen

Einleitung

Beobachtung

Ein **klassisches Bits** kann genau zwei unterschiedliche Zustände annehmen: 0 und 1. Sie haben zwei wesentliche Eigenschaften

- **Realismus:** Der Wert eines Bits ist zu jedem Zeitpunkt der Berechnung eindeutig bestimmt, d.h. entweder 0 oder 1. Er kann ausgelesen werden und der Prozess des Auslesens ändert den Wert des Bits nicht.
- **Lokalität:** Wird der Wert eines bestimmten einzelnen Bits verändert, so ändert das nicht den Wert *irgendeines* anderen Bits.

Einleitung

Quantenwelt

- Quantencomputer rechnen mit Quantenbits
- Quantenbits folgen den Gesetzen der Quantenmechanik (und können bspw. nicht *kopiert* werden)
- Quantenbits sind in einem Zustand der *Superposition*, d.h. sind von der Form $\alpha|0\rangle + \beta|1\rangle$
- Quantenbits können in einem *verschränkten* Zustand sein

Einleitung

Beobachtung

Ein **Quantenbit** ist in einem Zustand der Superposition. Im Vergleich zum klassischen Bit stellen wir fest:

- **Veränderung beim Messen:** Wird ein Quantenbit gemessen, so wird der Zustand der Superposition aufgehoben und das Quantenbit wechselt in einen der beiden (klassischen) Zustände 0 oder 1. Durch den Messvorgang wird das Quantenbit mit dem entsprechenden Werte 0 oder 1 überschrieben.
- **Verschränkung:** Die Veränderung eines Quantenbits kann unmittelbar (also im selben Augenblick) die Eigenschaft eines anderen Quantenbits verändern.

Einleitung

Quantencomputing hat bereits weitreichende Folgen

- Kryptographie (RSA & ECDH)
- Optimierungsverfahren & ML (Logistik & Transport)
- Chemie- & Pharmaindustrie (Material & Medikamente)

Einleitung

Quantencomputing hat bereits weitreichende Folgen

- Kryptographie (RSA & ECDH)
- Optimierungsverfahren & ML (Logistik & Transport)
- Chemie- & Pharmaindustrie (Material & Medikamente)

Es gibt aber nicht nur Vorteile:

- No-Cloning Theorem
- Quantencomputer können **NP-vollständige Probleme** (vermutlich) nicht effizient lösen
- Fehlerkorrektur

Einleitung - Komplexität

Rückblick & Ausblick Schwierigkeit von Berechnungsprobleme

Beispiel: Zur Addition zweier n -Bit Zahlen mittel *Carry-Ripple Addierer* wird ein Halbaddierer (ein XOR, ein AND) und $n - 1$ Volladdierer (zwei XOR, zwei AND, ein OR) benötigt; insgesamt $5n - 3$ logische Gatter. Wächst die Länge der binären Zahlen, dann auch die Anzahl der logischen Gatter *linear* in n .

Landau Notation

Sei $f, g : \mathbb{N} \rightarrow \mathbb{N}$. Wir schreiben $f(n) = O(g(n))$, wenn eine Konstante $c > 0$ und ein $n_0 \in \mathbb{N}$ existiert, sodass $f(n) \leq c \cdot g(n)$ für alle $n \geq n_0$ gilt.

Hier, $5n - 3 = O(n^2)$ für alle $n \geq 5$.

Einleitung - Komplexität

Die Anwendung eines logischen Gatter benötigt eine gewisse Zeit.
Von einem Algorithmus der $O(n)$ logische Gatter benötigt, sagen wir er läuft in **Zeit** $O(n)$.

Ein Algorithmus ist **effizient**, wenn seine Laufzeit (in Abhängigkeit der Inputgröße) durch ein Polynom beschrieben werden kann. Die Anzahl der Gatter skaliert **polynomial** in n , etwa $5n - 3$, n^2 , \sqrt{n} oder n^{100} .

Ein Algorithmus ist **ineffizient**, wenn er mehr als polynomielle Laufzeit benötigt, etwa

- **exponentielle Laufzeit**, wie 2^n oder $\exp\left(\frac{n}{1000}\right)$
- **superpolynomielle Laufzeit**, d.h. weniger als exponentielle Zeit aber mehr als polynomielle Zeit, wie $2^{n^{1/3}}$

Einleitung - Komplexität

Probleme die von einem klassischen Computer effizient (d.h. in Polynomialzeit) gelöst werden können, ordnen wir der Komplexitätsklasse **P** zu.

Beispiel:

- Addition zweier Zahlen
- Bestimmung ob eine Zahl eine Primzahl ist

Einleitung - Komplexität

Probleme für die eine gegebene Lösung effizient überprüft werden kann, werden der Komplexitätsklasse **NP** zugeordnet.

Beispiel:

- Faktorisierung einer Zahl
- Überprüfung ob zwei gegebene Graphen äquivalent sind

Einleitung - Komplexität

Probleme für die eine gegebene Lösung effizient überprüft werden kann, werden der Komplexitätsklasse **NP** zugeordnet.

Beispiel:

- Faktorisierung einer Zahl
- Überprüfung ob zwei gegebene Graphen äquivalent sind

Unter den Problemen in **NP** gibt es besondere Probleme, die wir **NP-vollständig** nennen. Können wir ein **NP-vollständiges** Problem effizient lösen, dann können wir **jedes** Problem in **NP** effizient lösen. Dazu gehören

- TSP (Traveling Salesperson Problem)
- spieltheoretische Probleme wie $n \times n$ Sudoku

Einleitung - Komplexität

Eine speicherbezogene Klasse ist **PSPACE**, die alle Probleme beinhaltet die auf einem klassischen Computer mit polynomial viel Speicher gelöst werden können, wobei an die Zeit keine Schranken vorgegeben werden.

Beispiel: Gewinnstrategie für Zwei-Personen Spiele wie Dame auf einem $n \times n$ Spielbrett.

Es gilt

- $\mathbf{P} \subset \mathbf{NP}$
- Umkehrung: \mathbf{P} vs. \mathbf{NP} Problem
- $\mathbf{NP} \subset \mathbf{PSPACE}$
- Vermutung: $\mathbf{P} \neq \mathbf{NP}$, $\mathbf{P} \neq \mathbf{PSPACE}$, $\mathbf{NP} \neq \mathbf{PSPACE}$

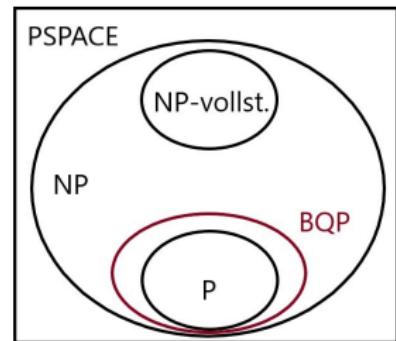
Einleitung - Komplexität

Die (begründete) Hoffnung ist das Quantencomputer Probleme effizient lösen können, die auf einem klassischen Computer nicht effizient gelöst werden können.

Dafür spricht

- Algorithmus von Shor
- Random Circuit Sampling

Ein Quantencomputer liefert in beiden Fällen superpolynomielle Beschleunigung.

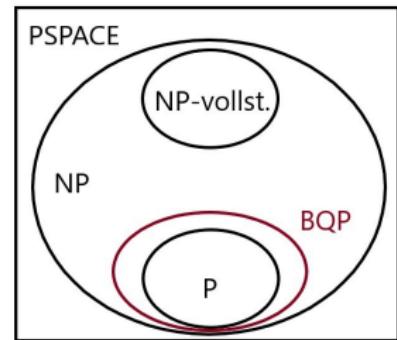


Einleitung - Komplexität

Für andere Probleme liefert ein Quantencomputer eine (beweisbar) polynomielle Beschleunigung (ggf. schon anwendungsrelevant).

Probleme die von einem Quantencomputer effizient gelöst werden können, liegen in **BQP**.
Quantencomputer können klassische Computer effizient simulieren, d.h. $P \subset BQP$.

Welche Probleme liegen in **BQP**?
Wie viel „größer“ als **P** ist **BQP**?

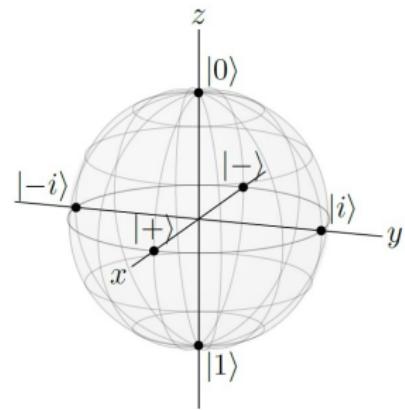


Ein-Qubit Systeme

Ein Qubit kann zwei Grundzustände annehmen: $|0\rangle$ und $|1\rangle$.

Wir identifizieren diese mit dem Nordpol $(1, 0, 0)$ und Südpol $(0, 0, -1)$ der *Blochschen Sphäre*.

Zustände eines Qubits können auch Superpositionen von $|0\rangle$ und $|1\rangle$ sein.

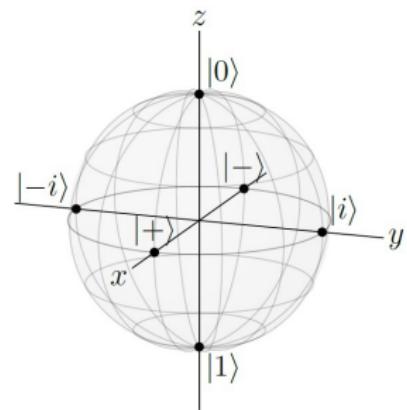


Ein-Qubit Systeme

Einige auf Äquatorebene gegenüberliegende Zustände sind von besonderer Bedeutung:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \\ |i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle), \quad |-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle).$$

Grundsätzlich kann ein Qubit jeder Punkt auf der Blochschen Sphäre sein.



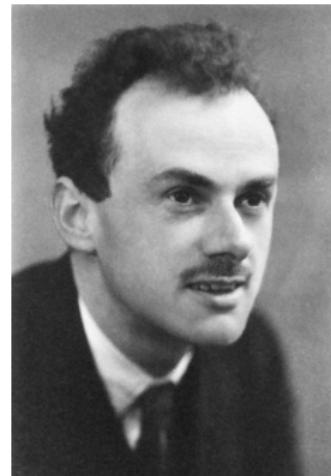
Ein-Qubit Systeme

Für die Beschreibung quantenmechanischer Zustände verwendet man die auf Paul Dirac zurückgehende *ket-Notation*.

Definition (Quantenbit)

Ein Quantenbit (*Qubit*) nimmt Zustände der Form $\alpha|0\rangle + \beta|1\rangle$ mit $\alpha, \beta \in \mathbb{C}$ an. Die Zahlen α, β heißen *Amplituden* und genügen der Bedingung $|\alpha|^2 + |\beta|^2 = 1$.

Klassische Bits: $|0\rangle, |1\rangle$.



Paul Dirac

Ein-Qubit Systeme

Übung:

1. Überprüfen Sie, ob es sich bei $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ und $\frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$ um zulässige Zustände handelt.

Ein-Qubit Systeme

Übung:

1. Überprüfen Sie, ob es sich bei $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ und $\frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$ um zulässige Zustände handelt.
2. Gilt für ein Qubit $|\alpha|^2 + |\beta|^2 = 1$, nennt man es *normalisiert*. Bestimmen Sie die *Normalisierungskonstante* A , sodass $A(\sqrt{2}|0\rangle + i|1\rangle)$ normalisiert ist.

Ein-Qubit Systeme

Übung:

1. Überprüfen Sie, ob es sich bei $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ und $\frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$ um zulässige Zustände handelt.
2. Gilt für ein Qubit $|\alpha|^2 + |\beta|^2 = 1$, nennt man es *normalisiert*. Bestimmen Sie die *Normalisierungskonstante* A , sodass $A(\sqrt{2}|0\rangle + i|1\rangle)$ normalisiert ist.

Während wir den Zustand klassischer Bits durch *lesen* feststellen können, ist das bei Qubits nicht ohne Weiteres möglich. Bei Qubits müssen wir *messen* und das Messergebnis hängt von den Amplituden ab.

Ein-Qubit Systeme

Messen eines Quantenbits

Messen wir ein Qubit im Zustand $\alpha|0\rangle + \beta|1\rangle$, wird die Superposition zerstört. Anschließend ist es mit Wahrscheinlichkeit $|\alpha|^2$ im Zustand $|0\rangle$ und mit Wahrscheinlichkeit $|\beta|^2$ im Zustand $|1\rangle$. Diesen Zustand nach dem Messen können wir beobachten.

Beispiel: Das Qubit $\frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$ ist nach dem Messen mit Wahrscheinlichkeit $1/3$ im Zustand $|0\rangle$ und mit Wahrscheinlichkeit $2/3$ im Zustand $|1\rangle$.

Ein-Qubit Systeme

Übung:

1. Was beobachten Sie beim Messen der Qubits $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ und $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$?
2. Bestimmen Sie die Messwahrscheinlichkeiten des Qubits $\frac{2}{3}|0\rangle + \frac{1-2i}{3}|1\rangle$.

Ein-Qubit Systeme

Übung:

- Was beobachten Sie beim Messen der Qubits $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ und $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$?
- Bestimmen Sie die Messwahrscheinlichkeiten des Qubits $\frac{2}{3}|0\rangle + \frac{1-2i}{3}|1\rangle$.

Wir unterscheiden verschiedene Messbasen (entsprechend der Lage der x, y, z -Achse auf der Blochschen Sphäre):

- Z-Basis (bzw. *computational basis*) $\{|0\rangle, |1\rangle\}$
- X-Basis (bzw. *Hadamard-Basis*) $\{|+\rangle, |-\rangle\}$
- Y-Basis (bzw. *LR-Basis*) $\{|i\rangle, |-i\rangle\}$

Messwahrscheinlichkeiten sind die jeweiligen Amplituden bestimmt.

Ein-Qubit Systeme

Wir bauen unsere vorheriges Beispiel weiter aus und vergleichen für bel. $\theta \in \mathbb{R}$

$$\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle \quad \text{und} \quad e^{i\theta} \left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle \right)$$

bzgl. einer Messung in Z-Basis. Hier ergibt sich für das zweite Qubit $|0\rangle$ bzw. $|1\rangle$ mit einer Wahrscheinlichkeit von

$$\left| e^{i\theta} \cdot \frac{\sqrt{3}}{2} \right|^2 = \frac{3}{4} \quad \text{bzw.} \quad \left| e^{i\theta} \cdot \frac{1}{2} \right|^2 = \frac{1}{4}$$

Ein-Qubit Systeme

denn $|\exp(i\theta)|^2 = 1$ für alle $\theta \in \mathbb{R}$. Die Messwahrscheinlichkeiten unterscheiden sich also nicht von denen für das erste Qubit.

Wir nennen $\exp(i\theta)$ eine *globale Phase*. Beide Qubits repräsentieren denselben Punkt auf der Blochschen Sphäre. Globalen Phasen sind vernachlässigbar und wir identifizieren Qubits die sich nur um eine globale Phase unterscheiden miteinander, und schreiben

$$e^{i\theta} \left(\frac{\sqrt{3}}{2} |0\rangle + \frac{1}{2} |1\rangle \right) \equiv \frac{\sqrt{3}}{2} |0\rangle + \frac{1}{2} |1\rangle.$$

Ein-Qubit Systeme

Globale Phasen sollten nicht mit *relative Phasen* verwechselt werden. Beim Vergleich von

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{2}}|1\rangle)$$

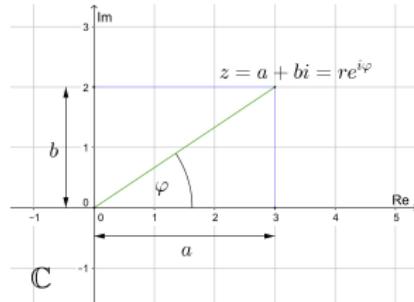
ist $\exp(i\frac{\pi}{2})$ eine relative Phase und die Qubits $|+\rangle, |i\rangle$ gehören zu verschiedenen Punkten auf der Blochschen Sphäre.

Die Darstellung von Qubits auf einer Kugel (*sphärische Koordinaten*) ist eine Erweiterung der aus Analysis bekannten Darstellung mittel Polarkoordinaten.

Ein-Qubit Systeme

Erinnerung: Jede komplexe Zahl $z \in \mathbb{C}$ kann in der Form $z = a + ib$ mit $a, b \in \mathbb{R}$ und $i := \sqrt{-1}$ geschrieben werden. Die Zahl $\bar{z} := a - ib$ heißt die *Konjugierte* von z . Der *Betrag* einer komplexen Zahl ist $|z| := \sqrt{a^2 + b^2}$.

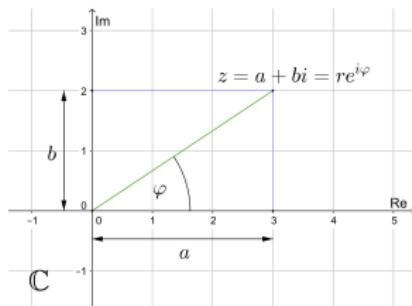
Die *Polarkoordinatendarstellung* einer komplexen Zahl ist $z = re^{i\varphi}$, wobei r der Betrag ist und φ die *Phase*.



Ein-Qubit Systeme

Wissen: Gilt $|z| = |z'|$ für $z \neq z'$ mit $z, z' \in \mathbb{C}$, so unterscheiden sich die komplexen Zahlen nur in der Phase.

Wie selbstverständlich identifizieren wir \mathbb{C} mit \mathbb{R}^2 mittels $1 = (1, 0)$ und $i = (0, 1)$.



Ein-Qubit Systeme

Identifizieren wir $|0\rangle$ mit $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ und $|1\rangle$ mit $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, können wir ein Qubit als Kombination linear unabhängiger Vektoren darstellen:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle.$$

Unter der Bedingung $|\alpha|^2 + |\beta|^2 = 1$ an die Amplituden $\alpha, \beta \in \mathbb{C}$ erhalten wir, dass ein Qubit ein Vektor aus \mathbb{C}^2 der Länge 1 ist.

D.h. die Superposition ist eine *Linearkombination* der klassischen (nicht überlagerten) Zustände $|0\rangle$ und $|1\rangle$.

Achtung: $\alpha, \beta \in \mathbb{C}$, d.h. wie befinden uns im \mathbb{C}^2 .

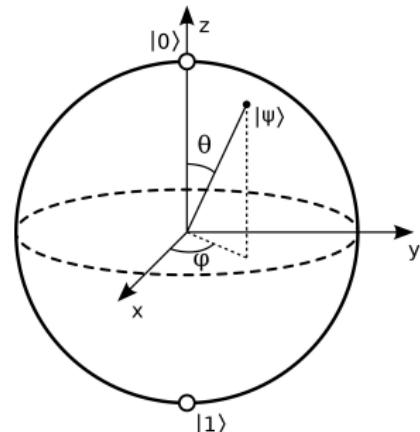
Ein-Qubit Systeme

Mittels

$$\alpha = \cos \frac{\theta}{2}, \quad \beta = e^{i\varphi} \sin \frac{\theta}{2}$$

können wir uns das Qubit
 $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ auf der
Blochschen Sphäre veranschaulichen.

Rechenoperationen auf einem Qubit
sind Rotationen auf der Blochschen
Sphäre.



Ein-Qubit Systeme

Wir wollen verstehen wie man

- Rechenoperationen auf einem Ein-Qubit System ausführt
- Register von mehreren Qubits generiert
- mit *Quantengattern* Operationen auf mehreren Qubits ausführt
- *Quantenalgorithmen* (Schaltkreise) entwickeln
- Register von Qubits misst

und benötigen dafür einen methodischen *Werkzeugkasten* (hier:
Lineare Algebra).

Lineare Algebra

Wir wiederholen einige Grundlagen aus der Linearen Algebra und machen uns mit der Notation im Kontext von Qubits vertraut.

Die Schreibweise $|0\rangle$ und $|1\rangle$ für Qubits ist eine abkürzende Notation für Spaltenvektoren (hier: Basisvektoren), d.h.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{und} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Ein (generisches) Qubit $|\psi\rangle$ mit Amplituden α, β ist demnach ein Vektor

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Lineare Algebra

Beispiel:

$$|i\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{i}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix}$$

Die Transponierte eines Spaltenvektors ist ein Zeilenvektor

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}^T = (\alpha \ \beta)$$

und die Konjugierte des transponierten Vektors ist

$$\left(\begin{pmatrix} \alpha \\ \beta \end{pmatrix}^T \right)^\dagger = (\bar{\alpha} \ \bar{\beta}).$$

Dagger
Transponieren
+
Konjugieren
=>Dagger-Notation

Lineare Algebra

// Komplexe Zahl $z = a + ib$
 $\bar{z} = a - ib$

Letzteres wird so häufig verwendet, dass es im Rahmen der bra-ket Notation eine eigene Symbolik hat

$$\langle \psi | = (\overline{\alpha} \quad \overline{\beta}) .$$

Beispiel:

$$| i \rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix} \quad \text{und} \quad \langle i | = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} \end{pmatrix}$$

Lineare Algebra

Für $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ und $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ ist $\langle 0| = (1 \ 0)$ und $\langle 1| = (0 \ 1)$, d.h.

$$\langle\psi| = (\bar{\alpha} \ \bar{\beta}) = \bar{\alpha}(1 \ 0) + \bar{\beta}(0 \ 1) = \bar{\alpha}\langle 0| + \bar{\beta}\langle 1|.$$

Beispiel:

$$|i\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle \quad \text{dann} \quad \langle i| = \frac{1}{\sqrt{2}}\langle 0| - \frac{i}{\sqrt{2}}\langle 1|$$

Lineare Algebra

Wir fassen unsere Beobachtungen zusammen:

bra-ket Notation

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \iff \langle\psi| = (\bar{\alpha} \quad \bar{\beta})$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \iff \langle\psi| = \bar{\alpha}\langle 0| + \bar{\beta}\langle 1|$$

Lineare Algebra

Wir fassen unsere Beobachtungen zusammen:

bra-ket Notation

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \iff \langle\psi| = (\bar{\alpha} \quad \bar{\beta})$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \iff \langle\psi| = \bar{\alpha}\langle 0| + \bar{\beta}\langle 1|$$

Zur Produktbildung von $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ und $|\phi\rangle = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}$ ist das innere Produkt $\langle\psi|\phi\rangle$ (auch *bra-ket* genannt) eine Möglichkeit:
Skalarprodukt

$$\langle\psi|\phi\rangle = (\bar{\alpha} \quad \bar{\beta}) \cdot \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \bar{\alpha}\gamma + \bar{\beta}\delta.$$

Lineare Algebra

(Konjugieren von komplexen Zahlen) // $\overline{z_1+z_2} = \overline{z_1} + \overline{z_2}$
 $\overline{z_1 z_2} = \overline{z_1} \cdot \overline{z_2}$

Für dieses gilt $\langle \phi | \psi \rangle = \overline{\langle \psi | \phi \rangle}$, denn

$$\langle \phi | \psi \rangle = \bar{\gamma}\alpha + \bar{\delta}\beta = \overline{\bar{\alpha}\gamma + \bar{\beta}\delta} = \overline{\langle \psi | \phi \rangle}.$$

Darüber hinaus ist das innere Produkt von $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ mit sich selbst

$$\langle \psi | \psi \rangle = |\alpha|^2 + |\beta|^2 = 1$$

In diesem Sinne kann $\langle \psi | \psi \rangle$ als Gesamtwahrscheinlichkeit interpretiert werden und $|\psi\rangle$ ist normalisiert, wenn $\langle \psi | \psi \rangle = 1$ gilt.

Lineare Algebra

Für das innere Produkt der verschiedenen Basen gilt

$$\langle 0|1\rangle = 0, \quad \langle +|-\rangle = 0, \quad \langle i|-i\rangle = 0.$$

Zustände (bzw. Vektoren) mit verschwindendem inneren Produkt nennen wir *orthogonal*.

Sind Zustände normalisiert und orthogonal, heißen sie *orthonormal*.

Also sind $|0\rangle, |1\rangle$ orthonormal, ebenso $|+\rangle, |-\rangle$ und $|i\rangle, |-i\rangle$.

Das innere Produkt kann auch verwendet werden um die Amplituden zu beschreiben bzw. zu bestimmen.

Lineare Algebra

Beispiel: Wir betrachten aus früherem Beispiel $|\psi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$ und messen bzgl. der Z-Basis $\{|0\rangle, |1\rangle\}$. Die Amplitude von $|0\rangle$ ist

$$\langle 0|\psi\rangle = \langle 0| \left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle \right) = \frac{\sqrt{3}}{2}\langle 0|0\rangle + \frac{1}{2}\underbrace{\langle 0|1\rangle}_{\text{verschwindet, weil orthonormal}} = \frac{\sqrt{3}}{2}$$

und analog für $|1\rangle$ ist

$$\langle 1|\psi\rangle = \frac{\sqrt{3}}{2}\underbrace{\langle 1|0\rangle}_{\text{verschwindet, weil orthonormal}} + \frac{1}{2}\langle 1|1\rangle = \frac{1}{2}.$$

Die Messwahrscheinlichkeiten sind $|\frac{\sqrt{3}}{2}|^2 = \frac{3}{4}$ und $|\frac{1}{2}|^2 = \frac{1}{4}$. Ebenso erkennen wir

$$|\psi\rangle = \langle 0|\psi\rangle|0\rangle + \langle 1|\psi\rangle|1\rangle.$$

Lineare Algebra

Das war nur bedingt hilfreich und hätte auch durch ablesen festgestellt werden können.

Übung: Messen Sie $|\psi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$ bzgl. der X-Basis $\{|+\rangle, |-\rangle\}$.

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \langle +| &= \frac{1}{\sqrt{2}}(\langle 0| + \langle 1|) \\ &= \frac{1}{\sqrt{2}}\langle 0| + \frac{1}{\sqrt{2}}\langle 1| \end{aligned}$$

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\langle -| = \frac{1}{\sqrt{2}}\langle 0| - \frac{1}{\sqrt{2}}\langle 1|$$

$$\begin{aligned} \langle +|\psi\rangle &= \langle +\left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle\right) \\ &= \frac{\sqrt{3}}{2}\langle +|0\rangle + \frac{1}{2}\langle +|1\rangle \\ &= \frac{\sqrt{3}}{2}\left(\frac{1}{\sqrt{2}}(\langle 0| + \langle 1|)|0\rangle + \right. \\ &\quad \left.\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(\langle 0| + \langle 1|)|1\rangle\right)\right) \\ &= \frac{\sqrt{3}}{2}\left(\frac{1}{2}\cdot\langle 0|0\rangle + \frac{1}{2}\cdot\langle 1|0\rangle + \right. \\ &\quad \left.\frac{1}{2}\cdot\frac{1}{\sqrt{2}}\cdot\langle 0|1\rangle + \frac{1}{2}\cdot\frac{1}{\sqrt{2}}\cdot\langle 1|1\rangle\right) \end{aligned}$$

Lineare Algebra

Das war nur bedingt hilfreich und hätte auch durch ablesen festgestellt werden können.

Übung: Messen Sie $|\psi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$ bzgl. der X-Basis $\{|+\rangle, |-\rangle\}$.

Darstellung bzgl. innerem Produkt

Für eine Orthonormalbasis $\{|a\rangle, |b\rangle\}$ kann der Zustand eines Qubits $|\psi\rangle$ geschrieben werden als $|\psi\rangle = \alpha|a\rangle + \beta|b\rangle$ für $\alpha = \langle a|\psi\rangle$ und $\beta = \langle b|\psi\rangle$.

$\langle a|\psi\rangle$ heißt auch *Projektion* von $|\psi\rangle$ auf $|a\rangle$.

Operationen auf Qubits

Logische Gatter operieren auf Bits. Von einem *Quantengatter* erwarten wir eine Operation, die ein Qubits in ein (verändertes) Qubit überführt, insbesondere also die Nebenbedingung an die Amplituden respektiert.

Für $|0\rangle$ und $|1\rangle$ und ein Quantengatter U erwarten wir

$$U|0\rangle = a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$$

$$U|1\rangle = c|0\rangle + d|1\rangle = \begin{pmatrix} c \\ d \end{pmatrix}$$

Operationen auf Qubits

Da wir Operationen auf einer Basis betrachten, resultiert

$$U = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

mit

$$U|0\rangle = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$$

$$U|1\rangle = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}$$

Operationen auf Qubits

Betrachten wir die Wirkung von U auf einem Qubit
 $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, dann ist

$$\begin{aligned} U|\psi\rangle &= \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} a\alpha + c\beta \\ b\alpha + d\beta \end{pmatrix} \\ &= (a\alpha + c\beta)|0\rangle + (b\alpha + d\beta)|1\rangle \end{aligned}$$

Dies ist ein zulässiges Qubit, falls

$$|a\alpha + c\beta|^2 + |b\alpha + d\beta|^2 = 1$$

gilt. In Fall von Operationen auf Qubits läuft es auf *unitäre* Matrizen hinaus.²

²Wir verzichten hier auf die mathematische Herleitung.

Operationen auf Qubits

Grundlegende Gatter, die auf einem Qubit operieren, sind

Gatter	Matrix	Gatter	Matrix
Identität	$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	Phase S	$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
Pauli X	$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	T	$T = \begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{i\pi}{4}\right) \end{pmatrix}$
Pauli Y	$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Hadamard H	$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Pauli Z	$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	-	-

Operationen auf Qubits

Allgemeiner: Rechenschritte auf Qubits

Definition (transponierte Matrix)

Sei

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

eine Matrix (mit komplexen Einträgen), dann heißt

$$A^T := \begin{pmatrix} a_{11} & \dots & a_{m1} \\ \vdots & & \vdots \\ a_{1n} & \dots & a_{mn} \end{pmatrix}$$

die *Transponierte* von A.

Operationen auf Qubits

Aus den Überlegungen zu Vektoren kennen wir im Prinzip schon

Definition (konjugierte und adjungierte Matrix)

Sei $A = (a_{ij}) \in \mathbb{C}^{m \times n}$. Die Matrix $\bar{A} := (\bar{a}_{ij}) \in \mathbb{C}^{m \times n}$ heißt die *Konjugierte* von A , und $A^\dagger := (\bar{A})^T$ die *Adjungierte* von A .

Definition (unitäre Matrix)

Eine Matrix $A = (a_{ij}) \in \mathbb{C}^{n \times n}$ heißt *unitär*, wenn $A^\dagger = A^{-1}$ gilt.

Es folgt, dass unitäre Matrizen / Quantengatter notwendig *invertierbar* sind, denn nach Definition gilt $A^\dagger A = AA^\dagger = I_n$. Insbesondere sind *reversible logische Gatter* auch zulässige Quantengatter.

Operationen auf Qubits

Wie schon bei der Operation eines Quantengatters auf einem einzelnen Qubit gesehen, wirkt ein Quantengatter (Matrix) durch Multiplikation von links auf einem Qubit (Vektor).

Erinnerung: Die Multiplikation eines Vektors mit einer (quadratischen) Matrix beschreibt eine lineare Abbildung.

In unserem Fall liefert die Multiplikation eines Vektors mit einer unitären Matrix $A \in \mathbb{C}^{n \times n}$ eine unitäre Transformation

$$A : \mathbb{C}^n \rightarrow \mathbb{C}^n, v \mapsto Av.$$

Operationen auf Qubits

Definition (Hadamard-Matrix)

Die Matrix

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

heißt *Hadamard-Matrix*.



Lemma

Die Hadamard-Matrix ist unitär.

Beweis: Übung.

Jacques Hadamard

Lösung nochmal ansehen !

Operationen auf Qubits

Wir untersuchen die Wirkung der Hadamard-Transformation auf der Z-Basis $\{|0\rangle, |1\rangle\}$. Wegen

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

gilt

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle.$$

Operationen auf Qubits

Analog:

$$|1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle.$$

Da $H = H^{-1}$ gilt, erhalten wir nach wiederholter Anwendung

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \xrightarrow{H} |0\rangle$$

und

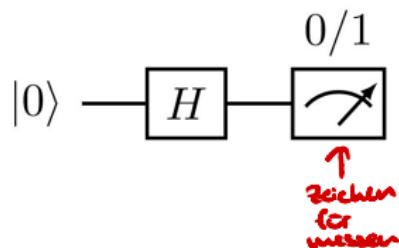
$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \xrightarrow{H} |1\rangle.$$

Zufallszahlengenerator

Erste Anwendung: Ein (theoretisch echter) Zufallszahlengenerator

Algorithmus: Zufallszahlengenerator

1. $|x\rangle \leftarrow |0\rangle$
2. $|x\rangle \leftarrow H|x\rangle$
3. Messe $|x\rangle$



Analyse:

- Schritt 1: Qubit wird in den Anfangszustand $|0\rangle$ versetzt.
- Schritt 2: Anwenden der Hadamard-Transformation. Das Qubit befindet sich dann im Zustand $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.
- Messen des Qubits liefert mit Wahrscheinlichkeit $1/2$ den Zustand $|0\rangle$ und mit Wahrscheinlichkeit $1/2$ den Zustand $|1\rangle$.

Zufallszahlengenerator

Übung: Ersetzen Sie die erste Zeile des Algorithmus durch

- $|x\rangle \leftarrow |1\rangle$, bzw.
- $|x\rangle \leftarrow \alpha|0\rangle + \beta|1\rangle$, mit $|\alpha|^2 + |\beta|^2 = 1$.

Welches Verhalten ergibt sich dann?

$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

$$|\alpha|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 \quad |\beta|^2 = \left|\frac{1}{\sqrt{2}}\right|^2$$

$$\underline{\underline{\alpha^2 = \frac{1}{2}}}$$

$$\underline{\underline{\beta^2 = \frac{1}{2}}}$$

1. Schritt: Qubit in zulässigen Zustand

2. Schritt: $\alpha|0\rangle + \beta|1\rangle$

$$\begin{aligned} & \alpha \cdot \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \beta \cdot \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ &= \frac{\alpha + \beta}{\sqrt{2}}|0\rangle + \frac{\alpha - \beta}{\sqrt{2}}|1\rangle \end{aligned}$$

Zufallszahlengenerator

Übung: Ersetzen Sie die erste Zeile des Algorithmus durch

- $|x\rangle \leftarrow |1\rangle$, bzw.
- $|x\rangle \leftarrow \alpha|0\rangle + \beta|1\rangle$, mit $|\alpha|^2 + |\beta|^2 = 1$.

Welches Verhalten ergibt sich dann?

Bitte beachten Sie

- In der Realität liefert die dafür notwendige Hardware keine gleichmäßige Werteverteilung.
- Die notwendige Nachbearbeitung ist nicht trivial.
- Kein aktuell von diversen Start-Ups vertriebener QRNG ist nach gängigen Kriterien zertifiziert.

Quantenregister

Ein Qubit ist für komplexere Anwendungen nicht ausreichend. Wir benötigen zunächst eine Möglichkeit mehrere Qubits zu beschreiben, um perspektivisch auch auf mehreren Qubits Berechnungen durchführen zu können.

Zur Beschreibung von *Quantenregistern* verwendet man das *Tensorprodukt*³.

³Formale Definition folgt später.

Quantenregister

Beispiel: Das (Tensor-)Produkt von $|0\rangle$ mit sich selbst ist $|0\rangle \otimes |0\rangle = |0\rangle|0\rangle = |00\rangle$ und durch kombinierte Produktbildung von $|0\rangle$ und $|1\rangle$ ergibt sich eine Basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

Ein allgemeiner Zustand bzgl. dieser Basis ist eine Superposition $\alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$ mit $|\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 = 1$.

Messen wir bzgl. dieses Basis, erhalten wir

- $|00\rangle$ mit Wahrscheinlichkeit $|\alpha_0|^2$.
- $|01\rangle$ mit Wahrscheinlichkeit $|\alpha_1|^2$.
- $|10\rangle$ mit Wahrscheinlichkeit $|\alpha_2|^2$.
- $|11\rangle$ mit Wahrscheinlichkeit $|\alpha_3|^2$.

Quantenregister

Allgemeiner: 2-Qubit Register

$R = |x_1\rangle|x_0\rangle$ mit $|x_0\rangle = \gamma_0|0\rangle + \gamma_1|1\rangle$ und $|x_1\rangle = \beta_0|0\rangle + \beta_1|1\rangle$.

Dann ist

$$\begin{aligned} R &= |x_1\rangle|x_0\rangle \\ &= \beta_0\gamma_0|0\rangle|0\rangle + \beta_0\gamma_1|0\rangle|1\rangle + \beta_1\gamma_0|1\rangle|0\rangle + \beta_1\gamma_1|1\rangle|1\rangle. \end{aligned}$$

Kurzschreibweise: $\alpha_{ij} = \beta_i\gamma_j$ und $|i\rangle|j\rangle = |ij\rangle$. Der Bitstring wird durch die in der Binärdarstellung repräsentierte Zahl ersetzt:

$$\begin{aligned} R &= \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \\ &= \alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle + \alpha_3|3\rangle. \end{aligned}$$

Quantenregister

Beobachtung: Die Amplituden $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ ergeben sich als Produkt der Amplituden der ursprünglichen Qubits
 $|x_0\rangle = \gamma_0|0\rangle + \gamma_1|1\rangle$ und $|x_1\rangle = \beta_0|0\rangle + \beta_1|1\rangle$.

Übung: Zeigen Sie: Aus $|\gamma_0|^2 + |\gamma_1|^2 = 1$ und $|\beta_0|^2 + |\beta_1|^2 = 1$ folgt $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

$$\begin{aligned} 1 = 1 \cdot 1 &= (|\beta_0|^2 + |\beta_1|^2)(|\gamma_0|^2 + |\gamma_1|^2) = \underbrace{|\beta_0\gamma_0|^2}_{= |\alpha_{00}|^2} + \underbrace{|\beta_0\gamma_1|^2}_{= |\alpha_{01}|^2} + \underbrace{|\beta_1\gamma_0|^2}_{= |\alpha_{10}|^2} + \underbrace{|\beta_1\gamma_1|^2}_{= |\alpha_{11}|^2} \\ &= |\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 \end{aligned}$$

Quantenregister

Beobachtung: Die Amplituden $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ ergeben sich als Produkt der Amplituden der ursprünglichen Qubits
 $|x_0\rangle = \gamma_0|0\rangle + \gamma_1|1\rangle$ und $|x_1\rangle = \beta_0|0\rangle + \beta_1|1\rangle$.

Übung: Zeigen Sie: Aus $|\gamma_0|^2 + |\gamma_1|^2 = 1$ und $|\beta_0|^2 + |\beta_1|^2 = 1$ folgt $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

Übung: Betrachten Sie das 2-Qubit Register $R = |x_1\rangle|x_0\rangle$ mit $|x_0\rangle = \frac{1}{2}|0\rangle - \frac{\sqrt{3}}{2}|1\rangle$ und $|x_1\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$. Bestimmen Sie die Amplituden $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ und führen Sie eine Messung bzgl. der (kombinierten) Basis durch.

Übung: Betrachten Sie das 2-Qubit Register $R = |x_1\rangle|x_0\rangle$ mit $|x_0\rangle = \frac{1}{2}|0\rangle - \frac{\sqrt{3}}{2}|1\rangle$ und $|x_1\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$. Bestimmen Sie die Amplituden $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ und führen Sie eine Messung bzgl. der (kombinierten) Basis durch.

$$\begin{aligned} R &= \frac{1}{2} \cdot \frac{1}{2} |0\rangle|0\rangle - \frac{1}{2} \cdot \frac{\sqrt{3}}{2} |0\rangle|1\rangle + \frac{\sqrt{3}}{2} \cdot \frac{1}{2} |1\rangle|0\rangle - \frac{\sqrt{3}}{2} \cdot \frac{\sqrt{3}}{2} |1\rangle|1\rangle \\ &= \frac{1}{4} |00\rangle - \frac{\sqrt{3}}{4} |01\rangle + \frac{\sqrt{3}}{4} |10\rangle - \frac{3}{4} |11\rangle \\ &= \frac{1}{4} |0\rangle - \frac{\sqrt{3}}{4} |1\rangle + \frac{\sqrt{3}}{4} |2\rangle - \frac{3}{4} |3\rangle \end{aligned}$$

Also:

$$\alpha_0 = \frac{1}{4}, \quad \alpha_1 = \frac{\sqrt{3}}{4}, \quad \alpha_2 = -\frac{\sqrt{3}}{4}, \quad \alpha_3 = \frac{3}{4}$$

und eine Messung führt

- mit Wk. $|\frac{1}{4}|^2 = \frac{1}{16}$ auf $|00\rangle$
- mit Wk. $|\frac{\sqrt{3}}{4}|^2 = \frac{3}{16}$ auf $|01\rangle$
- mit Wk. $|\frac{-\sqrt{3}}{4}|^2 = \frac{3}{16}$ auf $|10\rangle$
- mit Wk. $|\frac{3}{4}|^2 = \frac{9}{16}$ auf $|11\rangle$

Quantenregister

Nun für allgemeine Quantenregister:

Definition (Quantenregister)

Ein Quantenregister R der Länge $n \geq 1$ hat die Form

$R = |x_{n-1}\rangle|x_{n-2}\rangle\dots|x_0\rangle = |x_{n-1}x_{n-2}\dots x_0\rangle$. Es kann sich in einem Zustand der Form

$$\sum_{i=0}^{2^n-1} \alpha_i |i\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \dots + \alpha_{2^n-1} |2^n - 1\rangle$$

befinden, wobei $|i\rangle = |\text{bin}(i)\rangle$ und $\alpha_i \in \mathbb{C}$ für $i = 0, \dots, 2^n - 1$ gelte.
Es gilt $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ und beim Messen des Quantenregisters beobachtet man den Zustand $|i\rangle$ mit Wahrscheinlichkeit $|\alpha_i|^2$.

Quantenregister

Der Inhalt eines (logischen) n -Bit Registers ist ein n -Bit String, d.h. es sind 2^n Inhalte möglich. Ein n -Qubit Register befindet sich in Superposition all dieser Zustände.

Für $n = 300$ Qubits müssten $2^{300} \approx 2,04 \cdot 10^{90}$ Amplituden berücksichtigt werden. Die Anzahl der Atome im sichtbaren Universum wird auf 10^{78} bis 10^{82} geschätzt. Das ist ein Hinweis darauf, dass es schwierig ist für einen klassischen Computer einen Quantencomputer zu simulieren.

Quantenregister

Erinnerung: Ein Qubit ist ein Vektor im zweidimensionalen Raum.

Die Zustände eines Quantenregisters mit n Qubits entsprechen Vektoren in einem 2^n -dimensionalen komplexen Vektorraum. Die Basis bilden die einzelnen Komponenten der Superposition, also

Basiszustände → $|0\dots00\rangle, |0\dots01\rangle, \dots, |1\dots11\rangle$.

Beispiel: Für ein 2-Qubit Register ist $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ eine Basis mit der entsprechenden Zuordnung

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Quantenregister

Da die Zustandsvektoren eines n -Bit Quantenregisters je 2^n Komponenten besitzen, entsprechen die einzelnen Rechenschritte eines Quantencomputers unitären Transformationen, die durch unitäre $2^n \times 2^n$ -Matrizen darstellbar sind.

Im Fall von Ein-Qubit Systemen haben wir spezielle Matrizen bzw. Transformationen kennengelernt, die auf diesen operieren.

Es gibt auch Quantengatter die auf zwei Qubits gleichzeitig operieren. Einige dieser Gatter wollen wir näher betrachten (auch als Vorbereitung für weitere Quantenalgorithmen).

Quantenregister

Das **CNOT-Gatter** bzw. ***controlled-NOT* Gatter** invertiert das rechte Qubits, wenn das linke (Kontroll-)Qubit auf 1 gesetzt ist, d.h.

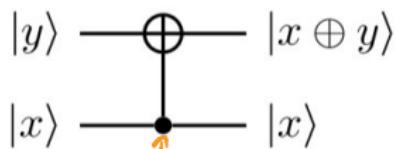
Controllbit (wenn Controllbit auf 1,
rechtes bit flippen)

$$\text{CNOT } |00\rangle = |00\rangle$$

$$\text{CNOT } |01\rangle = |01\rangle$$

$$\text{CNOT } |10\rangle = |11\rangle$$

$$\text{CNOT } |11\rangle = |10\rangle$$



wenn ausgewählt rechtes
flippen, wenn nicht ausgewählt
linkes flippen

$$A_{\text{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

oder kompakt

$$\text{CNOT} : |x, y\rangle \mapsto |x, x \oplus y\rangle.$$

Quantenregister

Notation: In Schaltungen wird absteigend sortiert, d.h. das am weitesten rechts geschriebene Qubit im Register steht in der Schaltung am weitesten oben und das am weitesten links stehende Qubit steht in der Schaltung unten.

Übung:

1. Leiten Sie die Matrixdarstellung A_{CNOT} von CNOT her.
2. Zeigen Sie, dass A_{CNOT} unitär ist.

Erinnerung: Die Multiplikation eines Vektors mit einer (unitären) Matrix $A \in \mathbb{C}^{n \times n}$ liefert eine (unitäre) Transformation $A : \mathbb{C}^n \rightarrow \mathbb{C}^n, v \mapsto Av$. In den Spalten von A stehen die Bilder der Standard-Basisvektoren bei der Abbildung $v \mapsto Av$.

Übung:

- Leiten Sie die Matrixdarstellung A_{CNOT} von CNOT her.
- Zeigen Sie, dass A_{CNOT} unitär ist.

$$A_{\text{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$1) \quad \text{CNOT } |00\rangle = |00\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\text{CNOT } |01\rangle = |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \left. \right\} \text{also: } A_{\text{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\text{CNOT } |10\rangle = |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\text{CNOT } |11\rangle = |10\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$2) \quad A_{\text{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}; \quad A_{\text{CNOT}}^{\dagger} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad // A_{\text{CNOT}} = A_{\text{CNOT}}^{\dagger}$$

$$A_{\text{CNOT}} \cdot A_{\text{CNOT}}^{\dagger} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = I_4$$

\rightarrow unitär

Quantenregister

Es gilt noch mehr: A_{CNOT} ist eine Permutationsmatrix.

Erinnerung: Sei $\pi \in S_n$ (S_n die symmetrische Gruppe; deren Elemente nennt man Permutationen). Dann ist die dazugehörige $(n \times n)$ -Permutationsmatrix $P_\pi = (p_{ij})$ definiert durch $p_{ij} = \delta_{\pi(i), j}$.
Per Definition ist P_π also eine quadratische Matrix, die in jeder Zeile und jeder Spalte genau einen Eintrag 1 und sonst nur Nullen enthält

Übung: Zeigen Sie: Permutationsmatrizen sind unitär.

Konkret
Funktion

$$\delta = \begin{cases} 1, & a=b \\ 0, & \text{sonst} \end{cases}$$

Übung: Zeigen Sie: Permutationsmatrizen sind unitär.

Erinnerung: Sei $\pi \in \text{Sn}$. Die zugehörige $(n \times n)$ -Permutationsmatrix $P_\pi = (p_{ij})$ durch:

$$p_{ij} = \delta_{\pi(i), j} = \begin{cases} 1 & \text{falls } \pi(i) = j, \\ 0 & \text{sonst} \end{cases}$$

Per Definition sind die Einträge von P_π Elemente aus $\{0, 1\} \subset \mathbb{Z}$, also gilt $\overline{P_\pi} = P_\pi$.

Transponieren liefert nun $P_\pi^T = (p_{ij}) = (\delta_{\pi(i), j})$ mit $\delta_{\pi(i), j} = \begin{cases} 1 & \text{falls } \pi(j) = i, \\ 0 & \text{sonst} \end{cases}$

$$= P_{\pi^{-1}} = P_\pi^{-1}$$

Wobei π^{-1} die zu π inverse Permutation bezgl. Komposition bezeichnet.
Mit $P_\pi^T = P_{\pi^{-1}}$ folgt $P_\pi^T \cdot (P_\pi)^T = P_\pi^T = P_\pi^{-1}$

→ Permutationsmatrizen sind also unitär.

Quantenregister

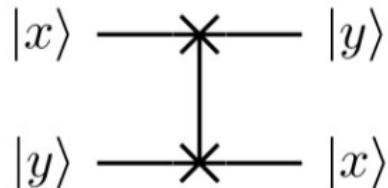
Das **SWAP-Gatter** vertauscht zwei Qubits, d.h.

$$\text{SWAP } |00\rangle = |00\rangle$$

$$\text{SWAP } |01\rangle = |10\rangle$$

$$\text{SWAP } |10\rangle = |01\rangle$$

$$\text{SWAP } |11\rangle = |11\rangle$$



oder kompakt

$$\text{SWAP} : |x\rangle|y\rangle \mapsto |y\rangle|x\rangle.$$

$$A_{\text{SWAP}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Quantenregister

Soll ein bel. 2-Qubit Gatter U mit einer Kontrolle versehen werden, wird daraus das **CU -Gatter** oder auch *controlled-U* Gatter. Dabei wird U auf das rechte Qubit angewendet, wenn das Kontrollqubit auf 1 gesetzt ist:

$$CU |00\rangle = |00\rangle$$

$$CU |01\rangle = |01\rangle$$

$$CU |10\rangle = |1\rangle \otimes U|0\rangle$$

$$CU |11\rangle = |1\rangle \otimes U|1\rangle.$$

Quantenregister

Für

$$U|0\rangle = a|0\rangle + b|1\rangle$$

$$U|1\rangle = c|0\rangle + d|1\rangle$$

folgt

$$A_{CU} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{pmatrix}$$

Quantenregister

Unter diesem Aspekt gilt

$$CX = CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

und natürlich sind weitere Konstruktionen spezieller 2-Qubit Gatter denkbar.

Mit 2-Qubit Gattern können wir den (historisch) ältesten Quantenalgorithmus untersuchen.

Algorithmus von Deutsch

Problem von Deutsch (nach David Deutsch, geb. 1953)

Wir wollen die *Parität* $b_0 \oplus b_1$ zweier unbekannter Bits b_0 und b_1 bestimmen. Äquivalent dazu ist die Frage, ob die Anzahl der Einsen *gerade* oder *ungerade* ist.

Klassische Analogie: Eine echte Münze (Kopf und Zahl) soll von einer gefälschten Münze (beide Seiten Kopf) unterschieden werden. Die Münze muss zweimal betrachtet werden, je einmal von jeder Seite.

Bietet uns ein Quantencomputer in einer solchen (oder ähnlichen) Situation Vorteile?

Algorithmus von Deutsch

Abstrakt: Gegeben eine Funktion $f : \{0, 1\} \rightarrow \{0, 1\}$ und es gibt ein *Orakel* das uns zu einem Bit $b \in \{0, 1\}$ den Wert $f(b)$ liefert. Das Orakel sagt immer die Wahrheit. Wir können es als eine *Subroutine* auffassen.

Die Funktion f heißt *konstant*, wenn $f(0) = f(1)$ gilt. Im Fall $f(0) \neq f(1)$ heißt f *balanciert*.

Frage: Ist f konstant oder balanciert?

Klassisch: Zwei Anfragen an das Orakel, nämlich $f(0)$ und $f(1)$.

Idee QC: Versetze ein Qubit in eine Superposition über die möglichen Eingaben 0 und 1 von f .

Algorithmus von Deutsch

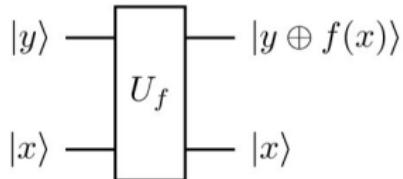
Achtung: f ist möglicherweise nicht umkehrbar (f konstant).
Rechenschritte auf Quantencomputern müssen aber umkehrbar sein.

Wir verwenden

$$U_f : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle.$$

Das extra Qubit $|y\rangle$ nennt man auch *Antwortqubit* oder *target Qubit*;
 $|x\rangle$ nennt man *input Qubit*.
Das Quantenorakel können wir als eine Quanten-Subroutine verstehen.

Übung: Zeige, das U_f unitär ist.



Übung: Zeige, das U_f unitär ist.

1. Möglichkeit: f ist die Nullfunktion, d.h. $f: \{0,1\} \rightarrow \{0,1\}$ mit $f(0)=0, f(1)=0$

Dann gilt: $U_f: \begin{cases} |00\rangle \mapsto |0,0 \oplus f(0)\rangle = |00\rangle \\ |01\rangle \mapsto |0,1 \oplus f(0)\rangle = |01\rangle \\ |10\rangle \mapsto |1,0 \oplus f(1)\rangle = |10\rangle \\ |11\rangle \mapsto |1,1 \oplus f(1)\rangle = |11\rangle \end{cases}$

\Rightarrow Das heißt U_f ist die Identität und wird durch I_4 beschrieben. $U_f = I_4$ ist unitär.

2. Möglichkeit: f ist die Identität, d.h. $f: \{0,1\} \rightarrow \{0,1\}$ mit $f(0)=0, f(1)=1$

Dann gilt: $U_f: \begin{cases} |00\rangle \mapsto |0,0 \oplus f(0)\rangle = |00\rangle \\ |01\rangle \mapsto |0,1 \oplus f(0)\rangle = |01\rangle \\ |10\rangle \mapsto |1,0 \oplus f(1)\rangle = |11\rangle \\ |11\rangle \mapsto |1,1 \oplus f(1)\rangle = |10\rangle \end{cases}$

\Rightarrow Die Abbildung kennen wir bereits unter CNOT, d.h. $U_f \cdot \text{CNOT} \Rightarrow U_f$ ist unitär.

3. Möglichkeit: f ist die Negation, d.h. $f: \{0,1\} \rightarrow \{0,1\}$, mit $f(0)=1, f(1)=0$

Dann gilt: $U_f: \begin{cases} |00\rangle \mapsto |0,0 \oplus f(0)\rangle = |01\rangle \\ |01\rangle \mapsto |0,1 \oplus f(0)\rangle = |10\rangle \\ |10\rangle \mapsto |1,0 \oplus f(1)\rangle = |10\rangle \\ |11\rangle \mapsto |1,1 \oplus f(1)\rangle = |11\rangle \end{cases}$

Wir wissen: U_f kann durch eine Matrix beschrieben werden.

Die mit $|i\rangle$ beschriftete Spalte beschreibt das Bild des Basisvektors $|i\rangle$

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

$\Rightarrow U_f$ ist eine Permutationsmatrix, also unitär.

4. Möglichkeit: f ist die Einheitsfunktion, also $f: \{0,1\} \rightarrow \{0,1\}, f(0)=1, f(1)=1$

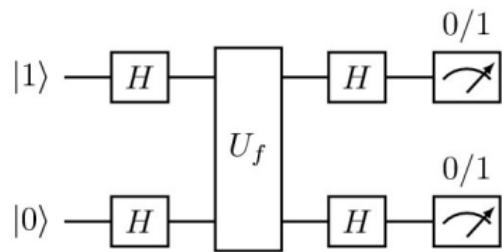
Dann gilt: $\begin{cases} |00\rangle \mapsto |0,0 \oplus f(0)\rangle = |01\rangle \\ |01\rangle \mapsto |0,1 \oplus f(0)\rangle = |00\rangle \\ |10\rangle \mapsto |1,0 \oplus f(1)\rangle = |11\rangle \\ |11\rangle \mapsto |1,1 \oplus f(1)\rangle = |10\rangle \end{cases}$ $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

$\Rightarrow U_f$ ist eine Permutationsmatrix und als solche unitär.

Algorithmus von Deutsch

Algorithmus von Deutsch

1. $|x\rangle|y\rangle \leftarrow |0\rangle|1\rangle$
2. $|x\rangle|y\rangle \leftarrow H|x\rangle H|y\rangle$
3. $|x\rangle|y\rangle \leftarrow U_f |x\rangle|y\rangle$
4. $|x\rangle|y\rangle \leftarrow H|x\rangle H|y\rangle$
5. Messe das Register $|x\rangle|y\rangle$:
 - $|0\rangle|1\rangle$: f ist konstant
 - $|1\rangle|1\rangle$: f ist balanciert



Algorithmus von Deutsch

Analyse des Algorithmus: In Schritt 2 wird $|x\rangle|y\rangle$ durch Hadamard-Transformation auf $|0\rangle|1\rangle$ in

$$\begin{aligned} |\phi_2\rangle &= H|0\rangle H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \cdot \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \frac{1}{2}(|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle) \end{aligned}$$

überführt. Das ist eine Superposition über alle Basiszustände des Registers.

Algorithmus von Deutsch

In Schritt 3 wird $U_f : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle$ angewandt:

$$\begin{aligned} |\phi_3\rangle &= U_f |\phi_2\rangle = \frac{1}{2} (U_f |0\rangle|0\rangle - U_f |0\rangle|1\rangle + U_f |1\rangle|0\rangle - U_f |1\rangle|1\rangle) \\ &= \frac{1}{2} (|0\rangle|f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle) \\ &= \frac{1}{2} (|0\rangle(|f(0)\rangle - |1 \oplus f(0)\rangle) + |1\rangle(|f(1)\rangle - |1 \oplus f(1)\rangle)). \end{aligned}$$

Wir können die Terme weiter vereinfachen...

Übung: Für $x \in \{0, 1\}$ ist $|f(x)\rangle - |1 \oplus f(x)\rangle = (-1)^{f(x)}(|0\rangle - |1\rangle)$.

$$|f(x)\rangle - |1 \oplus f(x)\rangle = \begin{cases} |0\rangle - |1 \oplus 0\rangle, & \text{for } f(x)=0 \\ |1\rangle - |1 \oplus 1\rangle, & \text{for } f(x)=1 \end{cases}$$

$$=(-1)^{f(x)}(|0\rangle - |1\rangle)$$

Algorithmus von Deutsch

... und erhalten

$$\begin{aligned} |\phi_3\rangle &= \frac{1}{2} \left((-1)^{f(0)} |0\rangle (|0\rangle - |1\rangle) + (-1)^{f(1)} |1\rangle (|0\rangle - |1\rangle) \right) \\ &= \frac{1}{2} \left((-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \right) (|0\rangle - |1\rangle). \end{aligned}$$

Der Funktionswert wurde in das Vorzeichen der Amplituden des ersten Bits $|x\rangle$ von $|\phi_3\rangle$ verlagert, wobei

$$|x\rangle = \frac{1}{\sqrt{2}} \left((-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \right).$$

Algorithmus von Deutsch

In **Schritt 4** erfolgt die Fallunterscheidung für die Funktion f :

1. Möglichkeit: f ist konstant, also $f(0) = f(1)$. Dann ist $(-1)^{f(0)} = (-1)^{f(1)}$ und entweder

$$|x\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \xrightarrow{H} |0\rangle$$

für $f(0) = f(1) = 0$ oder

$$|x\rangle = -\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \xrightarrow{H} -|0\rangle$$

für $f(0) = f(1) = 1$.

Algorithmus von Deutsch

Für das zweite Qubit in $|\phi_3\rangle$ gilt

$$\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \xrightarrow{H} |1\rangle.$$

Damit enthält das Register $\pm|0\rangle|1\rangle$.

2. Möglichkeit: f ist balanciert, also $f(0) \neq f(1)$. Dann ist

$$|x\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \xrightarrow{H} |1\rangle$$

für $f(0) = 0, f(1) = 1$;

Algorithmus von Deutsch

oder

$$|x\rangle = -\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \xrightarrow{H} -|1\rangle$$

für $f(0) = 1, f(1) = 0$. Das Register enthält dann $\pm|1\rangle|1\rangle$.

Damit wird im Fall f konstant $|0\rangle|1\rangle$ und im Fall f balanciert $|1\rangle|1\rangle$ gemessen.

Übung: Vollziehen Sie die Überlegungen für die Fälle

- $f(0) = 1, f(1) = 0$, d.h. f ist die Negation, und
- $f(0) = f(1) = 1$, d.h. f ist die 1-Funktion

nach.

Quantenregister

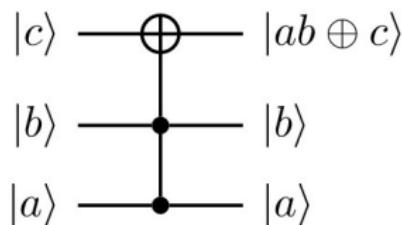
Für den Algorithmus von Deutsch war unser bisheriges Wissen über Ein-Qubit und 2-Qubit Systeme, sowie Operationen auf diesen, gerade ausreichend. Erwartungsgemäß kommt man auch mit 2-Qubit Systemen noch nicht sehr weit.

Tatsächlich gibt es auch spezielle Gatter die gleichzeitig auf drei Qubits arbeiten.

Das Toffoli Gatter

Toffoli $|a\rangle|b\rangle|c\rangle \mapsto |a\rangle|b\rangle|ab \oplus c\rangle$

ist ein solches Gatter.



Quantenregister

Das Toffoli Gatter ist

- reversibel
- und sogar *universell* für klassische Computer,

d.h. jeder effiziente klassische Algorithmus kann in einen effizienten Algorithmus der nur Toffoli Gatter verwendet, konvertiert werden.

Da das Toffoli Gatter auch ein Quantengatter ist, können Quantencomputer effizient berechnen, was auch klassische Computer effizient berechnen können. Anders:

$$P \subset BQP.$$

Tensorprodukt

Auf Dauer ist unsere bisherige Arbeitsweise (Steigerung in der Anzahl der Qubits führt zu (teils) grundlegend neuen Gattern) kein praktikabler Weg.

Wir benötigen eine *konstruktive* Methode, auch um bereits bekannte Gatter in einem erweiterten Setting verwenden zu können.

Tensorprodukt

Auf Dauer ist unsere bisherige Arbeitsweise (Steigerung in der Anzahl der Qubits führt zu (teils) grundlegend neuen Gattern) kein praktikabler Weg.

Wir benötigen eine *konstruktive* Methode, auch um bereits bekannte Gatter in einem erweiterten Setting verwenden zu können.

Vorgehen: Wir überlegen uns wie wir formal Register aus einzelnen Qubits zusammensetzen. Damit wollen wir Operationen auf einem Register durch Operationen auf einzelnen Qubits zusammenfügen.

Erinnerung: Qubits sind Linearkombinationen von Basisvektoren.

Tensorprodukt

Definition (Tensorprodukt von Vektorräumen - Teil 1)

Sei V_1 ein \mathbb{C} -Vektorraum mit Basis $\{e_0, \dots, e_{m-1}\}$ und V_2 ein \mathbb{C} -Vektorraum mit Basis $\{f_0, \dots, f_{n-1}\}$. Das *Tensorprodukt* $V_1 \otimes V_2$ dieser Räume ist ein mn -dimensionaler Vektorraum, dessen Basisvektoren mit

$$\begin{array}{cccc} e_0 \otimes f_0 & e_0 \otimes f_1 & \dots & e_0 \otimes f_{n-1} \\ e_1 \otimes f_0 & e_1 \otimes f_1 & \dots & e_1 \otimes f_{n-1} \\ \vdots & \vdots & & \vdots \\ e_{m-1} \otimes f_0 & e_{m-1} \otimes f_1 & \dots & e_{m-1} \otimes f_{n-1} \end{array}$$

bezeichnet werden.

Tensorprodukt

Definition (Tensorprodukt von Vektorräumen - Teil 2)

Ist

$$v_1 = \alpha_0 e_0 + \dots + \alpha_{m-1} e_{m-1} \in V_1$$

und

$$v_2 = \beta_0 f_0 + \dots + \beta_{n-1} f_{n-1} \in V_2,$$

dann ist ihr Tensorprodukt

$$v_1 \otimes v_2 = \left(\sum_{i=0}^{m-1} \alpha_i e_i \right) \otimes \left(\sum_{j=0}^{n-1} \beta_j f_j \right) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \alpha_i \beta_j (e_i \otimes f_j).$$

Tensorprodukt

Beispiel: Für $m = n = 2$ ist

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \otimes \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}.$$

Also ist

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

und damit (wie bisher) $|1\rangle \otimes |0\rangle = |10\rangle$. Analog zeigt man $|0\rangle \otimes |0\rangle = |00\rangle$, $|0\rangle \otimes |1\rangle = |01\rangle$ und $|1\rangle \otimes |1\rangle = |11\rangle$.

Tensorprodukt

Allgemeiner liefert die Definition des Tensorproduktes für $|\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ und $|\psi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$, dass

$$|\phi\rangle \otimes |\psi\rangle = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle.$$

Das ergibt sich (wie bisher) auch durch „Ausmultiplizieren“ von

$$(\alpha_0|0\rangle + \alpha_1|1\rangle) \cdot (\beta_0|0\rangle + \beta_1|1\rangle).$$

Übung: Berechnen Sie

$$\begin{aligned} & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \\ &= \frac{1}{\sqrt{2}}(10\rangle + 11\rangle) \cdot \frac{1}{\sqrt{2}}(10\rangle - 11\rangle) \\ &= \frac{1}{2}(100\rangle - 101\rangle + 110\rangle - 111\rangle) \end{aligned}$$

Tensorprodukt

Lemma (Produkt von Zuständen)

Die Beschreibung eines Registers aus m Bits lässt sich aus dem m -fachen Tensorprodukt der Beschreibung eines Bits erzeugen.

Sind die Bits $|x_1\rangle, \dots, |x_m\rangle$ in den Zuständen $|\phi_1\rangle, \dots, |\phi_m\rangle$, so befindet sich das Register $|x_1 \dots x_m\rangle$ im Zustand $|\phi_1\rangle \otimes \dots \otimes |\phi_m\rangle$.

Die Amplituden können (wie bisher) durch Ausmultiplizieren berechnet werden (weshalb wir häufig \cdot statt \otimes schreiben werden, auch wenn es unpräzise ist).

Tensorprodukt

Wenn (einige) Quantenzustände als Tensorprodukt entstehen, liegt es nahe auch nach der *Faktorisierung* von Zuständen zu fragen. So ist

$$\frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \underbrace{\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)}_{|+\rangle} \otimes \underbrace{\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)}_{|-\rangle}$$
$$= |+\rangle \otimes |-\rangle$$
$$= |+\rangle |-\rangle.$$

Solche faktorisierbaren Zustände nennt man *Produktzustände* oder *separabel*.

Tensorprodukt

Andererseits gibt es auch Zustände die sich nicht in ein Tensorprodukt zerlegen lassen, etwa

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle).$$

Solche Zustände nennen wir *verschränkt*. Wir später diese später weiter untersuchen und lernen wie man dies nutzen kann.

Momentan sind wir an einer konstruktiven Methode zur Generierung von Gattern interessiert.

Tensorprodukt

Definition (Tensorprodukt von Matrizen)

Seien A und B Matrizen mit komplexen Einträgen, wobei

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m-1} & \dots & a_{mn} \end{pmatrix}.$$

Das *Tensorprodukt* $A \otimes B$ von A und B ist

$$A \otimes B = \begin{pmatrix} a_{11} \cdot B & \dots & a_{1n} \cdot B \\ \vdots & & \vdots \\ a_{m-1} \cdot B & \dots & a_{mn} \cdot B \end{pmatrix}.$$

Tensorprodukt

Beispiel:

$$I_2 \otimes I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes I_2 = \begin{pmatrix} I_2 & 0 \\ 0 & I_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = I_4$$

Als Verallgemeinerung des Hadamard Gatters ergibt sich

Definition (H_n)

Die 2^n -dimensionale Hadamard-Transformation H_n ist definiert durch

$$H_n = \bigotimes_{i=1}^n H.$$

Tensorprodukt

Übung: Berechnen Sie H_2 .

Wir beobachten, dass die folgenden Aktionen auf dasselbe Resultat führen

- Anwendung der durch die Matrizen A_1, \dots, A_m beschriebenen Transformationen auf die Bits $|x_1\rangle, \dots, |x_m\rangle$. Dabei wird jeweils A_i auf $|x_i\rangle$ angewandt.
- Anwendung der Transformation $A_1 \otimes \dots \otimes A_m$ auf das Register $|x_1 \dots x_m\rangle$.

Mit dem Tensorprodukt haben wir also die Möglichkeit, Operationen auf Registern zu konstruieren / auszuführen.

Berechnen $S \otimes H_2$

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

$$H_2 = H \otimes H$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{bzw. } \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} = \dots$$

$$= \left(\boxed{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}} \right) \otimes \left(\boxed{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}} \right)$$

$$\left(\begin{array}{cccc} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{array} \right) = \frac{1}{2} \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}}$$

Tensorprodukt

Beispiel: Betrachte $R = |x\rangle|y\rangle$

- $H \otimes H = H_2$ beschreibt auf Registerebene die Anwendung von H auf $|x\rangle$ und auf $|y\rangle$
- $H \otimes I_2$ beschreibt die Anwendung von H auf $|x\rangle$ und lässt $|y\rangle$ unverändert

Übung: Berechnen Sie $H \otimes I_2$ und $I_2 \otimes H$. Ist das Tensorprodukt kommutativ?

Universelle Gatter

Für logische Gatter ist NAND universell, da mit diesem jedes andere logische Gatter konstruiert werden kann.

Eine Menge von Quantengattern ist eine **universelle Gattermenge**, wenn damit jedes Quantengatter mit beliebiger Genauigkeit approximiert werden kann.

Einige Besonderheiten sind das mit universellen Quantengattern die Möglichkeit gegeben sein muss

- Superposition zu erzeugen,
- Verschränkung herzustellen,
- Zustände mit komplexe Amplituden zu generieren,
- mehr als die **Clifford Gatter** {CNOT, H, S} enthalten sein muss.

Universelle Gatter

Unter den Clifford Gattern erzeugt H Superposition, CNOT Verschränkung und S liefert komplexwertige Amplituden. Nachweislich bilden diese Gatter aber keine universelle Gattermenge.

Nach dem **Satz von Gottesmann-Knill** können Quantenschaltkreise die nur aus den Clifford Gattern bestehen, von einem klassischen Computer effizient simuliert werden. Solche Schaltungen sind demnach nicht mächtiger als klassische Schaltkreise.

Universelle Gatter

Beispiele universeller Mengen für Quantencomputer sind

- $\{H, \text{alle Ein-Qubit Gatter}\}$
- $\{\text{CNOT}, H, T\}$, d.h. Austausch von S durch T liefert eine universelle Menge
- $\{\text{CNOT}, R_{\frac{\pi}{8}}, S\}$, d.h. Austausch von H durch $R_{\frac{\pi}{8}}$ liefert eine universelle Menge für

$$R_{\frac{\pi}{8}} = \begin{pmatrix} \cos\left(\frac{\pi}{8}\right) & -\sin\left(\frac{\pi}{8}\right) \\ \sin\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{8}\right) \end{pmatrix}$$

- $\{\text{Toffoli}, H, S\}$

Universelle Gatter

Der **Satz von Solovay-Kitaev** besagt, dass ein Quantengatter auf n Qubit durch eine universelle Menge von Quantengattern zu einer Präzision ε durch $\Theta(2^n \log^c(\frac{1}{\varepsilon}))$ Gatter approximiert werden kann; c eine Konstante.

Die Abhängigkeit von 2^n ist zu erwarten, da für die Operation auf n Qubits eine $2^n \times 2^n$ Matrix benötigt wird. Die Abhängigkeit von der Approximationsgüte ε in $\log^c(\frac{1}{\varepsilon})$ ist gut, da diese erwartbar klein sein soll und damit $\log^c(\frac{1}{\varepsilon})$ von der Größenordnung *polylog* ist. Die Approximation durch die universellen Gatter konvergiert damit schnell gegen das zu approximierende Gatter.

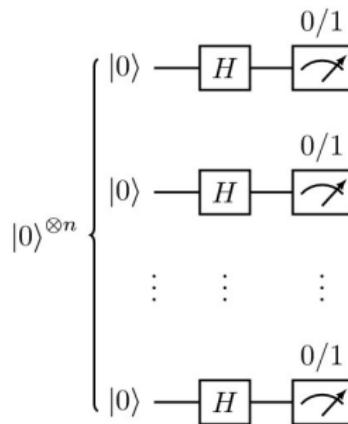
Erweiterter Zufallszahlengenerator

Mit den neuen Möglichkeiten können wir den Ansatz zur Erzeugung von Zufallszahlen erweitern:

Algorithmus: *n*-Bit Zufallsgenerator

Ausgabe: Zufallszahl zwischen 0 und $2^n - 1$

1. $R = |x_{n-1} \dots x_0\rangle \leftarrow |0 \dots 0\rangle$
2. $R \leftarrow H_n R$
3. Messe R



Erweiterter Zufallszahlengenerator

Analyse des Algorithmus: In **Schritt 2** wird H_n auf das Register angewandt; d.h. H wirkt auf jedes einzelne Bit, sodass

$$|0\rangle \dots |0\rangle \xrightarrow{H_n} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \dots \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle).$$

Für das Produkt zweier Faktoren beobachten wir

$$\begin{aligned} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \cdot \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) &= \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ &= \frac{1}{2} (|0\rangle + |1\rangle + |2\rangle + |3\rangle). \end{aligned}$$

Erweiterter Zufallszahlengenerator

Insgesamt liefert die Produktbildung

$$\frac{1}{\sqrt{2^n}} (|0\dots00\rangle + |0\dots01\rangle + \dots + |1\dots1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle.$$

Bei der Messung in **Schritt 3** wird jeder Basiszustand des Registers mit Wahrscheinlichkeit $(1/\sqrt{2^n})^2 = 1/2^n$ angenommen. Jeder dieser Basiszustände repräsentiert eine der Zahlen 0 bis $2^n - 1$.

Messen von Registern

Um weitere Algorithmen betrachten zu können, müssen wir auch unser bisheriges Verständnis einer Messung verfeinern.

Unser bisheriges Verständnis der Messung bzgl. einer Basis hat sich auf die gesamte Basis bezogen.

Erinnerung: Sei R ein Register aus n Quantenbits, das sich im Zustand $|\phi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ befindet. Die orthogonalen Vektoren $|0'\rangle, |1'\rangle, \dots, |(2^n - 1)'\rangle$ der Länge 1 seien die Messbasis von $|\phi\rangle$, d.h.

$$|\phi\rangle = \sum_{i=0}^{2^n-1} \alpha'_i |i'\rangle.$$

Dann befindet sich das Register nach Messung mit Wahrscheinlichkeit $|\alpha'_i|^2$ im Zustand $|i'\rangle$.

Messen von Registern

Es können auch einzelne Bits eines Registers gemessen werden:

Für $R = |xy\rangle$ im Zustand

$$|\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

kann bspw. das erste Bit gemessen werden. Das Ergebnis ist $|0\rangle$ oder $|1\rangle$.

1. Fall: Messen nach $|x\rangle = |0\rangle$

Das Register geht in eine Superposition von $|00\rangle$ und $|01\rangle$, genauer,

$$|\phi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

über. Die Wahrscheinlichkeit dafür beträgt $|\alpha_{00}|^2 + |\alpha_{01}|^2$.

Messen von Registern

2. Fall: Messen nach $|x\rangle = |1\rangle$

Das Register geht in eine Superposition von $|10\rangle$ und $|11\rangle$, genauer, den Zustand

$$|\phi'\rangle = \frac{\alpha_{10}|10\rangle + \alpha_{11}|11\rangle}{\sqrt{|\alpha_{10}|^2 + |\alpha_{11}|^2}}$$

über. Die Wahrscheinlichkeit dafür beträgt $|\alpha_{10}|^2 + |\alpha_{11}|^2$.

Um einen zulässigen Quantenzustand zu erhalten, musste *normiert* werden.

Hier ist durch Messung ein Übergang von einer Superposition in eine andere Superposition (Folgezustand) entstanden. Im Vergleich dazu hat eine Messung bisher zu einer Auflösung der Superposition geführt.

Messen von Registern

Allgemein gilt:

Ist R ein Register aus n Quantenbits im Zustand $|\phi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ und für $j \in \{1, \dots, n\}$ sei

$$I_{0,j} = \{i \in \{0, \dots, 2^n - 1\} : j\text{-tes Bit von links in } \text{bin}(i) \text{ von } i \text{ ist } |0\rangle\},$$
$$I_{1,j} = \{i \in \{0, \dots, 2^n - 1\} : j\text{-tes Bit von links in } \text{bin}(i) \text{ von } i \text{ ist } |1\rangle\}.$$

Wird das j -te Bit des Registers gemessen, so nimmt es mit

Wahrscheinlichkeit $\sum_{i \in I_{j,0}} |\alpha_i|^2$ den Wert $|0\rangle$ an. Das Register ist dann im Zustand

$$\frac{\sum_{i \in I_{j,0}} \alpha_i |i\rangle}{\sqrt{\sum_{i \in I_{j,0}} |\alpha_i|^2}}.$$

(Beachte: Alle $|i\rangle$ die hier auftreten, haben an Position j eine $|0\rangle$.)

Messen von Registern

Wird das j-te Bit des Registers gemessen, so nimmt es mit Wahrscheinlichkeit $\sum_{i \in I_{j,1}} |\alpha_i|^2$ den Wert $|1\rangle$ an. Das Register ist dann im Zustand

$$\frac{\sum_{i \in I_{j,1}} \alpha_i |i\rangle}{\sqrt{\sum_{i \in I_{j,1}} |\alpha_i|^2}}.$$

(Beachte: Alle $|i\rangle$ die hier auftreten, haben an Position j eine $|1\rangle$.)

Übung: Das 3-Qubit Register R sei im Zustand

$$|\phi\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{2}|101\rangle + \frac{1}{2}|111\rangle.$$

Bestimmen Sie für das zweite Qubit die Wahrscheinlichkeiten mit denen $|0\rangle$ bzw. $|1\rangle$ angenommen wird, sowie die Folgezustände.

An welcher Stelle unterscheiden

$$|\phi\rangle = \frac{1}{\sqrt{2}} |1000\rangle + \frac{1}{\sqrt{2}} |101\rangle + \frac{1}{\sqrt{2}} |111\rangle$$

(i) mit Wkeit $(\frac{1}{\sqrt{2}})^2 + (\frac{1}{\sqrt{2}})^2 = \frac{1}{2} + \frac{1}{2} = \frac{3}{4}$ wird $|10\rangle$ angenommen. Das Register ist dann im Zustand

$$\begin{aligned}\underbrace{\frac{1}{\sqrt{2}} |1000\rangle + \frac{1}{\sqrt{2}} |101\rangle}_{\sqrt{\frac{3}{4}}} &= \frac{1}{\sqrt{2}} \cdot \frac{2}{\sqrt{3}} |1000\rangle + \frac{1}{\sqrt{2}} \cdot \frac{2}{\sqrt{3}} |101\rangle = \frac{2}{\sqrt{6}} |1000\rangle + \frac{2}{\sqrt{6}} |101\rangle \\ &= \frac{\sqrt{2}}{\sqrt{3}} |1000\rangle + \frac{1}{\sqrt{3}} |101\rangle = \underline{\frac{\sqrt{2}}{\sqrt{3}} |1000\rangle + \frac{1}{\sqrt{3}} |101\rangle}\end{aligned}$$

(ii) mit Wkeit $(\frac{1}{\sqrt{2}})^2 = \frac{1}{4}$ wird $|1\rangle$ angenommen

$$\frac{1}{\sqrt{2}} |111\rangle = |111\rangle$$

Verschränkung

Im Abschnitt über das Tensorprodukt hatten wir bereits festgestellt, dass sich

$$\frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle) = |+\rangle \otimes |-\rangle$$

faktorisieren lässt (bzgl. dem Tensorprodukt), während (wir zumindest behauptete hatten, dass) dies für

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)$$

nicht gilt. Den ersten Zustand hatten wir *separabel* genannt, den zweiten *verschränkt*.

Verschränkung

Wir setzen die zuvor aufgeschobene Diskussion verschränkter Zustände nun fort.

Definition (Verschränkung)

Sei $|\phi\rangle$ der Zustand eines Quantenregisters aus n Bits. Der Zustand $|\phi\rangle$ heißt *unverschränkt* oder *separabel*, wenn er das Produkt von Zuständen einzelner Qubits ist, d.h.

$$|\phi\rangle = |\phi_{n-1}\rangle \otimes |\phi_{n-2}\rangle \otimes \dots \otimes |\phi_1\rangle.$$

Ein Zustand heißt *verschränkt*, wenn es keine solche Zerlegung gibt.

Verschränkung

Messung eines einzelnen Qubits in einem Produktzustand hat keine Auswirkungen auf die übrigen Qubits.

Messen wir das linke Qubit in

$$|+\rangle|-\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) |-\rangle$$

erhalten wir $|0\rangle$ oder $|1\rangle$ jeweils mit Wahrscheinlichkeit $\frac{1}{2}$.
Folgezustände sind $|0\rangle|-\rangle$ oder $|1\rangle|-\rangle$.

Das rechte Qubit $|-\rangle$ bleibt unverändert und wird durch die Messung des linken Qubits nicht beeinflusst.

Verschränkung

Messung eines einzelnen Qubits in einem verschränkten Zustand kann die anderen Qubits beeinflussen.

Wir betrachten

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle).$$

Messen wir das linke Qubit, erhalten wir $|0\rangle$ oder $|1\rangle$ jeweils mit Wahrscheinlichkeit $\frac{1}{2}$. Folgezustände sind $|00\rangle$ oder $|11\rangle$.

Messen wir erneut das linke Qubit (im Folgezustand) und erhalten $|0\rangle$, dann muss auch das rechte Qubit $|0\rangle$ sein.

Analog: Messen wir das linke Qubit (im Folgezustand) und erhalten $|1\rangle$, dann muss auch das rechte Qubit $|1\rangle$ sein.

Verschränkung

Im vorherigen Beispiel hat $|\Phi^+\rangle$ den größtmöglichen Grad an Verschränkung, da die Messung eines einzelnen Qubits die übrigen Qubits vollständig bestimmt. In einem solchen Fall nennt man die Zustände **maximal verschränkt**.

Für zwei Qubits gibt es vier maximal verschränkte Zustände, die sog. **Bell-Zustände**

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \quad |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \quad |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

Verschränkung

Übung: Zeigen Sie, dass $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ nicht in das Produkt zweier Zustände jeweils eines Qubits zerlegt werden kann.

Angenommen die Zerlegung des Produktes wäre möglich:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \stackrel{!}{=} (\alpha_0|0\rangle + \alpha_1|1\rangle)(\beta_0|0\rangle + \beta_1|1\rangle)$$

Nach Koeffizientenvergleich muss gelten:

$$\alpha_0\beta_0 = \alpha_1\beta_1 = \frac{1}{\sqrt{2}} \quad \text{und} \quad \alpha_0\beta_1 = \alpha_1\beta_0 = 0 \quad \xi$$

⇒ Also ist die Darstellung als Produkt nicht möglich

Verschränkung

Übung: Zeigen Sie, dass $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ nicht in das Produkt zweier Zustände jeweils eines Qubits zerlegt werden kann.

Um $|\Phi^+\rangle$ zu erzeugen, kann man CNOT auf den unverschränkten Zustand $\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$ anwenden und so einen verschränkten Zustand erhalten.

CNOT erzeugt also Verschränkung.

Übung: Zerlegen Sie $\frac{1}{2}(|0\rangle + |3\rangle + |12\rangle + |15\rangle)$ in ein Produkt von Bell-Zuständen.

Verschränkung

Betrachten wir den verschränkten Zustand

$$\frac{\sqrt{3}}{2\sqrt{2}}|00\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|01\rangle + \frac{\sqrt{3}}{4}|10\rangle + \frac{1}{4}|11\rangle$$

und messen das linke Qubit, erhalten wir

- $|0\rangle$ mit Wahrscheinlichkeit $\frac{3}{4}$ und den Folgezustand

$$\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

- $|1\rangle$ mit Wahrscheinlichkeit $\frac{1}{4}$ und den Folgezustand

$$\frac{\sqrt{3}}{2}|10\rangle + \frac{1}{2}|11\rangle = |1\rangle \otimes \left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle \right)$$

Verschränkung

Messung des linken Qubits hat also das rechte Qubit beeinflusst und in einem Fall erhalten wir $|0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ als Folgezustand, im anderen Fall $|1\rangle \otimes \left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle\right)$.

Wird nun das rechte Qubit gemessen, erhalten wir für $|0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ die Folgezustände $|0\rangle$ oder $|1\rangle$.

Im Fall von $|1\rangle \otimes \left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle\right)$ erhalten wir $|0\rangle$ mit Wahrscheinlichkeit $\frac{3}{4}$ und $|1\rangle$ mit Wahrscheinlichkeit $\frac{1}{4}$.

Verschränkung

Messung des linken Qubits hat also das rechte Qubit beeinflusst, aber nicht vollständig bestimmt was eine Messung des rechten Qubits ergeben würde.

Der ursprüngliche Zustand ist also nicht maximal verschränkt. Wir nennen einen solchen Zustand *teilweise verschränkt*.

Maximal verschränkte Zustände, wie die Bell-Zustände, sind für unsere Anwendungen interessanter.

Dichte Kodierung

Ausgangssituation: Alice möchte Bob klassische Informationen senden. Genauer: Alice will eine der vier möglichen Nachrichten $N \in \{00, 01, 10, 11\}$ senden. Sie kann dafür einen klassischen Kanal oder einen Quantenkanal verwenden.

Klassischer Kanal: Alice sendet Bob zwei Bits. Weniger als zwei Bits sind nicht möglich, da sie diese benötigt um eine der Kombinationen 00, 01, 10, 11 darzustellen.

Dichte Kodierung

Dichte Kodierung ermöglicht es, mit Hilfe eines Quantenkanals und eines Qubits *zwei* klassische Bits zu übertragen, d.h. es findet eine *Komprimierung* von *zwei* klassischen Bits auf ein Qubit statt.

Erkenntnis: Mit einem Qubit lässt sich unter bestimmten Umständen die **doppelte Informationsmenge** wie mit einem **klassischen Bit** übertragen.

Dichte Kodierung

Ausgangssituation: Alice besitzt ein Qubit $|a\rangle$ und Bob ein Qubit $|b\rangle$, die sich in einem verschränkten Zustand

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) = |ab\rangle$$

befinden.

$$\frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)$$

Alice  Bob

Dichte Kodierung

Kodierung Alice

Alice wendet je nach zu übermittelnder Nachricht eine der folgenden Operationen an:

- $N = 00$, keine Operation auf $|a\rangle$
- $N = 01$, dann $|a\rangle \leftarrow X|a\rangle$
- $N = 10$, dann $|a\rangle \leftarrow Z|a\rangle$
- $N = 11$, dann $|a\rangle \leftarrow ZX|a\rangle$ *//erst X-Gatter, dann Z-Gatter*

Anschließend schickt Alice ihr Qubit $|a\rangle$ an Bob.

Dichte Kodierung

Wir beobachten:

- Für $N = 00$ ist unverändert $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- Für $N = 01$ ist $X|0\rangle = |1\rangle$ und $X|1\rangle = |0\rangle$. Dadurch wird $|\Phi^+\rangle$ zu $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)$.
- Für $N = 10$, ist $Z|0\rangle = |0\rangle$ und $Z|1\rangle = -|1\rangle$. Dadurch wird $|\Phi^+\rangle$ zu $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$.
- Für $N = 11$, ist $ZX|0\rangle = -|1\rangle$ und $ZX|1\rangle = |0\rangle$. Dadurch wird $|\Phi^+\rangle$ zu $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$.

Dichte Kodierung

Dekodierung Bob

Bob wendet CNOT gefolgt von $H \otimes I$ an, um die Information von Alice zu dekodieren.

Es ist

$$|\Phi^+\rangle \xrightarrow{CNOT} \frac{1}{\sqrt{2}} (|00\rangle + |10\rangle) = |+\rangle|0\rangle \xrightarrow{H \otimes I} |00\rangle$$

$$|\Psi^+\rangle \xrightarrow{CNOT} \frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) = |+\rangle|1\rangle \xrightarrow{H \otimes I} |01\rangle$$

$$|\Phi^-\rangle \xrightarrow{CNOT} \frac{1}{\sqrt{2}} (|00\rangle - |10\rangle) = |-\rangle|0\rangle \xrightarrow{H \otimes I} |10\rangle$$

$$|\Psi^-\rangle \xrightarrow{CNOT} \frac{1}{\sqrt{2}} (|01\rangle - |11\rangle) = |-\rangle|1\rangle \xrightarrow{H \otimes I} |11\rangle$$

Dichte Kodierung

Auf maximal verschränkte Zustände setzt auch das Verfahren der *Quantenteleportation*⁴.

Während bei der *dichten Kodierung* klassische Information mit Hilfe eines Quantenkanals übertragen wurde, wird bei der Quantenteleportation Quanteninformation mit Hilfe eines klassischen Kanals übertragen.

Zurück zu Quantenalgorithmen: Wir wollen als nächstes die Grundidee vom Algorithmus von Deutsch auf größere Register von Qubits übertragen und benötigen dafür die Hadamard Transformation auf Registern.

⁴ vgl. Übungsaufgabe

Algorithmus von Deutsch-Jozsa

Aus der Analyse des n -Bit Zufallsgenerators ist

$$H_n |0\dots0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$$

bekannt. Das ist offensichtlich der Spezialfall für das Register $R = |0\dots0\rangle$ der Länge n .

Ein allgemeineres Verständnis von H_n ermöglicht uns die Diskussion fortgeschrittenerer Algorithmen bzw. Schaltkreise.

Algorithmus von Deutsch-Jozsa

$$\begin{aligned}|+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\|-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\|+\rangle &\approx H|0\rangle \\|-\rangle &\approx H|1\rangle\end{aligned}$$

Wir untersuchen die Wirkung von H_n auf ein Register

$R = |x_{n-1}...x_0\rangle$ mit $x_i \in \{0, 1\}$ für $i = 0, \dots, n - 1$ und beginnen mit dem Fall $n = 2$:

$$\begin{aligned}H_2|x_1x_0\rangle &= (H \otimes H)|x_1x_0\rangle \\&= \frac{1}{2}(|0\rangle + (-1)^{x_1}|1\rangle)(|0\rangle + (-1)^{x_0}|1\rangle)\end{aligned}$$

und mit $(-1)^{x_0}(-1)^{x_1} = (-1)^{x_0 \oplus x_1}$ folgt für $x = (x_1, x_0)^T$

Algorithmus von Deutsch-Jozsa

$$\begin{aligned}H_2|x_1x_0\rangle &= \frac{1}{2} (|00\rangle + (-1)^{x_0}|01\rangle + (-1)^{x_1}|10\rangle + (-1)^{x_0 \oplus x_1}|11\rangle) \\&= \frac{1}{2}((-1)^{(0,0)\cdot\mathbf{x}}|00\rangle + (-1)^{(0,1)\cdot\mathbf{x}}|01\rangle + (-1)^{(1,0)\cdot\mathbf{x}}|10\rangle + \\&\quad + (-1)^{(1,1)\cdot\mathbf{x}}|11\rangle),\end{aligned}$$

wenn \cdot das Skalarprodukt zweier Vektoren bezeichnet.

Allgemein ist für zwei Vektoren $x, y \in \{0, 1\}^n$ das Skalarprodukt $x \cdot y$ durch $\bigoplus_{i=1}^n x_i y_i$ gegeben. Auf ein n -Bit Register im Zustand $\mathbf{x} \in \{0, 1\}^n$ hat die Hadamard-Transformation H_n die Wirkung

$$H_n|\mathbf{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{y} \in \{0,1\}^n} (-1)^{\mathbf{x} \cdot \mathbf{y}} |\mathbf{y}\rangle.$$

Algorithmus von Deutsch-Jozsa

Dies ist eine Superposition über alle klassischen Zustände des Registers. Die Information über $|x\rangle$ wird in die Amplitude verlagert.

Beispiel: Als *gleichgewichtete* Superposition bezeichnet man

$$H_n|0\dots0\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^0 |y\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} |y\rangle.$$

Die *alternierende* Superposition ist für $y = (y_{n-1}, \dots, y_0)$

$$H_n|0\dots01\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{y_0} |y\rangle,$$

Algorithmus von Deutsch-Jozsa

wobei für die Summe gilt

$$\begin{aligned} \sum_{y=0}^{2^n-1} (-1)^{y0}|y\rangle &= (-1)^0|0\dots00\rangle + (-1)^1|0\dots01\rangle + \\ &\quad + (-1)^0|0\dots010\rangle + (-1)^1|0\dots011\rangle + \dots \\ &\quad + (-1)^0|1\dots10\rangle + (-1)^1|1\dots11\rangle \\ &= |0\dots00\rangle - |0\dots01\rangle + |0\dots010\rangle - |0\dots011\rangle + \dots \\ &\quad + |1\dots10\rangle - |1\dots11\rangle. \end{aligned}$$

Die allgemeine Formel für die Hadamard Transformation werden wir noch häufig benötigen.

Algorithmus von Deutsch-Jozsa

Wollen wir Quantenalgorithmen besser verstehen und mögliche Vorteile identifizieren benötigen wir Vergleichsgrößen.

Für Schaltkreise wäre es am präzisesten, die kleinstmögliche Anzahl von Gattern (bzgl. einer Gatterbasis) zu zählen, die zum Lösen einer Problemstellung benötigt werden. Diese Größe nennt man *Schaltkreiskomplexität*.

In diesem Sinne wäre ein Quantenalgorithmus *effizient*, wenn er **polynomiale Schaltkreiskomplexität** hätte.

Die Schaltkreiskomplexität zu bestimmen ist mitunter schwer.

Algorithmus von Deutsch-Jozsa

Wir verwenden die *Anfragekomplexität* (*query complexity*).

Das ist die Anzahl der Aufrufe bzw. Anfragen an eine Funktion bzw. ein Orakel, die benötigt wird um ein Problem zu lösen.

Das Orakel ist für uns eine Black Box, an die wir einen Input übergeben und die uns eine Output zurück liefert, ohne das wir näher wissen was im Inneren des Orakel passiert. Wir können dies als eine externe (Quanten-)Subroutine verstehen.

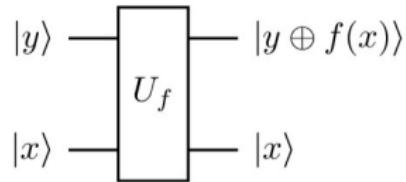
Algorithmus von Deutsch-Jozsa

Tatsächlich kennen wir diese Arbeitsweise schon vom Algorithmus von Deutsch. Dort wurde U_f verwendet, für $f : \{0, 1\} \rightarrow \{0, 1\}$.

Ein Quantenorakel ist

$$U_f : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle.$$

Das Qubit $|y\rangle$ nennt man
Antwortqubit oder *target Qubit*;
 $|x\rangle$ nennt man *input Qubit*.



Algorithmus von Deutsch-Jozsa

Bei zuvor beschriebener Anfrage an ein Orakel wird das input Qubit $|x\rangle$ nicht verändert, während das target Qubit von $|y\rangle$ zu $|y \oplus f(x)\rangle$ übergeht.

Im Algorithmus von Deutsch haben wir (indirekt) schon eine Möglichkeit kennengelernt um Anfragen zu stellen, bei denen das target Qubit unverändert bleibt, während das input Qubit $|x\rangle$ mit einer Phase multipliziert wird:

Initialisiere das target Qubit mit $|0\rangle$ und

$$|x\rangle|0\rangle \xrightarrow{I \otimes X} |x\rangle|1\rangle \xrightarrow{I \otimes H} |x\rangle|-\rangle.$$

Algorithmus von Deutsch-Jozsa

Dies führt weiter zu

$$\begin{aligned}|x\rangle|-\rangle &= |x\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}} (|x\rangle|0\rangle - |x\rangle|1\rangle) \\&\xrightarrow{U_f} \frac{1}{\sqrt{2}} (|x\rangle|f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle) \\&= \begin{cases} \frac{1}{\sqrt{2}} (|x\rangle|0\rangle - |x\rangle|1\rangle) = |x\rangle|-\rangle & \text{falls } f(x) = 0 \\ \frac{1}{\sqrt{2}} (|x\rangle|1\rangle - |x\rangle|0\rangle) = -|x\rangle|-\rangle & \text{falls } f(x) = 1 \end{cases} \\&= (-1)^{f(x)}|x\rangle|-\rangle\end{aligned}$$

Da die Funktionsauswertung des input Qubit nun als Phase auftritt, nennt man dies auch **phase kickback**.

Algorithmus von Deutsch-Jozsa

Manchmal wird auch die Bezeichnung **Phasenorakel** verwendet.

Das ist insbesondere dann der Fall, wenn die Notation von $|-\rangle$ vernachlässigt wird und man abkürzend

$$|x\rangle \xrightarrow{U_f} (-1)^{f(x)}|x\rangle$$

schreibt.

Algorithmus von Deutsch-Jozsa

Ausgangssituation: Gegeben sei eine Funktion

$f : \{0,1\}^n \rightarrow \{0,1\}$ für die genau eine der folgenden beiden Möglichkeiten gilt:

- f ist *konstant*, d.h. alle Eingaben werden auf die gleiche Ausgabe abgebildet, oder
- f ist *balanciert*, d.h. die Hälfte der Eingaben wird auf 1 abgebildet und die andere Hälfte wird auf 0 abgebildet. Also $|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$ für die Urbilder.

Das Quantenorakel ist

$\downarrow i^n$ } ist es 0
 i } ist es 1

$$U_f : |x_{n-1} \dots x_0, y\rangle \mapsto |x_{n-1} \dots x_0, y \oplus f(x)\rangle.$$

Frage: Ist f konstant oder balanciert?

Algorithmus von Deutsch-Jozsa

Klassische Lösung: Um mit Sicherheit zu bestimmen welche der Möglichkeiten der Fall ist, müssen wir die Hälfte der Inputs abfragen, plus einen zusätzlichen Input. Wurde für die Hälfte der Inputs eine 0 als Ausgabe erhalten, dann löst die zusätzliche Abfrage eines weiteren Inputs die Frage, ob alle Outputs 0 sind oder nicht.

Für einen binären string der Länge n gibt es 2^n mögliche Belegungen, d.h. die Anfragekomplexität an f ist

$$2^{n-1} + 1 = O(2^n)$$

also exponentiell.

Algorithmus von Deutsch-Jozsa

Es gibt probabilistische Algorithmen (vgl. BPP), die die richtige Antwort mit einem beschränkten Fehler in einer konstanten Anzahl der Anfragen an f liefern. In diesem Fall ist die Komplexität $O(1)$.

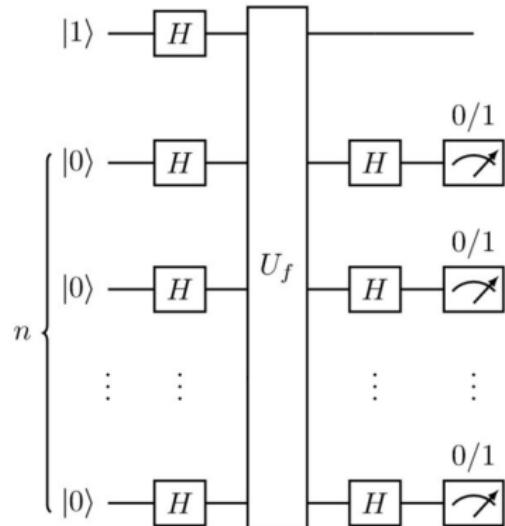
Quantenalgorithmus: Der Algorithmus von Deutsch-Jozsa löst das Problem mit *genau einer* Anfrage.

Das ist eine exponentielle Beschleunigung gegenüber dem exakten klassischen Lösungsansatz, aber nicht gegenüber einem probabilistischen klassischen Algorithmus.

Algorithmus von Deutsch-Jozsa

Algorithmus von Deutsch-Jozsa

1. $|x_{n-1} \dots x_0\rangle |y\rangle \leftarrow |0 \dots 0\rangle |1\rangle$
2. $|x\rangle |y\rangle \leftarrow H_{n+1} |x\rangle |y\rangle$
3. $|x\rangle |y\rangle \leftarrow U_f |x\rangle |y\rangle$
4. $|x\rangle |y\rangle \leftarrow (H_n |x\rangle) |y\rangle$
5. Messe das Register $|x\rangle$:
 - Ist $|x\rangle = |0 \dots 0\rangle$: f ist konstant
 - Sonst: f ist balanciert



Algorithmus von Deutsch-Jozsa

Analyse des Algorithmus: In Schritt 2 ergibt sich

$$|\phi_2\rangle = H_n|0\dots0\rangle H|1\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \right) |- \rangle.$$

Wir erkennen die gleichgewichtete Superposition. Mittels phase kickback ist

$$U_f(|x\rangle |- \rangle) = (-1)^{f(x)}|x\rangle |- \rangle.$$

Algorithmus von Deutsch-Jozsa

Damit folgt in **Schritt 3**

$$\begin{aligned} |\phi_3\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} U_f(|x\rangle |-\rangle) \\ &= \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \right) |-\rangle. \end{aligned}$$

In **Schritt 4** erfolgt erneute Anwendung von H_n (diesmal nur auf n Register):

$$|\phi_4\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} H_n |x\rangle \right) |-\rangle.$$

Algorithmus von Deutsch-Jozsa

Entsprechend der Vorbetrachtung zur Hadamard- Transformation auf **allgemeinen** Registern ist

$$\begin{aligned} |\phi_4\rangle &= \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \left(\frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle \right) \right) |- \rangle \\ &= \left(\sum_{z \in \{0,1\}^n} \left(\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)+x \cdot z} \right) |z\rangle \right) |- \rangle. \end{aligned}$$

Um zu verstehen wie eine Messung uns bestimmen lässt, ob die Funktion konstant oder balanciert ist, berechnen wir die Wahrscheinlichkeit den Zustand $|z\rangle = |0\dots00\rangle$ zu messen.

Algorithmus von Deutsch-Jozsa

Zu beachten ist, dass nicht alle Qubits gemessen werden (vgl. Beschreibung des Algorithmus).

Die Amplitude von $|0\dots00\rangle$ vor der Messung ist

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)+x \cdot 0} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)}.$$

Dies bestimmt schon, ob f konstant oder balanciert ist.

Algorithmus von Deutsch-Jozsa

1. Fall: Ist f konstant, dann nimmt $f(x)$ immer den gleichen Wert an, also $f(x) = f(0\dots 00)$ für alle $x \in \{0,1\}^n$. In diesem Fall ist

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(0\dots 00)} = (-1)^{f(0\dots 00)}.$$

Ist f konstant, dann ist die Wahrscheinlichkeit $|0\dots 00\rangle$ zu messen, also 1.

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(0\dots 00)} = \frac{(-1)^{f(0\dots 00)}}{2^n} \cdot \sum_{x \in \{0,1\}^n} 1 = \frac{(-1)^{f(0\dots 00)}}{2^n} \cdot 2^n = (-1)^{f(0\dots 00)}$$

Algorithmus von Deutsch-Jozsa

2. Fall: Ist f **balanciert**, dann ist $(-1)^{f(x)}$ in der Hälfte der Fälle 1 und in der anderen Hälfte der Fälle -1 . In diesem Fall ist die Summe alternierend und

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} = 0.$$

Ist f balanciert, dann ist die Wahrscheinlichkeit $|0\dots00\rangle$ zu messen demnach 0. Wir erhalten bei Messung also garantiert etwas anderes als $|0\dots00\rangle$.

Algorithmus von Deutsch-Jozsa

Um zu bestimmen ob f konstant oder balanciert ist, genügt es also n Qubits zu messen.

Erhalten wir dabei $|0\dots00\rangle$, dann ist f konstant. Ergibt unsere Messung irgendeinen anderes Ergebnis, dann ist f balanciert.

Um die Ausgangsfrage zu beantworten, war genau eine Anfrage an das Quantenorakel notwendig.

Algorithmus von Bernstein-Vazirani

Ein weiterer Fragestellung die sich mit derselben Prozedur wie im Deutsch-Jozsa Algorithmus lösen lässt, ist die Frage nach der Bestimmung eines unbekannten n -Bit Strings, der als Punktprodukt gegeben ist.

Gegeben $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Wir wissen das $f(x) = s \cdot x$, wobei $s = s_{n-1} \dots s_1 s_0$ ein unbekannter String ist und (wie üblich)

$$s \cdot x = s_{n-1}x_{n-1} \oplus \dots \oplus s_1x_1 \oplus s_0x_0.$$

Problemstellung: Bestimme $s = s_{n-1} \dots s_1 s_0$.

Algorithmus von Bernstein-Vazirani

Klassische Lösung: Es werden n Anfragen an f benötigt um jedes Bit von s zu lernen.

Für $n = 3$ sind dies die Anfragen

$$f(001) = s_2 \cdot 0 + s_1 \cdot 0 + s_0 \cdot 1 = s_0$$

$$f(010) = s_2 \cdot 0 + s_1 \cdot 1 + s_0 \cdot 0 = s_1$$

$$f(100) = s_2 \cdot 1 + s_1 \cdot 0 + s_0 \cdot 0 = s_2$$

Auch klassische probabilistische Algorithmen benötigen für dieses Problem mindestens n Anfragen an f .

Algorithmus von Bernstein-Vazirani

Quantenalgorithmus: Der Algorithmus von Bernstein-Vazirani benötigt *genau eine* Anfrage. Der Algorithmus bzw. Schaltkreis ist exakt der gleiche wie im Algorithmus von Deutsch-Jozsa, nur das nun eine spezielle Form von f vorgegeben ist.

Die Analyse des Algorithmus ist eine Aufgabe auf dem Übungsblatt.

Da klassische Algorithmen n Anfragen benötigen, der Algorithmus von Bernstein-Vazirani jedoch nur eine einzige Anfrage, liefert dieser eine *polynomielle* Beschleunigung gegenüber klassischen Computern.

Algorithmus von Simon

In der nächsten Problemstellung liefert der Quantenalgorithmus tatsächlich eine exponentielle Beschleunigung.

Gegeben $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$, d.h. f erwartet einen n -Bit String $x = x_{n-1} \dots x_1 x_0$ und liefert einen n -Bit String $f(x) = f_{n-1} \dots f_1 f_0$. Wir wissen das $f(x) = f(y)$ genau dann gilt, wenn die beiden Inputs x und y durch $x = y \oplus s$ bzw. $y = x \oplus s$ in Verbindung stehen, für einen unbekannten n -Bit String $s = s_{n-1} \dots s_1 s_0 \neq 0 \dots 00$ (wobei \oplus bitweises XOR bezeichnet).

Also gilt $f(x) = f(y)$ genau dann, wenn $x_i = y_i \oplus s_i$ bzw. $y_i = x_i \oplus s_i$.

Problemstellung: Bestimme den String $s = s_{n-1} \dots s_1 s_0$.

Algorithmus von Simon

Den String s nennt man auch *Maske* und da diese durch ein XOR Verknüpft wird, spricht man auch von einer *XOR-Maske*.

Klassische Lösung: Wir können s durch Wertekollision ermitteln, d.h. wir benötigen x und y mit $f(x) = f(y)$. Aus unserem Wissen über f folgt dann $x = y \oplus s$ bzw. $y = x \oplus s$ und

$$x \oplus y = x \oplus (x \oplus s) = (x \oplus x) \oplus s = s.$$

Zur Ermittlung einer Kollision werden $\sqrt{2^n} = O\left(2^{\frac{n}{2}}\right)$ Anfragen (d.h. exponentiell viele Anfragen) an f benötigt.

Algorithmus von Simon

Quantenalgorithmus: Der Algorithmus von Simon löst das Problem mit $O(n)$ Anfragen an das Orakel und folgt dem bisherigen Schema: Anwendung von Hadamard-Gattern, Orakelanfrage und erneute Anwendung von Hadamard-Gattern.

Das Orakel operiert nun auf Strings, d.h. wir verwenden n input Qubits und n target Qubits, also $|x\rangle = |x_{n-1} \dots x_1 x_0\rangle$, $|y\rangle = |y_{n-1} \dots y_1 y_0\rangle$ und

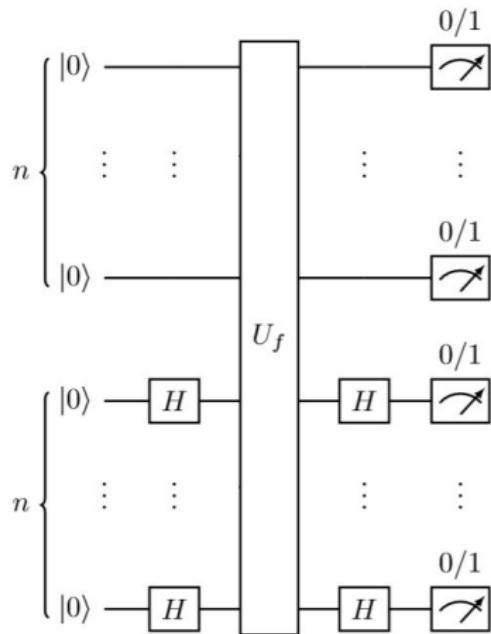
$$|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$$

mit $|y \oplus f(x)\rangle = |y_{n-1} \oplus f_{n-1}, \dots, y_1 \oplus f_1, y_0 \oplus f_0\rangle$.

Algorithmus von Simon

Algorithmus von Simon

1. $|x_{n-1} \dots x_0\rangle |y_{n-1} \dots y_0\rangle \leftarrow |0\rangle^{\otimes n} |0\rangle^{\otimes n}$
2. $|x\rangle |y\rangle \leftarrow (H_n |x\rangle) |y\rangle$
3. $|x\rangle |y\rangle \leftarrow U_f |x\rangle |y\rangle$
4. $|x\rangle |y\rangle \leftarrow (H_n |x\rangle) |y\rangle$
5. Messe Register $|y\rangle$, dann $|x\rangle$
6. Wdh. Schritt 1-5 $O(n)$ mal, stelle ein LGS (n Variablen, n Unbekannte) auf.
7. Lösung des LGS ergibt s .



Algorithmus von Simon

Im Algorithmus bilden Schritt 1-5 den Quantenanteil, die Schritte 6 & 7 den klassischen Anteil zur Nachbearbeitung.

Analyse des Algorithmus: Die Anwendung von Hadamard Gattern in **Schritt 2** liefert im Teilregister die gleichgewichtete Superposition

$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0\dots0\rangle$$

und in **Schritt 3** ergibt die Anfrage an das Quantenorakel

$$|\phi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle.$$

Algorithmus von Simon

Erneute Anwendung von Hadamard Gattern in **Schritt 4** führt auf den bereits bekannten Ausdruck

$$\begin{aligned} |\phi_4\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} H^{\otimes n} |x\rangle |f(x)\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle |f(x)\rangle. \end{aligned}$$

Messen wir die target Qubits $|f(x)\rangle$, dann erhalten wir einen speziellen Wert f' . Für diesen gibt es (per Definition von f) zwei mögliche Belegungen von x , für die $f(x) = f'$ gilt, wir nennen diese x' und x'' . Also $f(x') = f(x'') = f'$ und x', x'' liefern eine Kollision.

Algorithmus von Simon

Durch Messung findet in **Schritt 5** ein Übergang von $|\phi_4\rangle$ zu

$$\begin{aligned} & \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} \left((-1)^{x' \cdot z} + (-1)^{x'' \cdot z} \right) |z\rangle |f'\rangle \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{z \in \{0,1\}^n} \left((-1)^{x' \cdot z} + (-1)^{x'' \cdot z} \right) |z\rangle |f'\rangle \end{aligned}$$

statt. Dann werden die input Qubits gemessen. Abhängig von dem Ergebnis des Produktes im Exponenten ist $(-1)^{x' \cdot z} = \pm 1$ und $(-1)^{x'' \cdot z} = \pm 1$.

Algorithmus von Simon

Die Summe dieser Werte ist

$$(-1)^{x' \cdot z} + (-1)^{x'' \cdot z} = \begin{cases} \pm 2 & \text{falls } x' \cdot z = x'' \cdot z \pmod{2} \\ 0 & \text{falls } x' \cdot z \neq x'' \cdot z \pmod{2} \end{cases}$$

Bei Messung der input Qubits erhalten also nur dann ein Messergebnis, wenn $x' \cdot z = x'' \cdot z \pmod{2}$, bzw.

$$(x' + x'') \cdot z = 0 \pmod{2}.$$

Da für x' und x'' Kollision stattfindet, ist $x' \oplus x'' = s$. Anders,

$$s \cdot z = 0 \pmod{2}.$$

Algorithmus von Simon

Aus Schritt 5 erhalten wir also einen Wert von $|z\rangle = |z_{n-1} \dots z_1 z_0\rangle$, für den gilt

$$s_{n-1}z_{n-1} + \dots + s_1z_1 + s_0z_0 = 0 \pmod{2} \quad (1)$$

Das ist eine Gleichung den n unbekannten Bestandteilen s_i von $s = s_{n-1} \dots s_1 s_0$.

Wiederholung der vorherigen Schritte liefert erneut ein $|z\rangle$ das (1) erfüllt. Dieses ist höchstwahrscheinlich von der vorherigen Belegung von $|z\rangle$ verschieden, denn die Messwahrscheinlichkeit für ein $|z\rangle$ ist

$$\left| \frac{\pm 2}{\sqrt{2^{n+1}}} \right|^2 = \frac{4}{2^{n+1}} = \frac{1}{2^{n-1}}.$$

Algorithmus von Simon

Da es 2^{n-1} Möglichkeiten für $|z\rangle$ gibt, von denen jede ein verschwindendes Produkt mit s hat, ist die Wahrscheinlichkeit gering zweimal gleiches $|z\rangle$ zu erhalten.

Wiederholen wir in **Schritt 6** den vorherigen Prozess $O(n)$ mal, ergeben sich verschiedene $|z\rangle$ die (1) erfüllen. Diese bilden ein Gleichungssystem von n Variablen und n Unbekannten.

Lösung des Gleichungssystems liefert $s = s_{n-1} \dots s_1 s_0$. Dafür wurden insgesamt $O(n)$ Anfragen an das Quantenorakel benötigt. Das ist eine *exponentielle* Beschleunigung gegenüber dem klassischen Lösungsansatz.

Lösen des Gleichungssystems mittels Gaußschem Eliminationsverfahren benötigt $O(n^3)$ Schritte.

Algorithmus von Simon

Wir fassen noch einmal zusammen:

Problem	klassische Anfragen	Quanten-algorithmen	Quanten-anfragen	asymptotische Beschleunigung
konstant vs. balanciert	Exakt: $2^{n-1} + 1$ Prob. $O(1)$	Deutsch-Jozsa	1	exponentiell keine
Punktprodukt	n	Bernstein-Vazirani	1	polynomial
XOR Maske	$O\left(2^{\frac{n}{2}}\right)$	Simon	$O(n)$	exponentiell

Deutsch-Jozsa höherer Ordnung

Exkurs: Deutsch-Jozsa höherer Ordnung

Walsh-Transformation

Erinnerung: In der Analyse des Deutsch-Jozsa Algorithmus sind wir auf die (innere) Summe

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus x \cdot z}$$

gestoßen, die wir für $|z\rangle = |0\rangle^{\otimes n}$ weiter untersucht hatten. Hier ergeben sich weitere Anknüpfungspunkte an die Theorie der *kryptographischen Booleschen Funktionen*.

Walsh-Transformation

Walsh Transformation

Sei $f : \{0, 1\}^n \rightarrow \{0, 1\}$ und $z \in \{0, 1\}^n$. Dann heißt

$$W_f(z) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus z \cdot x}$$

die *Walsh-Transformation* von f , wenn das $z \cdot x$ das innere Produkt $z \cdot x = z_1 x_1 \oplus \dots \oplus z_n x_n$ bezeichnet.

Die Walsh-Transformation von f an der Stelle z nennt man auch den *Walsh Koeffizienten* von f an z und die Liste aller 2^n Walsh Koeffizienten $[W_f(0), W_f(1), \dots, W_f(2^n - 1)]$ das *Walsh Spektrum*.

Walsh-Transformation

Wir beobachten im Algorithmus von Deutsch-Jozsa

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus x \cdot z} = \frac{W_f(z)}{2^n}$$

und die Messwahrscheinlichkeit eines bel. Zustands $|z\rangle$ ist dann

$$\Pr(|z\rangle) = \left(\frac{W_f(z)}{2^n} \right)^2.$$

Das ist insbesondere aus kryptographischer Sicht interessant.

Walsh-Transformation

Eine Boolesche Funktion $f : \{0,1\}^n \rightarrow \{0,1\}$ ist *korrelationsimmun von Ordnung k*, wenn ihre Funktionswerte für jede k -elementige Inputmenge statistisch unabhängig sind.

Es ist also ein Maß dafür, ob und wie viel Information man aus dem Funktionswert der Booleschen Funktion über ihre Argumente gewinnen kann.

Eine Boolesche Funktion $f : \{0,1\}^n \rightarrow \{0,1\}$ ist genau dann korrelationsimmun von Ordnung k , wenn

$$W_f(z) = 0, \quad 1 \leq \text{wt}(z) \leq k.$$

Walsh-Transformation

Eine Boolesche Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ist *k-resilient*, wenn sie balanciert und korrelationsimmun von Ordnung k ist.

Eine Boolesche Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ist genau dann k-resilient, wenn

$$W_f(z) = 0, \quad 0 \leq \text{wt}(z) \leq k.$$

Messung des finalen Zustands im Algorithmus von Deutsch-Jozsa liefert uns Teilinformationen in Form von

$$\Pr(|z\rangle) = \left(\frac{W_f(z)}{2^n} \right)^2 := \hat{f}(z)$$

und wiederholte Ausführung des Algorithmus liefert eine Approximation für $\hat{f}(z)$.

Deutsch-Jozsa höherer Ordnung

Von ähnlicher Form ist die folgende Größe:

Definition (Autokorrelationskoeffizient)

Für eine Boolesche Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$ und $a \in \{0, 1\}^n$ heißt

$$\Delta f(a) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus f(x \oplus a)}$$

der Autokorrelationskoeffizient von f an der Stelle a .

Interpretation: Mittlere Korrelation zwischen Werten der Funktion an der Stelle x und x verschoben um a , entsprechend über alle Werte von x .

Deutsch-Jozsa höherer Ordnung

Die Liste der Autokorrelationskoeffizienten (an allen 2^n Punkten) nennt man das *Autokorrelationsspektrum*:

$$\Delta f = [\Delta f(0), \Delta f(1), \dots, \Delta f(2^n - 1)] .$$

In Verbindung dazu steht die Ableitung einer Booleschen Funktion.

Deutsch-Jozsa höherer Ordnung

Definition (Ableitung einer Booleschen Funktion)

Für eine Boolesche Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$ heißt

$$\Delta f_a(x) = f(x \oplus a) \oplus f(x)$$

die Ableitung erster Ordnung von f in Richtung $a \in \{0, 1\}^n$. Die k -te Ableitung von f an $\mathcal{A} = (a_1, \dots, a_k)$ (wobei $k \leq n$), ist

$$\Delta f_{\mathcal{A}}^{(k)}(x) = \Delta f_{a_k} \left(\Delta f_{a_1, \dots, a_{k-1}}^{(k-1)}(x) \right) = \bigoplus_{S \subseteq \mathcal{A}} f(x \oplus S)$$

Deutsch-Jozsa höherer Ordnung

In der Kryptoanalyse von Blockchiffren kommen Ableitungen erster (oder höherer) Ordnung vor. Der wesentliche Baustein einer Blockchiffre, die S-Box, kann als Boolesche Funktion verstanden werden.

Eine Verallgemeinerung des Deutsch-Jozsa Algorithmus liefert uns Ableitungen in den Amplituden der Qubits.

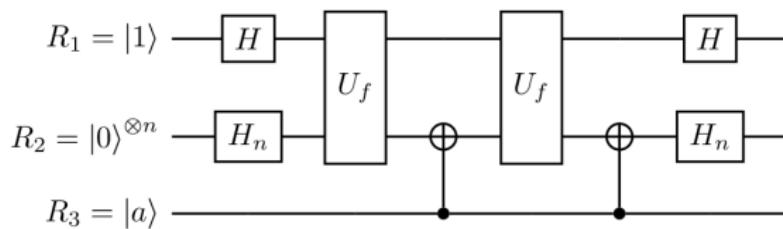
Deutsch-Jozsa höherer Ordnung

$HoDJ_n^k$ - Higher-order Deutsch-Jozsa Schaltkreis

- Quantenschaltkreis, der das Walsh-Spektrum der k -ten Ableitung einer (n -Bit Input) Booleschen Funktion an $\mathcal{A} = (a_1, \dots, a_k)$ erzeugt
- operiert auf $k + 2$ Registern $R_1, \dots, R_k, R_{k+1}, R_{k+2}$, wobei
 - R_1 aus einem Qubit besteht, initialisiert mit $|1\rangle$,
 - R_2 aus n Qubits besteht, jeweils initialisiert mit $|0\rangle$,
 - jedes der R_3, \dots, R_{k+2} aus jeweils n Qubits besteht, von denen R_{2+t} im Zustand $|a_t\rangle$ aus \mathcal{A} initialisiert wurde.

Deutsch-Jozsa höherer Ordnung

Wir betrachten den Fall $k = 1$, d.h. die Ableitung erster Ordnung:



Analyse: In Schritt 2 ist

$$|\phi_2\rangle = (H \otimes H_n \otimes I_2)|1\rangle|0\rangle^{\otimes n}|a\rangle = \frac{1}{\sqrt{2^n}} \sum_x |-\rangle|x\rangle|a\rangle.$$

Deutsch-Jozsa höherer Ordnung

Wir wissen bereits

$$U_f |x\rangle |-\rangle = (-1)^{f(x)} |x\rangle |-\rangle$$

(*phase kickback*) und beobachten das mittels SWAP $|a\rangle |b\rangle = |b\rangle |a\rangle$ folgt

$$(SWAP \circ U_f \circ SWAP) = (-1)^{f(x)} |-\rangle |x\rangle.$$

Häufig werden U_f und $SWAP \circ U_f \circ SWAP$ miteinander identifiziert; so auch in $HoDJ_k^n$.

Deutsch-Jozsa höherer Ordnung

Damit folgt in **Schritt 3**

$$|\phi_3\rangle = (U_f \otimes I_2)|\phi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |-\rangle |x\rangle |a\rangle.$$

Schritt 4 dient zur Vorbereitung

$$|\phi_4\rangle = (I_2 \otimes CNOT)|\phi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |-\rangle |x \oplus a\rangle |a\rangle$$

Deutsch-Jozsa höherer Ordnung

für die erneute Anwendung von U_f in **Schritt 5**

$$|\phi_5\rangle = (U_f \otimes I_2)|\phi_4\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x) \oplus f(x \oplus a)} |-\rangle |x \oplus a\rangle |a\rangle,$$

gefolgt von wiederholter Anwendung von CNOT in **Schritt 6**

$$\begin{aligned} |\phi_6\rangle &= (I_2 \otimes \text{CNOT})|\phi_5\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x) \oplus f(x \oplus a)} |-\rangle |x\rangle |a\rangle \\ &= \frac{1}{\sqrt{2^n}} |-\rangle \sum_x (-1)^{f(x) \oplus f(x \oplus a)} |x\rangle |a\rangle. \end{aligned}$$

Deutsch-Jozsa höherer Ordnung

Final ergibt sich in **Schritt 7**

$$\begin{aligned} |\phi_7\rangle &= (H \otimes H_n \otimes I_2) |\phi_6\rangle \\ &= |1\rangle \sum_y \left(\frac{1}{2^n} \sum_x (-1)^{x \cdot y} (-1)^{f(x) \oplus f(x \oplus a)} \right) |y\rangle |a\rangle \\ &= |1\rangle \sum_y \widehat{\Delta f_a(y)} |y\rangle |a\rangle. \end{aligned}$$

Register R_2 befindet sich mit Wahrscheinlichkeit $(\widehat{\Delta f_a(y)})^2$ im Zustand $|y\rangle$, d.h. die Amplitude enthält Informationen über die Ableitung der Booleschen Funktion f .

Deutsch-Jozsa höherer Ordnung

Darauf lässt sich weiter aufbauen, wobei es verschiedene Anwendungen der bisherigen Algorithmen im Kontext

- kryptographischer Boolescher Funktionen
- (Quanten-)Kryptoanalyse symmetrischer Verfahren
- Implementierungen symmetrischer Verfahren als quantenlogische Schaltungen (z.B. AES)

gibt. An dieser Stelle reicht die Zeit nicht um dieses Thema weiter zu vertiefen.

Inhaltsverzeichnis

1 Grundlagen

- Einleitung & Ein-Qubit Systeme
- n -Qubit Systeme & Grundlegende Schaltkreise
- Verschränkung, Deutsch-Jozsa, Bernstein-Vazirani & Simon

2 Quantensuche

- Algorithmus von Grover
- Varianten der Quantensuche

3 Fouriertransformation & Shor

- RSA & periodische Funktionen
- Diskrete- & Quanten-Fouriertransformation
- Algorithmus von Shor

4 Ausblick: QEC & QKD

Algorithmus von Grover

Wir betrachten nun einen Quantenalgorithmus für das Problem der *unstrukturierten Suche* in Datenmengen (auch *brute-force* Suche genannt).

Als Beispiel können wir uns die

- Suche nach kryptographischen Schlüsseln, oder das
- *umgekehrte Telefonbuchproblem* (finde anhand der Telefonnummer den Namen einer Person)

vorstellen.

Algorithmus von Grover

Für das Quantenorakel ist uns die Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$ mit

$$f(x) = \begin{cases} 1 & \text{falls } x = \omega \\ 0 & \text{sonst} \end{cases}$$

gegeben, die genau dann den Wert 1 liefert, wenn x der gesuchte Wert ω ist.

Problemstellung: Finde den Wert ω .

Algorithmus von Grover

Klassischer Ansatz: Im schlimmsten Fall müssen alle $N = 2^n$ möglichen Eingabewerte von f angefragt werden.

Im Schnitt müssen die Hälfte der möglichen Eingabewerte an f angefragt werden, d.h. es werden $\frac{N}{2}$ Anfragen an f gestellt.

In jedem Fall ist die Komplexität $O(N)$.

Algorithmus von Grover

Quantenalgorithmus: Der Algorithmus von Grover löst das Problem mit $O(\sqrt{N})$ Anfragen (*quadratische Beschleunigung*).

Wie unsere Analysen vorheriger Algorithmen gezeigt haben, ist im Fall eines *phase kickback* das *target* Qubit von besonderer Bedeutung, während auf dem *input* Qubit wenig geschieht.

Wir richten unseren Fokus also zunächst auf das *target* Qubit.

Algorithmus von Grover

Ausgangspunkt unserer Überlegungen ist wieder eine gleichgewichtete Superposition des Startzustands

$$|s\rangle = H^{\otimes n} |0\rangle^{\otimes n} = |+\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$$

für $N = 2^n$. Als Superposition über alle n -Bit Strings enthält diese zwangsläufig auch den gesuchten Wert $|\omega\rangle$.

Algorithmus von Grover

Also

$$\begin{aligned} |s\rangle &= \frac{1}{\sqrt{N}} \left(|\omega\rangle + \sum_{x \neq \omega} |x\rangle \right) = \frac{1}{\sqrt{N}} |\omega\rangle + \frac{1}{\sqrt{N}} \sum_{x \neq \omega} |x\rangle \\ &= \frac{1}{\sqrt{N}} |\omega\rangle + \sqrt{\frac{N-1}{N}} \cdot \underbrace{\frac{1}{\sqrt{N-1}} \sum_{x \neq \omega} |x\rangle}_{=:|r\rangle} \\ &= \frac{1}{\sqrt{N}} |\omega\rangle + \sqrt{\frac{N-1}{N}} |r\rangle \end{aligned}$$

wobei $|r\rangle$ die gleichgewichtete Superposition über alle n -Bit Strings ungleich $|\omega\rangle$ ist.

Algorithmus von Grover

Wegen

$$0 \leq \frac{1}{\sqrt{N}}, \quad \sqrt{\frac{N-1}{N}} \leq 1$$

und trig. Pythagoras, existiert ein θ mit

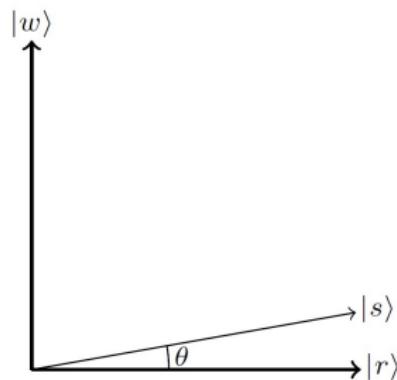
$$\sin(\theta) = \frac{1}{\sqrt{N}}, \quad \cos(\theta) = \sqrt{\frac{N-1}{N}}.$$

Also

$$|s\rangle = \sin(\theta)|\omega\rangle + \cos(\theta)|r\rangle$$

Algorithmus von Grover

Wir zeichnen den Zustand $|s\rangle$ in eine Koordinatenebene, die von $|r\rangle$ und $|w\rangle$ aufgespannt wird.

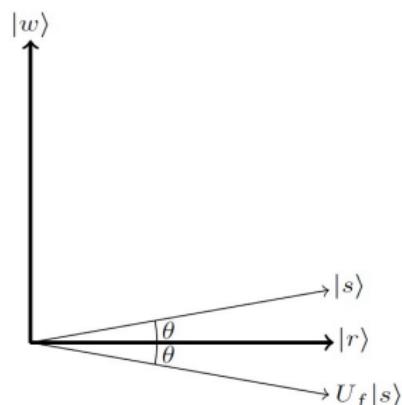


Algorithmus von Grover

Nun wird eine Orakelanfrage (*phase oracle*) ausgeführt. Wegen $f(x) = 1$ gdw. $x = \omega$, folgt

$$\begin{aligned} U_f |s\rangle &= (-1)^{f(\omega)} \sin(\theta)|\omega\rangle \\ &\quad + (-1)^{f(r)} \cos(\theta)|r\rangle \\ &= -\sin(\theta)|\omega\rangle + \cos(\theta)|r\rangle \end{aligned}$$

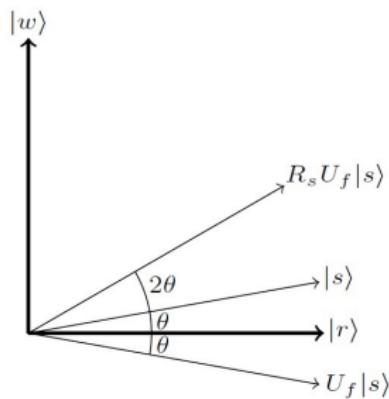
Die Amplitude von $|\omega\rangle$ wird invertiert. Auf Koordinatenebene entspricht dies einer Spiegelung durch $|\omega\rangle$.



Algorithmus von Grover

Angenommen wir hätten Zugriff auf ein Gatter R_s das eine Spiegelung um $|s\rangle$ ausführt (wir konstruieren ein solches Gatter noch).

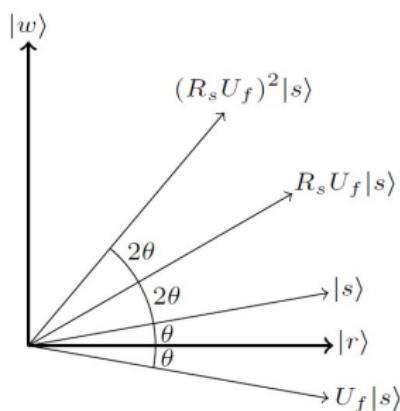
Beide Spiegelungen zusammen entsprechen in der rw -Ebene einer Drehung um 2θ .



Algorithmus von Grover

Erneute Anwendung von U_f und R_s führt zu einer weiteren Rotation um 2θ .

Wir fahren auf diese Weise mit Rotationen um 2θ fort, indem wir U_f gefolgt von R_s ausführen. Dadurch nähern wir uns dem gesuchten $|\omega\rangle$ weiter an.



Algorithmus von Grover

Angenommen es wurden insgesamt t Rotationen ausgeführt, um in der Nähe von $|\omega\rangle$ zu landen. Der Winkel zwischen $|r\rangle$ und $|\omega\rangle$ beträgt 90° bzw. $\frac{\pi}{2}$, also ist $\theta + t \cdot 2\theta = \frac{\pi}{2}$ bzw.

$$t = \frac{\pi}{4\theta} - \frac{1}{2}.$$

Für genügend großes N liefert eine Approximation

$$\theta = \sin^{-1} \left(\frac{1}{\sqrt{N}} \right) \approx \frac{1}{\sqrt{N}}$$

also

$$t \approx \frac{\pi}{4} \sqrt{N} - \frac{1}{2} \approx \frac{\pi}{4} \sqrt{N}.$$

Algorithmus von Grover

Die Anzahl der Anfragen an f ist also $O(\sqrt{N})$, was eine quadratische Beschleunigung gegenüber dem klassischen Fall mit $O(N)$ darstellt.

Das wir den Zustand $|\omega\rangle$ mit unserer Annäherung leicht verfehlten können, fällt bei obiger Abschätzung nicht ins Gewicht.

Algorithmus von Grover

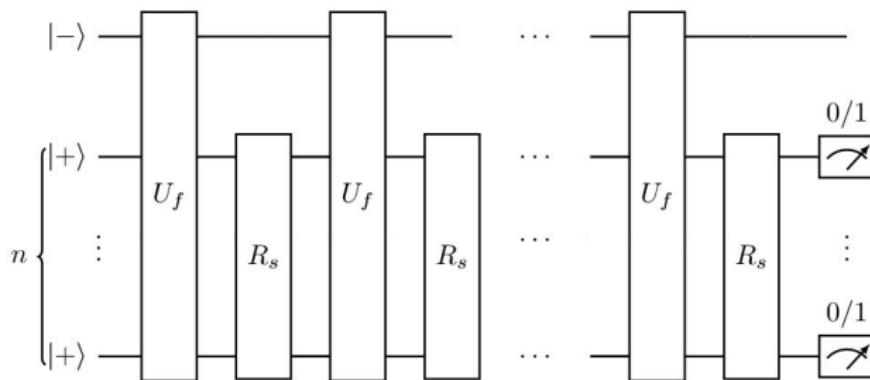
Algorithmus von Grovers

Gegeben: Eine Menge $\{0, \dots, N - 1\}$, $N = 2^n$, und ein Orakel U_f zur Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$ mit $f(\omega) = 1$ für gesuchtes ω und $f(x) = 0$ für $x \neq \omega$.

1. $|s\rangle \leftarrow H^{\otimes n}|0\rangle^{\otimes n}$
2. Wende t -mal die Grover-Iteration (Rotationen um 2θ , d.h. U_f gefolgt von R_s) auf $|s\rangle$ an
3. Messe $|s\rangle$ und gib das Ergebnis aus.

Algorithmus von Grover

Als Schaltkreis (inkl. target Qubit $|-\rangle$) erhalten wir



Algorithmus von Grover

Die Konstruktion von R_s hatten wir zunächst aufgeschoben. Diese soll $|s\rangle$ unverändert lassen und jeden Zustand senkrecht zu $|s\rangle$ spiegeln. Entsprechend ist

$$R_s = 2|s\rangle\langle s| - I,$$

denn

$$\underbrace{R_s |s\rangle}_{\text{Skalarprodukt}} = 2|s\rangle\langle s|s\rangle - |s\rangle = 2|s\rangle - |s\rangle = |s\rangle.$$

Für Zustände $|s^\perp\rangle$ orthogonal zu $|s\rangle$ hingegen ist $\langle s|s^\perp\rangle = 0$ und

$$R_s |s^\perp\rangle = 2|s\rangle\langle s|s^\perp\rangle - |s^\perp\rangle = -|s^\perp\rangle.$$

Algorithmus von Grover

Nun ist R_s noch durch elementare Gatter zu beschreiben (was gleichzeitig zeigt das R_s überhaupt ein zulässigen Quantengatter ist). Mit

$$|s\rangle = |+\rangle^{\otimes n} = H^{\otimes n}|0\rangle^{\otimes n} = H|0\rangle \dots H|0\rangle$$

ist wegen $H^\dagger = H$

$$\langle s| = \langle 0|H^\dagger \dots \langle 0|H^\dagger = \langle 0|H \dots \langle 0|H = \langle 0|^{\otimes n}H^{\otimes n}.$$

Einsetzen in $R_s = 2|s\rangle\langle s| - I$ ergibt

$$R_s = 2H^{\otimes n}|0^n\rangle\langle 0^n|H^{\otimes n} - I.$$



Algorithmus von Grover

Da H selbstinvers ist, können wir die Einheitsmatrix auch durch Hadamard-Transformation beschreiben:

$$\begin{aligned}I &= I \otimes \dots \otimes I = HH \otimes \dots \otimes HH = (H \otimes \dots \otimes H)(H \otimes \dots \otimes H) \\&= H^{\otimes n} H^{\otimes n}\end{aligned}$$

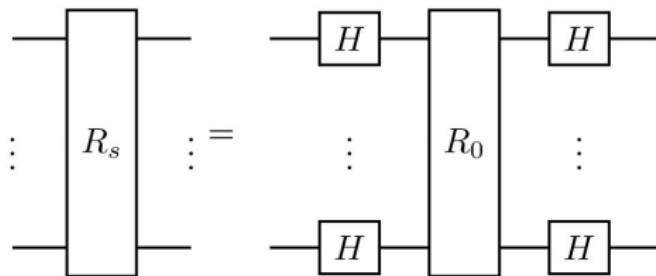
und es folgt

$$\begin{aligned}R_s &= 2H^{\otimes n}|0^n\rangle\langle 0^n|H^{\otimes n} - H^{\otimes n}H^{\otimes n} = H^{\otimes n}(2|0^n\rangle\langle 0^n| - I)H^{\otimes n} \\&= H^{\otimes n} R_0 H^{\otimes n}\end{aligned}$$

für $R_0 := 2|0^n\rangle\langle 0^n| - I$.

Algorithmus von Grover

Letzte Gleichung besagt das wir R_s durch R_0 beschreiben können, wenn wir auf R_0 beidseitig Hadamard-Transformation anwenden:



Algorithmus von Grover

Nun ist noch ein Schaltkreis für R_0 zu ermitteln. Die Wirkung von R_0 auf $|0^n\rangle$ ist

$$R_0 |0^n\rangle = 2|0^n\rangle\langle 0^n|0^n\rangle - |0^n\rangle = |0^n\rangle$$

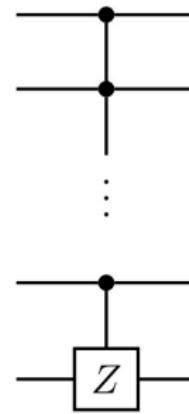
und auf einem bel. anderen Zustand $|a\rangle$ ist

$$R_0 |a\rangle = 2|0^n\rangle\langle 0^n|a\rangle - |a\rangle = -|a\rangle.$$

R_0 ist also eine Spiegelung an $|0^n\rangle$.

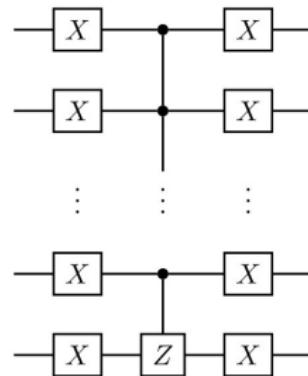
Algorithmus von Grover

Der abgebildete Schaltkreis führt zu einem Vorzeichenwechsel des Zustands $|1\dots1\rangle$.



Algorithmus von Grover

Wird auf beiden Seiten ein X-Gatter angewendet, dann führt der Schaltkreis zu einem Vorzeichenwechsel des Zustands $|0\dots0\rangle$.

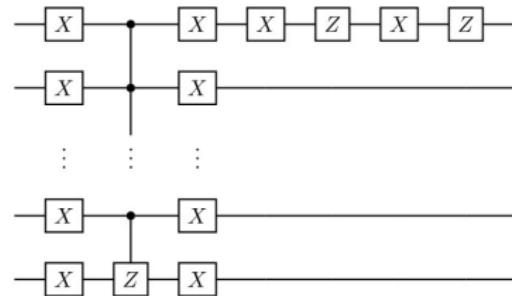


Algorithmus von Grover

Wir benötigen jedoch das der Zustand $|0\dots0\rangle$ unverändert bleibt, während alle anderen Zustände ein negatives Vorzeichen erhalten sollen.
Mittels

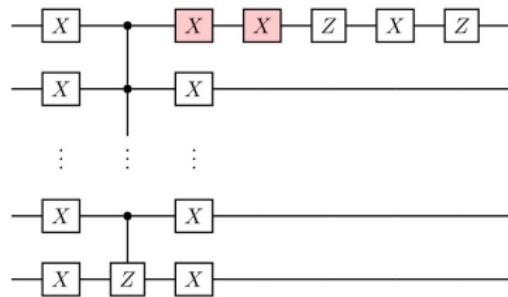
$$\begin{aligned} XZXZ (\alpha|0\rangle + \beta|1\rangle) \\ = -(\alpha|0\rangle + \beta|1\rangle) \\ = ZXZX (\alpha|0\rangle + \beta|1\rangle) \end{aligned}$$

können wir diesen Vorzeichenwechsel erreichen.



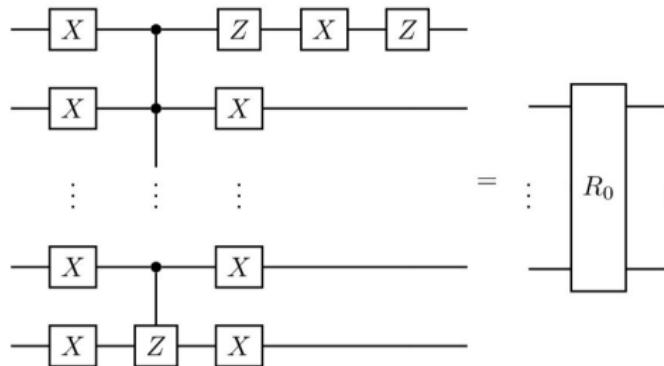
Algorithmus von Grover

An dieser Stelle wird nun direkt hintereinander das X-Gatter verwendet. Dieses ist selbstinvers.



Algorithmus von Grover

Damit erhalten wir gesuchtes R_0 als



Fügen wir um dieses noch Hadamard Gatter, erhalten wir R_s .

Algorithmus von Grover

Man kann zeigen das ein Quantencomputer brute-force Suchprobleme nicht schneller als $O(\sqrt{N})$ lösen kann. Der Algorithmus von Grover ist also optimal.

Benötigt ein klassischer Computer zur Lösung eines solchen Problems aus **NP** exponentiell viele Anfragen, dann benötigt auch ein Quantencomputer exponentiell viele Anfragen (nur mit niedrigerem Exponenten).

Das legt die Vermutung nahe, dass Quantencomputer **NP**-schwere Probleme nicht *effizient* lösen können.

Nichtsdestotrotz kann die Beschleunigung durch Quantencomputer für manche Praxisprobleme schon einen Unterschied machen.

Varianten der Quantensuche

Im Algorithmus von Grover wurde nach genau einem Element ω gesucht. Wird stattdessen nach k **Elementen** $\omega_1, \dots, \omega_k$ aus der Menge $\{0, \dots, N - 1\}$, $N = 2^n$ gesucht, dann wird $f : \{0, 1\}^n \rightarrow \{0, 1\}$ mit

$$f(x) = \begin{cases} 1 & \text{falls } x \in \{\omega_1, \dots, \omega_k\} \\ 0 & \text{sonst} \end{cases}$$

verwendet. Hierbei sollte $k \leq \frac{3}{4}N$ gelten.

Ist $k > \frac{3}{4}N$, dann wählt man zufällig ein Element $x \in \{0, \dots, N - 1\}$. Mit Wahrscheinlichkeit größer $3/4$ handelt es sich dabei schon um ein gesuchtes Element.

Varianten der Quantensuche

Für die Anzahl der Grover-Iterationen t gilt in diesem Fall

$$t \approx \frac{\pi}{4} \sqrt{\frac{N}{k}}$$

für $N \gg k$.

Bisher wurde stets vorausgesetzt, dass die Anzahl der gesuchten Elemente bekannt ist. Das ist nicht immer der Fall.

Die **G-BBHT Suche** ist eine Variante des Grover-Algorithmus (nach M. Boyer, G. Brassard, P. Hoyer, A. Tapp) bei der die Anzahl der Iterationen geraten wird.

Am Ende wird dann getestet, ob der Algorithmus das gesuchte Element geliefert hat.

Varianten der Quantensuche

G-BBHT Suche

Gegeben: Eine Menge $\{0, \dots, N - 1\}$, $N = 2^n$, und ein Orakel U_f zur Funktion $f : \{0, 1\}^n \rightarrow \{0, 1\}$ mit

$$f(x) = \begin{cases} 1 & \text{falls } x \in \{\omega_1, \dots, \omega_k\} \\ 0 & \text{sonst} \end{cases}$$

Die Zahl k der gesuchten Elemente ist *nicht* bekannt. Zu bestimmen ist eines der Elemente $\omega_1, \dots, \omega_k$.

1. Wähle zufällig ein $x' \in \{0, 1\}^n$. Falls $f(x') = 1$, wird x' ausgegeben und gestoppt.

Varianten der Quantensuche

G-BBHT Suche

2. Sei t ist die Anzahl der Iterationen. Wähle zufällig $t \in \{1, \dots, \sqrt{N}\}$.
3. Starte mit gleichverteilter Superposition $|s\rangle = H^{\otimes n}|0\rangle^{\otimes n}$.
4. Wende t -mal die Grover-Iteration auf $|s\rangle$ an.
5. Messe das Register und erhalte als Ergebnis der Messung x' . Werte f an x' aus. Ist $f(x') = 1$, dann gib x' aus und stoppe. Andernfalls beginne erneut mit Schritt 1.

Varianten der Quantensuche

Dabei ist zu bemerken:

- Ist $k \geq \frac{3}{4}N$, dann wird mit hoher Wahrscheinlichkeit bereits in Schritt 1 eines der gesuchten Elemente ausgewählt.
- Die zu erwartende Laufzeit ist $O(\sqrt{N})$, auch wenn die Anzahl der Iterationen geraten wird.

Es gibt noch weitere Varianten des Grover-Algorithmus, etwa

- für die Suche nach dem Minimum / Maximum in einer Datenbank,
- für das Zählen von markierten Elementen in einer Datenbank.

Varianten der Quantensuche

Wir halten abschließend fest:

Quadratische Beschleunigung

Jeder Quantenalgorithmus zur unstrukturierten Suche benötigt mindestens $\Omega(\sqrt{N})$ Orakelanfragen.

In diesem Sinne ist der Algorithmus von Grover optimal.

Inhaltsverzeichnis

1 Grundlagen

- Einleitung & Ein-Qubit Systeme
- n -Qubit Systeme & Grundlegende Schaltkreise
- Verschränkung, Deutsch-Jozsa, Bernstein-Vazirani & Simon

2 Quantensuche

- Algorithmus von Grover
- Varianten der Quantensuche

3 Fouriertransformation & Shor

- RSA & periodische Funktionen
- Diskrete- & Quanten-Fouriertransformation
- Algorithmus von Shor

4 Ausblick: QEC & QKD

RSA & periodische Funktionen

Wir wollen uns nun mit den Auswirkungen von Quantencomputing auf die Kryptographie befassen. Dabei ist zwischen symmetrischen (private key) und asymmetrischen (public key) Verfahren zu unterscheiden.

Während die Auswirkungen auf symmetrische Verfahren (bisher) überschaubar und im Wesentlichen nicht von besonderer Bedeutung sind (*Erhöhung der Schlüssellänge löst entstehende Probleme*), ist die Situation bei asymmetrischen Verfahren grundlegend verschieden.

Das führt zum (neuen) Fachgebiet *Post-Quanten Kryptographie*.

RSA & periodische Funktionen

Ausgangssituation: Alice und Bob möchten über einen nicht vertrauenswürdigen Kanal miteinander kommunizieren. Sie greifen deshalb auf Verschlüsselungsverfahren zurück. Dabei wird zunächst mit einem asymmetrischen Verfahren ein Schlüssel für ein symmetrisches Verfahren ausgetauscht. Die eigentliche Kommunikation erfolgt dann mit dem symmetrischen Verfahren.

Eve interessiert sich ebenfalls für den Inhalt der verschlüsselten Nachricht. Natürlich soll verhindert werden dass Eve an die Informationen gelangt.

RSA & periodische Funktionen

Public Key Kryptographie: Bob erwartet eine vertrauliche Nachricht.

1. Bob erzeugt zwei Schlüssel: einen geheimen, den er für sich behält, und einen öffentlichen, auf den Alice Zugriff hat.
2. Alice verschlüsselt die Nachricht an Bob mit dem öffentlichen Schlüssel (von Bob).
3. Bob verwendet seinen geheimen Schlüssel um die Nachricht von Alice zu entschlüsseln.

Ein Beispiel für ein public key Verfahren ist RSA. Wir wollen uns zunächst mit diesem befassen.

RSA & periodische Funktionen

Schlüsselerzeugung bei RSA

1. Wähle zwei Primzahlen $p \neq q$.
2. Berechne $n = pq$ und $\phi(n) = (p - 1)(q - 1)$.
3. Wähle eine kleine Zahl e mit $ggT(e, \phi(n)) = 1$.
4. Löse $ed \equiv 1 \pmod{(p - 1)(q - 1)}$.

Das Paar (e, n) ist der *öffentliche Schlüssel* und das Paar (d, n) der *geheime Schlüssel*.

Für jeden der oben genannten Schritte gibt es effiziente Algorithmen.

RSA & periodische Funktionen

Ver- und Entschlüsselung bei RSA

- Verschlüsselung einer Nachricht m : Löse $c \equiv m^e \pmod{n}$
- Entschlüsselung einer Nachricht c : Löse $m \equiv c^d \pmod{n}$

Beobachtung: Das Paar (e, n) ist als öffentlicher Schlüssel auch für Eve zugänglich; ebenso die übertragene Nachricht. Um den Klartext m zu bestimmen, benötigt Eve den geheimen Schlüssel d und muss dafür

$$e\color{red}{d} \equiv 1 \pmod{(p-1)(q-1)}$$

lösen. Dafür benötigt Eve das Produkt $(p-1)(q-1) = \phi(n)$. Sie kann versuchen $n = pq$ zu faktorisieren.

RSA & periodische Funktionen

Aber: Ist n groß genug gewählt, dann ist die Faktorisierung von n mit einem klassischen Computer (bisher) nicht effizient möglich.

Das klassisch beste Verfahren zur Faktorisierung ist das *Zahlkörpersieb* (vgl. *algebraische /algorithmische Zahlentheorie*) mit einer asymptotischen Laufzeit von

$$\exp \left((C + o(1)) (\log n)^{\frac{1}{2}} (\log \log n)^{\frac{2}{3}} \right)$$

wobei $C = \left(\frac{32}{9}\right)^{\frac{1}{3}}$ für das *spezielle Zahlkörpersieb* bzw. $C = \left(\frac{64}{9}\right)^{\frac{1}{3}}$ für das *allgemeine Zahlkörpersieb* gilt.

Die Laufzeit ist also subexponentiell, aber immer noch superpolynomial.

RSA & periodische Funktionen

Verfügt Eve jedoch über einen Quantencomputer (mit ausreichend vielen Qubits), dann kann Sie n effizient faktorisieren, d.h. das Faktorisierungsproblem liegt in **BQP**.

Wir lösen dies auf einem Quantencomputer mit dem *Algorithmus von Shor* (der eigentlich die Perioden von Funktionen bestimmt).

Achtung: Bei der Interpretation dieses Ergebnisses ist Vorsicht geboten. Das Faktorisierungsproblem liegt zwar in **BQP** und in **NP**, man kann aber (bisher) nicht beweisen ob es nicht doch auch in **P** liegt.

RSA & periodische Funktionen

Einige Fragen die man sich bei einer Einordnung stellen könnte, wären

- Ist der Algorithmus von Shor ein Hinweis darauf, dass das Faktorisierungsproblem eigentlich schon in P liegt und wir es bisher einfach nicht geschafft haben einen effizienten klassischen Algorithmus dafür zu finden?
- Ist der Algorithmus von Shor ein Hinweis darauf, dass Quantencomputer schwere Probleme effizient lösen können, sogar **NP**-vollständige Probleme?
Das ist vielleicht etwas viel verlangt und tatsächlich wird das von der Quantenkomplexitätstheorie-Community bezweifelt.
- Oder ...

RSA & periodische Funktionen

- ...
- Oder gilt beides vorherige nicht und Quantencomputer liefert eine Beschleunigung bei einer Art von *Zwischenproblemen* wie dem Faktorisierungsproblem, dass zwar klassisch nicht leicht zu lösen scheint, aber auch nicht NP-schwer ist?

Tatsächlich wissen wir einfach (noch) nicht welche dieser Möglichkeiten gilt.

RSA & periodische Funktionen

Zurück zu RSA. Wir rufen uns in Erinnerung

Definition (Periode einer Funktion)

Sei $f : \mathbb{R} \rightarrow \mathbb{R}$ eine Funktion und $p \in \mathbb{R}^+$ die kleinste Zahl, sodass $f(x + p) = f(x)$ für alle $x \in \mathbb{R}$ gilt. Dann heißt p die *Periode* von f .

Beispiel:

- $\sin(x + 2\pi) = \sin(x)$
- $f(x) = 2^x \pmod{5}$ hat die Periode $p = 4$, denn
 $f(0) = 1, f(1) = 2, f(2) = 4, f(3) = 3, f(4) = 1, \dots$

RSA & periodische Funktionen

Ziel: $n (= pq)$ aus dem RSA-Verfahren faktorisieren, d.h. einen echten Teiler $1 < a < n$ bestimmen.

O.B.d.A. kann n als ungerade angenommen werden (sonst sind 2 und $n/2$ echte Teiler).

Idee: Nutze den Euklidischen Algorithmus um zu prüfen ob a und n einen gemeinsamen Teiler haben. Der Algorithmus hat eine Laufzeit von $O((\log(n))^3)$, denn in unserem Fall ist $a < n$.

RSA & periodische Funktionen

Randomisierter Ansatz (so nicht)

- Wähle zufällig $a \in \{2, \dots, n - 1\}$ und bestimme einen Teiler durch Berechnung des $\text{ggT}(a, n)$.
- Dabei wird oft 1 als gemeinsamer Teiler auftreten, d.h. a und n sind teilerfremd. Denn $n = pq$ ist das Produkt großer Primzahlen p, q und

$$\begin{aligned}\phi(n) &= \#\{1 \leq a \leq n \mid \text{ggT}(a, n) = 1\} \\ &= (p - 1)(q - 1)\end{aligned}$$

d.h. die Wahrscheinlichkeit zufällig eine teilerfremde Zahl auszuwählen ist mit $\frac{(p-1)(q-1)}{pq}$ nahe bei 1 und *fast alle* Zahlen $\{2, \dots, n - 1\}$ sind teilerfremd zu n .

RSA & periodische Funktionen

Angenommen wir verfügen über folgende

Basisfähigkeit: Für $1 < a < n$ können wir die Periode p von $f(x) = a^x \pmod{n}$ bestimmen.

Da p die Periode ist, folgt

$$\begin{aligned}f(x + p) &= f(x) \Rightarrow f(p) = f(0) \\&\Rightarrow a^p \equiv 1 \pmod{n} \\&\Rightarrow a^p = 1 + kn \quad (k \in \mathbb{Z}).\end{aligned}$$

Also ist n ein Teiler von $a^p - 1$.

RSA & periodische Funktionen

Nehmen wir weiter an das die Periode p gerade ist, folgt

$$a^p - 1 = (a^{p/2} - 1)(a^{p/2} + 1) = kn.$$

Daraus folgt $\text{ggT}(a^{p/2} - 1, n) > 1$ oder $\text{ggT}(a^{p/2} + 1, n) > 1$.

Denn: Angenommen $\text{ggT}(a^{p/2} - 1, n) = \text{ggT}(a^{p/2} + 1, n) = 1$.

Dann sind wegen der Eindeutigkeit der Primfaktorzerlegung die Faktoren $a^{p/2} \pm 1$ beide Teiler von k , also

$$(a^{p/2} - 1)(a^{p/2} + 1) = kn = k'(a^{p/2} - 1)(a^{p/2} + 1)n$$

für $k' \in \mathbb{Z}$. Insbesondere folgt $n = 1$, im Widerspruch zu $n > 1$.

RSA & periodische Funktionen

Also gilt $\text{ggT}(a^{p/2} - 1, n) > 1$ oder $\text{ggT}(a^{p/2} + 1, n) > 1$.

Das beinhaltet auch die Möglichkeit das $a^{p/2} + 1$ ein Vielfaches von n ist. Dann ist $\text{ggT}(a^{p/2} + 1, n) = n$ und wir erhalten keine Information.

Übung: Zeigen Sie, dass $a^{p/2} - 1$ kein Vielfaches von n ist.

Damit ist $\text{ggT}(a^{p/2} - 1, n)$ oder $\text{ggT}(a^{p/2} + 1, n)$ ein echter Teiler von n , außer

- $\text{ggT}(a^{p/2} - 1, n) = 1$, und
- $a^{p/2} + 1$ ist ein Vielfaches von n .

RSA & periodische Funktionen

Der Fall das $a^{p/2} + 1$ ist ein Vielfaches von n , ist jedoch unwahrscheinlich.

Proposition

Sei n ungerade und keine Primzahlpotenz. Dann gilt für mindestens die Hälfte der Fälle der Zahlen $0 \leq a \leq n - 1$ mit $\text{ggT}(a, n) = 1$

1. die Periode p der Funktion $f(x) = a^x \pmod{n}$ ist gerade,
2. n teilt nicht $a^{p/2} + 1$, d.h. $a^{p/2} + 1$ ist kein Vielfaches von n .

Unsere Überlegungen können wir als Algorithmus festhalten. Dabei haben wir (bisher) noch keinen Quantencomputer eingesetzt.

RSA & periodische Funktionen

Faktorisierungsalgorithmus - klassischer Teil

Eingabe: Eine ungerade ganze Zahl n , die keine Primzahlpotenz ist.

Ausgabe: Ein echter Teiler von n .

1. Wähle zufällig eine Zahl $a \in \{2, \dots, n - 1\}$.
2. $z \leftarrow ggT(a, n)$.
Falls $z > 1$: Ausgabe von z . Abbruch
3. Ermittle die Periode p von $a^x \pmod{n}$.
4. Falls p ungerade ist: Beginne erneut mit Schritt 1.
5. Ermittle $ggT(a^{p/2} - 1, n)$ und $ggT(a^{p/2} + 1, n)$. Hat sich kein echter Teiler ergeben, beginne erneut mit Schritt 1.
Sonst: Ausgabe z .

RSA & periodische Funktionen

Schritt 3 des Algorithmus werden wir (später) mit einem Quantenalgorithmen lösen, was uns zum Algorithmus von Shor führt.

Fehlerwahrscheinlichkeit

Ist n ungerade und nicht die Potenz einer Primzahl, d.h. $n \neq p^k$ für $p \in \mathbb{P}$ und $k \geq 1$. Dann führt der klassische Teil des Algorithmus mit einer Wahrscheinlichkeit größer $1/2$ im ersten Anlauf zum Erfolg.

Die Wahrscheinlichkeit, nach k Versuchen noch keine Lösung gefunden zu haben, ist demnach kleiner $1/2^k$.

RSA & periodische Funktionen

Laufzeitbetrachtung:

- Schritt 1: zufällige Auswahl: konstante Laufzeit $\rightsquigarrow O(1)$
- Schritt 2: Euklidischer Algorithmus $\rightsquigarrow O((\log n)^3)$
- Schritt 3: Periodenbestimmung: unbekannte Laufzeit $T(n)$
- Schritt 4: prüfe gerade, ungerade: konstante Laufzeit $\rightsquigarrow O(1)$
- Schritt 5: Euklidischer Algorithmus $\rightsquigarrow O((\log n)^3)$

RSA & periodische Funktionen

Wir wissen bereits, dass die Wahrscheinlichkeit nach k Versuchen keine Lösung gefunden zu haben, kleiner als $1/2^k$ ist. Die erwartete Laufzeit ist also

$$\sum_{k \geq 1} \frac{k}{2^k} \cdot O((\log n)^3) = O((\log n)^3).$$

Übung: Zeigen Sie die Konvergenz der Reihe $\sum_{k \geq 1} \frac{k}{2^k}$.

RSA & periodische Funktionen

Gesamlaufzeit

Sei n ungerade und nicht die Potenz einer Primzahl, d.h. $n \neq p^k$ für $p \in \mathbb{P}$ und $k \geq 1$. Ein echter Teiler von n kann mit einer Laufzeit von

$$O((\log n)^3) + T(n)$$

bestimmt werden, wenn $T(n)$ die Laufzeit zur Bestimmung einer Periode der Funktion $a^x \pmod{n}$ für gegebenes $a \in \{2, \dots, n-1\}$ bezeichnet.

Frage: Wie bestimmt man die Periode?

Diskrete Fouriertransformation

Bevor wir auf die QFT zu sprechen kommen, wollen wir uns die DFT in Erinnerung rufen. Eine Variante der DFT, die sogenannte *schnelle Fouriertransformation*, findet vielfältige Anwendung in der Signalverarbeitung und Datenkompression.

Die Wirkungsweise der DFT ist für uns an dieser Stelle interessant, weil sie uns die Wirkungsweise der QFT besser verstehen lässt.

Wir diskutieren dies im Kontext von Polynomen (bleiben also weiter im Kontext von Vektorräumen).

Diskrete Fouriertransformation

Zur Erinnerung (LA): Die Menge der Polynome $K[X]$ über einem Körper K bildet mit der üblichen Addition und Multiplikation von Körperelementen einen unendlichdimensionalen Vektorraum. Die Basis bildet die Menge der Monome $\{1, x, x^2, x^3, \dots\}$.

Die Menge der Polynome deren Grad durch $n - 1$ beschränkt ist, bilden einen endlichdimensionalen Unterraum vom Grad n .

Wir stellen uns die Frage, wie wir Polynome effizient multiplizieren können. Die *Schulmethode* kennen wir bereits.

Diskrete Fouriertransformation

Proposition: Multiplikation („Schulmethode“)

Seien $A(x) = \sum_{i=0}^{n-1} a_i x^i$, $B(x) = \sum_{i=0}^{n-1} b_i x^i \in \mathbb{C}[x]$ Polynome.

Dann gilt für das Produkt der Polynome

$A(x) \cdot B(x) = C(x) = \sum_{i=0}^{2n-2} c_i x^i$, wobei $c_i = \sum_{j=0}^i a_j b_{i-j}$.

Die Berechnung benötigt eine Laufzeit von $O(n^2)$.

Wir wählen nun eine andere Darstellung für die Polynome. Bekannt ist uns etwa schon die Koeffizientendarstellung:

Definition: Koeffizientendarstellung

Sei $A(x) = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{C}[x]$ ein Polynom vom Grad $n - 1$. Dann heißt $(a_0, \dots, a_{n-1}) \in \mathbb{C}^n$ die *Koeffizientendarstellung* von A.

Diskrete Fouriertransformation

Bei der Wahl einer anderen Darstellung (in unserem Fall die *Stützstellendarstellung*) stellt sich zwangsläufig die Frage nach der Eindeutigkeit:

Proposition (Stützstellendarstellung, Interpolationspolynom)

Seien $(x_0, y_0), \dots, (x_{n-1}, y_{n-1}) \in \mathbb{C}^2$ n paarweise verschiedene Punkte. Dann gibt es genau ein Polynom A vom Grad $n - 1$ mit

$$A(x_i) = y_i, \quad 1 \leq i \leq n - 1.$$

Die Punkte (x_i, y_i) heißen *Stützstellen* von A und die Punkte $(x_0, y_0), \dots, (x_{n-1}, y_{n-1})$ die *Stützstellendarstellung* von A . Das Polynom A nennt man *Interpolationspolynom*.

Diskrete Fouriertransformation

Das lässt sich auch mit der *Vandermonde Matrix* $V(x)$ beschreiben:

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix}}_{=V(x)} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Die Vandermonde Matrix ist invertierbar, d.h. die Koeffizientendarstellung $\mathbf{a} = (a_0, \dots, a_{n-1})^T$ lässt sich eindeutig aus $\mathbf{y} = (y_0, \dots, y_{n-1})^T$ berechnen:

$$\mathbf{a} = V(x)^{-1} \mathbf{y}.$$

Diskrete Fouriertransformation

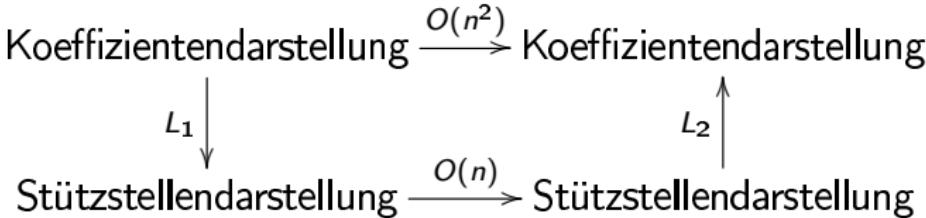
Für uns sind die folgende Beobachtungen interessant:

- Ist $(x_0, A(x_0))$ Stützstelle von A und $(x_0, B(x_0))$ Stützstelle von B , dann ist $(x_0, A(x_0)B(x_0))$ Stützstelle des Produktes C , wobei $C(x) = A(x)B(x)$.
- Sind A und B jeweils vom Grad $n - 1$, dann hat C den Grad $2n - 2$; d.h. wir benötigen $2n - 2$ Stützstellen.
- Die Berechnung der Stützstellen dieses Produktes benötigt eine Laufzeit von $O(n)$.

Diskrete Fouriertransformation

Zusammengefasst ergibt sich:

- Schulmultiplikation \rightsquigarrow Laufzeit $O(n^2)$
- punktweise Multiplikation mittels Stützstellen \rightsquigarrow Laufzeit $O(n)$
- Laufzeiten L_1 (punktweise Auswertung) und L_2 (Interpolation) sind uns (noch) unbekannt
- **Ziel:** L_1, L_2 in Laufzeit $O(n \log n)$



Diskrete Fouriertransformation

Für den Wechsel zwischen Koeffizientendarstellung und Stützstellendarstellung verwenden wir komplexe Einheitswurzeln:

Definition (Einheitswurzel)

Die (komplexen) Lösungen der Gleichung $x^n = 1$ heißen *n-te Einheitswurzeln*.

Es gibt genau n Einheitswurzeln. Diese sind die Potenzen von

$$\omega_n = \exp\left(\frac{2\pi i}{n}\right),$$

also $1 = \omega_n^0, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}$, wobei $\omega_n^n = \exp(2\pi i) = 1$.

Diskrete Fouriertransformation

Polynome lassen sich an Einheitswurzeln leicht auswerten. Auch Zwischenergebnisse solcher Auswertungen können weiter verwendet werden (Verwendung der Ergebnisse zu ω_n für die Auswertung an der Stelle ω_n^2).

Definition (diskrete Fouriertransformation, DFT)

Sei A ein Polynom vom Grad $n - 1$, gegeben in Koeffizientendarstellung $\mathbf{a} = (a_0, \dots, a_{n-1})^T$. Dann heißt $\mathbf{y} = (y_0, \dots, y_{n-1})^T$ mit

$$y_k = A(\omega_n^k), \quad 0 \leq k \leq n - 1$$

diskrete Fouriertransformation von \mathbf{a} . Bezeichnung: $\mathbf{y} = DFT_n(\mathbf{a})$.

Diskrete Fouriertransformation

Die DFT eines Polynoms A vom Grad $n - 1$ ist also dessen Stützstellendarstellung bzgl. der n -ten Einheitswurzel.

Die zu ω_n^k gehörende Komponente ist gegeben durch

$$y_k = \sum_{j=0}^{n-1} a_j \omega_n^{kj},$$

bzw.

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Diskrete Fouriertransformation

Die angegebene Matrix ist invertierbar, also ist auch DFT_n invertierbar und in der entsprechenden Matrix zu DFT_n^{-1} ist $\frac{1}{n}\omega_n^{-(j-1)(k-1)}$ der Eintrag an Stelle (j, k) .

Wir betrachten die Einheitswurzeln genauer und beobachten:

Ist $n > 1$ gerade, dann sind die Quadrate n -ter Einheitswurzeln selbst wieder $n/2$ -te Einheitswurzeln. Also

$$\left\{1, (\omega_n)^2, (\omega_n^2)^2, \dots, (\omega_n^{n-1})^2\right\} = \left\{1, \omega_{\frac{n}{2}}, \omega_{\frac{n}{2}}^2, \dots, \omega_{\frac{n}{2}}^{\frac{n}{2}-1}\right\},$$

denn

$$(\omega_n^k)^2 = \exp\left(\frac{2\pi i}{n} \cdot 2k\right) = \exp\left(\frac{2\pi i}{n/2} \cdot k\right) = \omega_{\frac{n}{2}}^k \quad \forall k \in \mathbb{N}.$$

Diskrete Fouriertransformation

Übung Zeigen Sie $\{1, \omega_4^2, (\omega_4^2)^2, (\omega_4^3)^2\} = \{1, \omega_2\}$.

Es treten also Wiederholungen auf.

Genauer: Durch quadrieren der n -ten Einheitswurzeln ergibt sich jede $n/2$ -te Einheitswurzel genau zweimal.

Insbesondere halbiert sich die Anzahl der Einheitswurzeln, die man berechnen muss!

Diskrete Fouriertransformation

Es folgt

$$\begin{aligned}A(x) &= a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1} \\&= (a_0 + a_2x^2 + a_4x^4 + \dots + a_{n-2}x^{n-2}) + \\&\quad + x(a_1 + a_3x^2 + a_5x^4 + \dots + a_{n-1}x^{n-2}) \\&= A_0(x^2) + xA_1(x^2)\end{aligned}$$

für

$$\begin{aligned}A_0(x) &:= a_0 + a_2x + a_4x^2 + \dots + a_{n-2}x^{\frac{n}{2}-1} \\A_1(x) &:= a_1 + a_3x + a_5x^2 + \dots + a_{n-1}x^{\frac{n}{2}-1}.\end{aligned}$$

Schnelle Fouriertransformation

Das Polynom A kann also an $\omega_n^0, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}$ ausgewertet werden, indem man die Polynome A_0 und A_1 vom halben Grad an den Einheitswurzeln $(\omega_n^0)^2, \omega_n^2, (\omega_n^2)^2, \dots, (\omega_n^{n-1})^2$ auswertet.

Algorithmus FFT_n (Fast Fourier Transformation)

Eingabe: (a_0, \dots, a_{n-1}) , wobei $n = 2^k, k \in \mathbb{N}$

Ausgabe: $y = DFT_n(a_0, \dots, a_{n-1})$

1. Falls $n - 1 = 0$. Ausgabe a_0 .
2. $a^0 \leftarrow (a_0, a_2, \dots, a_{n-2}), \quad a^1 \leftarrow (a_1, a_3, \dots, a_{n-1})$
3. $y^0 \leftarrow DFT_{\frac{n}{2}}(a^0), \quad y^1 \leftarrow DFT_{\frac{n}{2}}(a^1)$
4. Setze y^0 und y^1 entsprechend $A(x) = A_0(x^2) + xA^1(x^2)$ zu $y = DFT_n(a)$ zusammen.

Schnelle Fouriertransformation

Bezeichnet $T(n)$ die Laufzeit von FFT_n , so ergibt sich die Rekursion

$$T(1) = O(1)$$

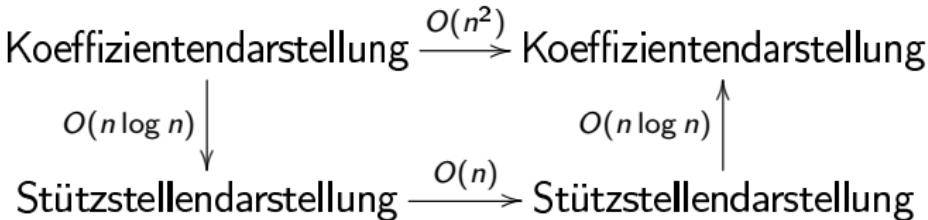
$$T(n) = 2T(n/2) + O(n)$$

Lösen der Rekursionsgleichung liefert $T(n) = O(n \log n)$.

Schnelle Fouriertransformation

Zusammen:

- Schulmultiplikation \rightsquigarrow Laufzeit $O(n^2)$
- punktweise Multiplikation mittels Stützstellen \rightsquigarrow Laufzeit $O(n)$
- Auswertung mittels FFT_n \rightsquigarrow Laufzeit $O(n \log n)$
- Interpolation mittels FFT_n^{-1} \rightsquigarrow Laufzeit $O(n \log n)$



Quanten-Fouriertransformation

Wie sich herausstellt ist die in der DFT verwendete Matrix

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix}$$

unter geeigneter Normierung unitär, also ein zulässiges Quantengatter.

Da sich die DFT für Anwendungen auf klassischen Computern (siehe Signalverarbeitung oder Datenkompression) als erfolgreich erwiesen hat, stellt sich natürlich die Frage wie es um eine *Quanten-Fouriertransformation* bestellt ist.

Quanten-Fouriertransformation

Definition (Quanten-Fouriertransformation - Teil I)

Die *Quanten-Fouriertransformation* der Ordnung N ist durch die unitäre Matrix

$$QFT_N := \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)(N-1)} \end{pmatrix}$$

definiert, wobei ω_N die N -te Einheitswurzel $\exp\left(\frac{2\pi i}{N}\right)$ bezeichnet.

Da es sich dabei tatsächlich um eine unitäre Matrix handelt, werden Sie auf dem Übungsblatt nachweisen.

Quanten-Fouriertransformation

Definition (Quanten-Fouriertransformation - Teil II)

Ist $|0\rangle, |1\rangle, \dots, |N-1\rangle$ eine Orthonormalbasis, dann operiert QFT_N auf dem N -dimensionalen Basisvektor $|j\rangle$ vermöge

$$QFT_N|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i j k}{N}\right) |k\rangle.$$

Die Quanten-Fouriertransformation führt demnach eine Transformation der Basis durch.

Quanten-Fouriertransformation

Für einen beliebigen Zustandsvektor $|v\rangle = \sum_{j=0}^{N-1} \alpha_j |j\rangle$ gilt also

$$\begin{aligned} QFT_N |v\rangle &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left(\sum_{j=0}^{N-1} \alpha_j \omega_N^{jk} \right) |k\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left(\sum_{j=0}^{N-1} \alpha_j \exp \left(\frac{2\pi i j k}{N} \right) \right) |k\rangle. \end{aligned}$$

Perspektivisch wird die *QFT* für uns ein einziger großer Quantenschaltkreis sein der auf n Qubits operiert, von denen ein Zustand $N = 2^n$ Amplituden hat.

Quanten-Fouriertransformation

Zunächst müssen wir *QFT* aber mittels Ein-Qubit und 2-Qubit Gattern implementieren.

Dafür gibt es eine geschickte Möglichkeit, die nur $O((\log N)^2)$ Ein-Qubit und 2-Qubit Gatter benötigt. Im Sinne der Schaltkreiskomplexität ist das eine exponentielle Beschleunigung gegenüber *FFT*, die Zeit $O(N \log N)$ benötigt.

Im Vergleich dazu sagt der Satz von Solovay-Kitaev das man für die Zerlegung eines n -Qubit Gatters in eine universelle Gattermenge zur Präzision ε etwa $\Theta(2^n \log^c(\frac{1}{\varepsilon}))$ Gatter benötigt.
Für $N = 2^n$ also $\Theta(N \log^c(\frac{1}{\varepsilon}))$.

Es lohnt sich also, die Konstruktion zu verstehen!

Quanten-Fouriertransformation

Wir schreiben zunächst j als n -Bit binäre Zahl

$$j = j_{n-1}j_{n-2}\dots j_1j_0 = j_{n-1}2^{n-1} + j_{n-2}2^{n-2} + \dots + j_12 + j_0$$

und erhalten so

$$\begin{aligned}\frac{j}{N} &= \frac{j_{n-1}2^{n-1} + j_{n-2}2^{n-2} + \dots + j_12 + j_0}{2^n} \\ &= \frac{j_{n-1}}{2} + \frac{j_{n-2}}{2^2} + \frac{j_1}{2^{n-1}} + \frac{j_0}{2^n} \\ &= 0.j_{n-1}j_{n-2}\dots j_1j_0\end{aligned}$$

Ebenso schreiben wir $k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \dots + k_12 + k_0$

Quanten-Fouriertransformation

Das liefert in der transformierten Amplitude

$$\begin{aligned} \exp\left(2\pi i \cdot \frac{j}{N} \cdot k\right) &= \\ &= \exp\left(2\pi i (0.j_{n-1}j_{n-2}\dots j_1j_0) (k_{n-1}2^{n-1} + \dots + k_12 + k_0)\right) \\ &= \exp\left(2\pi i (0.j_{n-1}j_{n-2}\dots j_1j_0) k_{n-1}2^{n-1}\right) \cdot \\ &\quad \cdot \exp\left(2\pi i (0.j_{n-1}j_{n-2}\dots j_1j_0) k_{n-2}2^{n-2}\right) \cdot \dots \cdot \\ &\quad \cdot \exp\left(2\pi i (0.j_{n-1}j_{n-2}\dots j_1j_0) k_12\right) \cdot \\ &\quad \cdot \exp\left(2\pi i (0.j_{n-1}j_{n-2}\dots j_1j_0) k_0\right) \end{aligned}$$

Die Multiplikation mit den **Zweierpotenzen** sorgt für eine Verschiebung der Nachkommastelle,

Quanten-Fouriertransformation

sodass wir in der Amplitude

$$\begin{aligned} \exp \left(2\pi i \cdot \frac{j}{N} \cdot k \right) &= \\ &= \exp (2\pi i (j_{n-1} j_{n-2} \dots j_1 \cdot j_0) k_{n-1}) \cdot \\ &\quad \cdot \exp (2\pi i (j_{n-1} j_{n-2} \dots j_2 \cdot j_1 j_0) k_{n-2}) \cdot \dots \cdot \\ &\quad \cdot \exp (2\pi i (j_{n-1} \cdot j_{n-2} \dots j_1 j_0) k_1) \cdot \\ &\quad \cdot \exp (2\pi i (0 \cdot j_{n-1} j_{n-2} \dots j_1 j_0) k_0) \end{aligned}$$

erhalten.

Quanten-Fouriertransformation

Nun verwenden wir noch das die komplexe Exponentialfunktion periodisch ist, d.h. $\exp(2\pi i \ell) = 1$ für alle $\ell \in \mathbb{Z}$, sodass

$$\begin{aligned}\exp\left(2\pi i \cdot \frac{j}{N} \cdot k\right) &= \\ &= \exp(2\pi i (0.j_0) k_{n-1}) \cdot \\ &\quad \cdot \exp(2\pi i (0.j_1 j_0) k_{n-2}) \cdot \dots \cdot \\ &\quad \cdot \exp(2\pi i (0.j_{n-2} \dots j_1 j_0) k_1) \cdot \\ &\quad \cdot \exp(2\pi i (0.j_{n-1} j_{n-2} \dots j_1 j_0) k_0)\end{aligned}$$

bleibt.

Quanten-Fouriertransformation

Einsetzen in die *QFT* eines Basisvektors ergibt

$$\begin{aligned} QFT_N |j\rangle &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i j k}{N}\right) |k\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp(2\pi i (0.j_0) k_{n-1}) \exp(2\pi i (0.j_1 j_0) k_{n-2}) \cdot \dots \\ &\quad \cdot \exp(2\pi i (0.j_{n-2} \dots j_1 j_0) k_1) \exp(2\pi i (0.j_{n-1} j_{n-2} \dots j_1 j_0) k_0) |k\rangle \end{aligned}$$

Quanten-Fouriertransformation

Da über alle n -Bit binären Zahlen k addiert wird, durchläuft jedes Bit $k_{n-1}, k_{n-2}, \dots, k_0$ die Summanden 0 und 1, sodass

$$\begin{aligned} QFT_N |j\rangle = & \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^1 \dots \sum_{k_0=0}^1 \exp(2\pi i (0.j_0) k_{n-1}) \cdot \\ & \cdot \exp(2\pi i (0.j_1 j_0) k_{n-2}) \cdot \dots \cdot \exp(2\pi i (0.j_{n-2} \dots j_1 j_0) k_1) \cdot \\ & \cdot \exp(2\pi i (0.j_{n-1} j_{n-2} \dots j_1 j_0) k_0) |k_{n-1} k_{n-2} \dots k_1 k_0\rangle \end{aligned}$$

Quanten-Fouriertransformation

Wir sortieren die Terme und zerlegen das Tensorprodukt

$$\begin{aligned}QFT_N|j\rangle = \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^1 \exp(2\pi i (0.j_0) k_{n-1}) |k_{n-1}\rangle \cdot \\ \cdot \sum_{k_{n-2}=0}^1 \exp(2\pi i (0.j_1 j_0) k_{n-2}) |k_{n-2}\rangle \cdot \dots \cdot \\ \cdot \sum_{k_1=0}^1 \exp(2\pi i (0.j_{n-2} \dots j_1 j_0) k_1) |k_1\rangle \cdot \\ \cdot \sum_{k_0=0}^1 \exp(2\pi i (0.j_{n-1} j_{n-2} \dots j_1 j_0) k_0) |k_0\rangle\end{aligned}$$

Quanten-Fouriertransformation

Wir schreiben die Summen aus und mittels $\exp(0) = 1$ bleibt für den jeweils ersten Summanden nur $|0\rangle$, d.h.

$$\begin{aligned} QFT_N |j\rangle = \frac{1}{\sqrt{N}} & (|0\rangle + \exp(2\pi i (0.j_0)) |1\rangle) \cdot \\ & \cdot (|0\rangle + \exp(2\pi i (0.j_1 j_0)) |1\rangle) \cdot \dots \\ & \cdot (|0\rangle + \exp(2\pi i (0.j_{n-2} \dots j_1 j_0)) |1\rangle) \cdot \\ & \cdot (|0\rangle + \exp(2\pi i (0.j_{n-1} j_{n-2} \dots j_1 j_0)) |1\rangle) \end{aligned}$$

Quanten-Fouriertransformation

Abschließend verwenden wir $\sqrt{N} = \sqrt{2^n}$ um jedem Faktor noch seine Normierung zuzuweisen, also

$$\begin{aligned}QFT_N|j\rangle &= \frac{1}{\sqrt{2}} (|0\rangle + \exp(2\pi i (0.j_0)) |1\rangle) \cdot \\&\quad \cdot \frac{1}{\sqrt{2}} (|0\rangle + \exp(2\pi i (0.j_1 j_0)) |1\rangle) \cdot \dots \cdot \\&\quad \cdot \frac{1}{\sqrt{2}} (|0\rangle + \exp(2\pi i (0.j_{n-2} \dots j_1 j_0)) |1\rangle) \cdot \\&\quad \cdot \frac{1}{\sqrt{2}} (|0\rangle + \exp(2\pi i (0.j_{n-1} j_{n-2} \dots j_1 j_0)) |1\rangle)\end{aligned}$$

Quanten-Fouriertransformation

Können wir einen Quantenschaltkreis konstruieren der $|j\rangle = |j_{n-1}...j_0\rangle$ in dieses (vorherige) Produkt überführt, dann haben wir damit einen Quantenschaltkreis für die *QFT*.

Wie wir feststellen werden, genügt uns dafür eine Kombination von Hadamard-Gattern und kontrollierten Rotationen.

Wir beginnen unsere Konstruktion mit dem im Produkt rechts stehenden Faktor

$$\frac{1}{\sqrt{2}} (|0\rangle + \exp(2\pi i (0.j_{n-1}j_{n-2}...j_1j_0)) |1\rangle)$$

Quanten-Fouriertransformation

Wenden wir das Hadamard-Gatter auf $|j_{n-1}\rangle$ an, erhalten wir

$$\begin{aligned} H|j_{n-1}\rangle &= \frac{1}{\sqrt{2}} (|0\rangle + (-1)^{j_{n-1}}|1\rangle) \\ &= \frac{1}{\sqrt{2}} (|0\rangle + (e^{i\pi})^{j_{n-1}}|1\rangle) \\ &= \frac{1}{\sqrt{2}} \left(|0\rangle + \left(\exp\left(2\pi i \cdot \frac{j_{n-1}}{2}\right) \right) |1\rangle \right) \\ &= \frac{1}{\sqrt{2}} (|0\rangle + (\exp(2\pi i(0.j_{n-1})))|1\rangle) \end{aligned}$$

Quanten-Fouriertransformation

Betrachten wir nun das Ein-Qubit Gatter R_r definiert durch

$$R_r |0\rangle = |0\rangle, \quad R_r |1\rangle = \exp\left(\frac{2\pi i}{2^r}\right) |1\rangle$$

bzw. als Matrix

$$R_r = \begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{2\pi i}{2^r}\right) \end{pmatrix}$$

R_r wird auch *Phasen-Rotation* genannt.

Nach Anwendung des Hadamard-Gatters wenden wir R_2 auf das $(n - 1)$ te Qubit an, kontrolliert durch das $(n - 2)$ te Qubit.

Quanten-Fouriertransformation

Im Zustand des $(n - 1)$ te Qubits wird dadurch die Amplitude von $|1\rangle$ mit $\exp\left(\frac{2\pi i}{2^r}\right)$ multipliziert, wenn $j_{n-2} = 1$ gilt:

$$\begin{aligned}& \frac{1}{\sqrt{2}} (|0\rangle + (\exp(2\pi i(0.j_{n-1}))) |1\rangle) \longmapsto \\& \longmapsto \frac{1}{\sqrt{2}} \left(|0\rangle + (\exp(2\pi i(0.j_{n-1}))) \exp\left(2\pi i \frac{1}{2^2}\right)^{j_{n-2}} |1\rangle \right) \\& = \frac{1}{\sqrt{2}} (|0\rangle + (\exp(2\pi i(0.j_{n-1}))) \exp(2\pi i(0.0j_{n-2})) |1\rangle) \\& = \frac{1}{\sqrt{2}} (|0\rangle + (\exp(2\pi i(0.j_{n-1}j_{n-2}))) |1\rangle)\end{aligned}$$

Quanten-Fouriertransformation

Analog liefert die Anwendung von R_3 auf das $(n - 1)$ te Qubit an, kontrolliert durch das $(n - 3)$ te Qubit

$$\frac{1}{\sqrt{2}} (|0\rangle + (\exp(2\pi i(0.j_{n-1}j_{n-2}j_{n-3}))) |1\rangle)$$

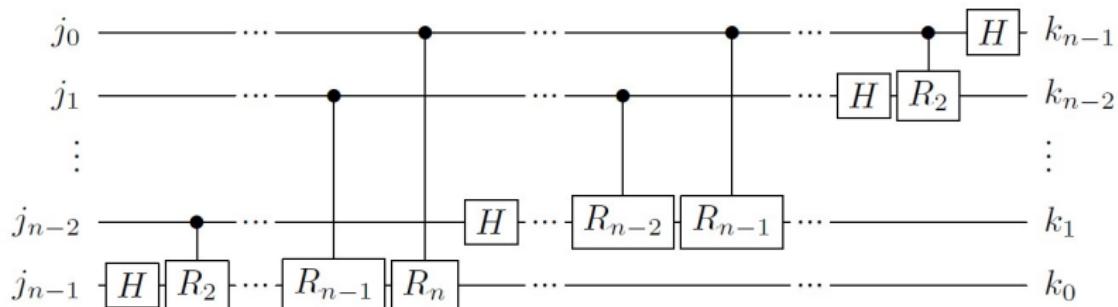
und Fortsetzung dieser Prozesses bis R_n , kontrolliert durch das 0-te Qubit, ergibt für das $(n - 1)$ te Qubit

$$\frac{1}{\sqrt{2}} (|0\rangle + (\exp(2\pi i(0.j_{n-1}j_{n-2}j_{n-3}\dots j_0))) |1\rangle)$$

Das ist gerade der gesuchte Faktor in der Darstellung von $QFT_N|j\rangle$. Analog ergeben sich die anderen Faktoren.

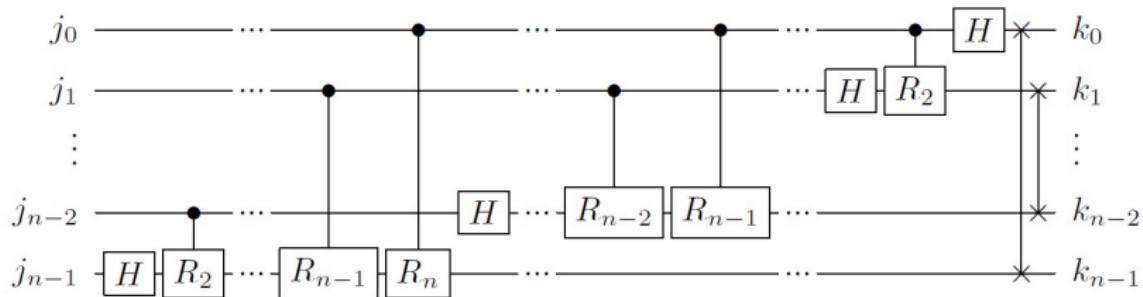
Quanten-Fouriertransformation

Auf Schaltungsebene ist dies



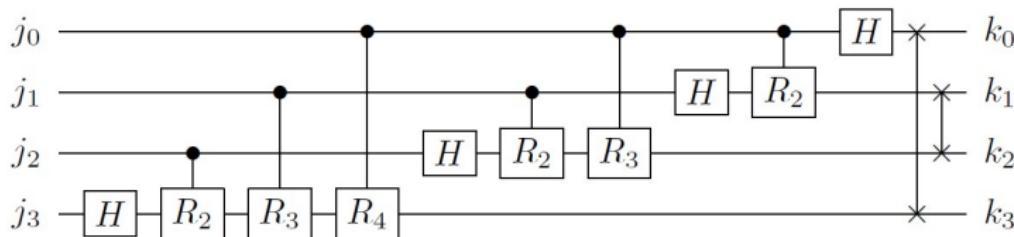
Quanten-Fouriertransformation

Dabei ist der Output noch in der falschen Reihenfolge. Wir korrigieren dies mittels SWAP-Gattern



Quanten-Fouriertransformation

Für $n = 4$ erhalten wir also die *QFT* als Schaltkreis



Quanten-Fouriertransformation

Wir müssen nun noch die Anzahl der benötigten Gatter für die *QFT* auf n Qubits bestimmen:

Auf unterster Ebene (j_{n-1} Ebene) werden n Gatter benötigt ($n - 1$ Rotationen und ein Hadamard-Gatter),
auf der darüber liegenden Ebene (j_{n-2} Ebene) werden $n - 1$ Gatter benötigt ($n - 2$ Rotationen und ein Hadamard-Gatter), usw.,
bis auf oberster Ebene (j_0 Ebene) nur noch ein Hadamard-Gatter verwendet wird.

Quanten-Fouriertransformation

Die Anzahl der verwendeten Hadamard-Gatter und kontrollierten R_r -Gatter ist also

$$n + (n - 1) + \dots + 2 + 1 = \frac{n(n - 1)}{2}.$$

Dazu kommen noch die SWAP-Gatter, von denen $\frac{n}{2}$ verwendet werden. Insgesamt ist die Anzahl der verwendeten Ein-Qubit und 2-Qubit Gatter

$$\frac{n(n - 1)}{2} + \frac{n}{2} = O(n^2) = O((\log N)^2)$$

für $N = 2^n$.

Quanten-Fouriertransformation

Die Laufzeit $O((\log N)^2)$ für die *QFT* ist eine exponentielle Beschleunigung gegenüber *FFT*, die Zeit $O(N \log N)$ benötigt.

Diese Beschleunigung gibt es aber *nicht umsonst*. Während wir bei Anwendung des klassischen Algorithmus alle Terme der *DFT* erhalten, liefert uns *QFT* einen Quantenzustand dessen Amplituden die für uns relevanten Informationen enthalten. Dieses Qubit können wir nur messen und damit einen teilweisen Zugriff auf die Information erhalten.

Für eine gewinnbringenden Einsatz der *QFT* bedarf es also ein wenig Geschick und einen dazu passenden Anwendungsfall.

Quanten-Fouriertransformation

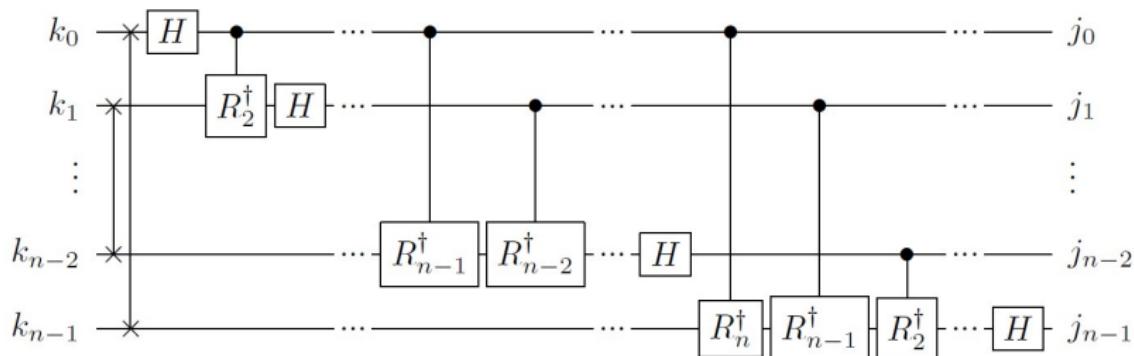
Die *inverse Quanten-Fouriertransformation* kehrt notwendigerweise die Wirkung der *QFT* um. Sie ist gegeben durch

$$QFT_N^{-1} \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp \left(2\pi i \cdot \frac{jk}{N} \right) |k\rangle \right) = |j\rangle.$$

Als Quantenschaltkreis erhält man die inverse *QFT* durch Umkehrung der Reihenfolge der Gatter und dem Austausch jedes Gatters durch sein jeweiliges Inverses.

Quanten-Fouriertransformation

Das SWAP-Gatter und das Hadamard-Gatter sind selbstinvers. In R_r ersetzt man $\exp\left(\frac{2\pi i}{2^r}\right)$ durch $\exp\left(-\frac{2\pi i}{2^r}\right)$ um R_r^\dagger zu erhalten.



Phasenabschätzung

Klassischer Ansatz: Aus der Problemstellung wissen wir bereits das $|v\rangle$ ein Eigenvektor von U ist und das Eigenwerte von der Form $\exp(i\theta)$ sind, also

$$U|v\rangle = \exp(i\theta)|v\rangle$$

Ist $|v\rangle$ eine N -dimensionaler Vektor und U eine $(N \times N)$ -Matrix, können wir dies als Gleichung schreiben:

$$\begin{pmatrix} U_{11} & U_{12} & \cdots & U_{1N} \\ U_{21} & U_{22} & \cdots & U_{2N} \\ \vdots & \vdots & & \vdots \\ U_{N1} & U_{N2} & \cdots & U_{NN} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} = \exp(i\theta) \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}$$

Phasenabschätzung

Tritt ein Eigenvektor in Form eines Quantenzustands auf, spricht man auch vom *Eigenzustand*. $|+\rangle$ und $|-\rangle$ sind also Eigenzustände des X-Gatters.

Problemstellung: Gegeben eine unitäre Matrix U und einen ihrer Eigenvektoren $|v\rangle$, bestimme den Eigenwert oder eine Abschätzung für den Eigenwert.

Wir wissen das die Eigenwerte einer unitären Matrix von der Form $\exp(i\theta)$ für reelles $0 \leq \theta < 2\pi$ sind.⁵ Die Bestimmung der Eigenwerte ist also äquivalent zur Bestimmung der Phase θ . Man spricht deshalb auch vom Problem der *Phasenabschätzung*.

⁵ vgl. Übungsaufgabe

Phasenabschätzung

Multiplikation von Matrix und Vektor ergibt

$$\begin{pmatrix} U_{11}v_1 + U_{12}v_2 + \dots + U_{1N}v_N \\ U_{21}v_1 + U_{22}v_2 + \dots + U_{2N}v_N \\ \vdots \\ U_{N1}v_1 + U_{N2}v_2 + \dots + U_{NN}v_N \end{pmatrix} = \exp(i\theta) \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}$$

Wir wählen eine beliebige Zeile zur Bestimmung von $\exp(i\theta)$, etwa die erste Zeile, und erhalten die Gleichung

$$U_{11}v_1 + U_{12}v_2 + \dots + U_{1N}v_N = \exp(i\theta)v_1$$

Phasenabschätzung

Dann ist der Eigenwert

$$\exp(i\theta) = \frac{U_{11}v_1 + U_{12}v_2 + \dots + U_{1N}v_N}{v_1}$$

Dafür werden N Multiplikationen, $N - 1$ Additionen und eine Division benötigt, insgesamt also

$$N + (N - 1) + 1 = 2N = O(N)$$

elementare Rechenoperationen.

Phasenabschätzung

Quantenalgorithmus: Angenommen die unitäre Matrix U ist ein Gatter auf n Qubits, also eine $(N \times N)$ -Matrix für $N = 2^n$. Wir nehmen weiter an der Eigenvektor $|v\rangle$ bestehe aus n Qubits und sein Eigenwert soll auf eine Genauigkeit von m Bits abgeschätzt werden. Dafür benötigen wir m zusätzliche (Hilfs-)Qubits die alle mit $|0\rangle$ initialisiert werden. Insgesamt

$$\underbrace{|0\dots0\rangle}_{m \text{ Qubits}} \otimes \underbrace{|v\rangle}_{n \text{ Qubits}}$$

Die Anzahl der Qubits in unserem Schaltkreis ist also $m + n$.

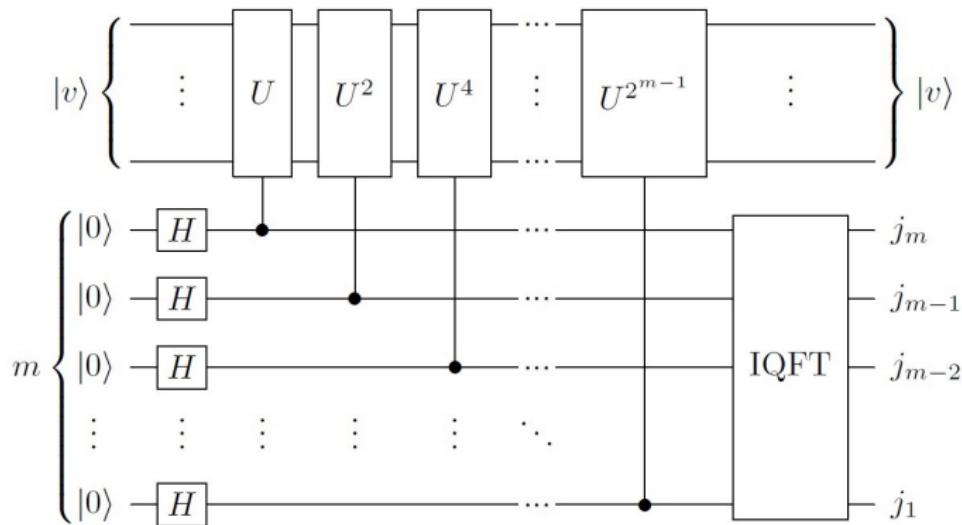
Phasenabschätzung

Die ersten m Qubits nennen wir auch *Eigenwertregister*, denn diese werden die m -Bit Approximation an die Phase enthalten.

Die verbleibenden n Qubits nennen wir *Eigenzustandregister*, denn diese enthalten den Eigenzustand $|v\rangle$.

Phasenabschätzung

Den Schaltkreis zur Abschätzung der Phase erhalten wir als



Phasenabschätzung

Analyse: In **Schritt 2** erhalten wir durch Anwendung von Hadamard Gattern

$$\begin{aligned} |\psi_2\rangle &= H^{\otimes m} |0\rangle |v\rangle \\ &= \frac{1}{\sqrt{2^m}} (|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle) |v\rangle \end{aligned}$$

Dann wird ein kontrollierten U -Gatter ausgeführt, wobei die Kontrolle durch das im Eigenwertregister ganz rechts stehende Qubit ausgeübt wird und das Zielqubit das Eigenzustandregister ist.

Wegen $U|v\rangle = \exp(i\theta)|v\rangle$ führt das dazu, dass der Zustand eine Phase $\exp(i\theta)$ erhält, wenn das Kontrollqubit $|1\rangle$ ist.

Phasenabschätzung

In **Schritt 3** ergibt sich somit

$$|\psi_3\rangle = \frac{1}{\sqrt{2^m}} (|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle) (|0\rangle + \exp(i\theta)|1\rangle) |\nu\rangle$$

Dann wird in **Schritt 4** ein kontrolliertes U^2 -Gatter ausgeführt, welches für das im Eigenwertregister an zweiter Stelle von rechts stehende Qubit zu einer Phase $\exp(2i\theta)$ führt, wenn das Kontrollqubit $|1\rangle$ ist:

$$\begin{aligned} |\psi_4\rangle = & \frac{1}{\sqrt{2^m}} (|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle) \cdot \\ & \cdot (|0\rangle + \exp(2i\theta)|1\rangle) (|0\rangle + \exp(i\theta)|1\rangle) |\nu\rangle \end{aligned}$$

Phasenabschätzung

In **Schritt 5** führt Anwendung eines kontrollierten U^4 -Gatters auf

$$|\psi_5\rangle = \frac{1}{\sqrt{2^m}} (|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle) \cdot \\ \cdot (|0\rangle + \exp(4i\theta)|1\rangle) (|0\rangle + \exp(2i\theta)|1\rangle) (|0\rangle + \exp(i\theta)|1\rangle) |v\rangle$$

Wir führen diesen Prozess fort und erhalten in **Schritt $2^{m-1} + 2$** (d.h. im vorletzten Schritt) durch Anwendung eines kontrollierten $U^{2^{m-1}}$ -Gatters

$$|\psi_{2^{m-1}+2}\rangle = \frac{1}{\sqrt{2^m}} (|0\rangle + \exp(2^{m-1}i\theta)|1\rangle) \dots (|0\rangle + \exp(4i\theta)|1\rangle) \cdot \\ \cdot (|0\rangle + \exp(2i\theta)|1\rangle) (|0\rangle + \exp(i\theta)|1\rangle) |v\rangle$$

Phasenabschätzung

Zur besseren Analyse führen wir eine Substitution der Variablen aus und setzen $\theta = 2\pi j$. Können wir j bestimmen, dann auch θ . Der durch Substitution erhaltene Zustand ist dann

$$\frac{1}{\sqrt{2^m}} (|0\rangle + \exp(2\pi i \cdot 2^{m-1}j)|1\rangle) \dots (|0\rangle + \exp(2\pi i \cdot 4j)|1\rangle) \cdot \\ \cdot (|0\rangle + \exp(2\pi i \cdot 2j)|1\rangle) (|0\rangle + \exp(2\pi i \cdot j)|1\rangle) |v\rangle$$

Wegen $0 \leq \theta < 2\pi$ ist auch $0 \leq j < 1$. Wir schreiben j als m -Bit binäre Zahl $0.j_1j_2\dots j_m$ (also eine Zahl kleiner 1), sodass

Phasenabschätzung

$$\frac{1}{\sqrt{2^m}} (|0\rangle + \exp(2\pi i(j_1 j_2 \dots j_{m-1} j_m)) |1\rangle) \dots \\ \cdot (|0\rangle + \exp(2\pi i(j_1 j_2 \dots j_m)) |1\rangle) \\ \cdot (|0\rangle + \exp(2\pi i(j_1 \dots j_m)) |1\rangle) \\ \cdot (|0\rangle + \exp(2\pi i(0 \dots j_m)) |1\rangle) |\nu\rangle$$

Mittels $\exp(2\pi i k) = 1$ für alle $k \in \mathbb{Z}$ ist die äquivalent zu

$$\frac{1}{\sqrt{2^m}} (|0\rangle + \exp(2\pi i(0 \dots j_m)) |1\rangle) \dots \\ \cdot (|0\rangle + \exp(2\pi i(0 \dots j_m)) |1\rangle) \\ \cdot (|0\rangle + \exp(2\pi i(0 \dots j_m)) |1\rangle) \\ \cdot (|0\rangle + \exp(2\pi i(0 \dots j_m)) |1\rangle) |\nu\rangle$$

Phasenabschätzung

Das ist gerade die *QFT* von $|j_1 j_2 \dots j_m\rangle$. Wir können also $|j_1 j_2 \dots j_m\rangle$ ermitteln, indem wir die inverse *QFT* auf das Eigenwertregister anwenden. Das führt im letzten Schritt auf

$$|j_1 j_2 \dots j_m\rangle |v\rangle$$

und vervollständigt die Analyse des Quantenschaltkreises für die Phasenabschätzung.

Die Mittels Messung erhaltenen j_1, j_2, \dots, j_m werden dann noch nachverarbeitet.

Phasenabschätzung

Zunächst ist

$$j = 0.j_1j_2\dots j_m = \frac{j_1}{2} + \frac{j_2}{4} + \dots + \frac{j_m}{2^m}$$

Die Phase des Eigenwertes ist entsprechend der Substitution $\theta = 2\pi j$ und der Eigenwert selbst ist $\exp(i\theta)$.

Um eine Abschätzung bis auf m -Bit Genauigkeit zu erhalten, wurden m Hadamard-Gatter, m kontrollierte U^P -Gatter und die inverse QFT mit $O(m^2)$ Gattern verwendet. Insgesamt also

$$m + m + O(m^2) = O(m^2)$$

Gatter.

Phasenabschätzung

Die Anzahl $O(m^2)$ für die Phasenabschätzung ist also bestimmt durch die Bitgenauigkeit m .

Im klassischen Fall wurden $O(N) = O(2^n)$ elementare Rechenoperationen verwendet.

In Abhängigkeit der Bitgenauigkeit m kann der Quantenalgorithmus also Vorteile liefern (auch wenn wir an dieser Stelle die Erzeugung von $|v\rangle$ und der kontrollierten U^P -Gatter erst einmal vernachlässigt haben).

Ordnungsbestimmung

Im Faktorisierungsverfahren (Schritt 3) zum RSA-Verfahren war die Bestimmung der Periode der Funktion

$$f(x) = a^x \pmod{N}$$

ein wesentlicher Bestandteil.

Die Berechnung von Potenzen mit anschließender Modulorechnung (Reduktion modulo N) bezeichnet man auch als *modulare Exponentiation*.

Ordnungsbestimmung

Wir haben bereits gesehen, dass dabei Wiederholungen auftreten können:

$$2^0 \pmod{7} = 1$$

$$2^1 \pmod{7} = 2$$

$$2^2 \pmod{7} = 4$$

$$2^3 \pmod{7} = 1$$

$$2^4 \pmod{7} = 2$$

$$2^5 \pmod{7} = 4, \dots$$

In diesem Fall ist die Periode $r = 3$.

Ordnungsbestimmung

Bei der modularen Exponentiation $a^x \pmod{N}$ treten immer dann Wiederholungen (Perioden) auf, wenn $\text{ggT}(a, N) = 1$ gilt (vgl. Elementare Zahlentheorie).

Die Periode r ist auch stets kleiner als N .

Da die sich wiederholende Folge immer mit 1 startet, kann die Periode auch als die kleinste positive Zahl r mit $a^r \pmod{N} = 1$ verstanden werden. Aus algebraischen Gründen nennt man r auch *Ordnung*.

Problemstellung: Bestimme die Periode bzw. Ordnung bei modularer Exponentiation.

Ordnungsbestimmung

Klassischer Ansatz: Für die modulare Exponentiation gibt es ein schnelles klassisches Verfahren, die sog. *schnelle Exponentiation* (oder auch *Square-and-Multiply*). Wir verdeutlichen das Vorgehen an einem Beispiel, in dem wir die $91^{43} \pmod{131}$ berechnen.

Zunächst schreiben wir den Exponenten als Binärzahl

$$\begin{aligned} 43 &= 101011_2 \\ &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2 + 1 \cdot 2^0 \\ &= 32 + 8 + 2 + 1 \end{aligned}$$

Also

$$91^{43} \pmod{131} = 91^{32} \cdot 91^8 \cdot 91^2 \cdot 91 \pmod{131}$$

Ordnungsbestimmung

Die auftretenden Potenzen können wir durch schrittweises Quadrieren berechnen:

$$91^2 \pmod{131} = 8281 \pmod{131} = 28$$

$$91^4 \pmod{131} = 28^2 \pmod{131} = 784 \pmod{131} = 129$$

$$91^8 \pmod{131} = 129^2 \pmod{131} = 16641 \pmod{131} = 4$$

$$91^{18} \pmod{131} = 4^2 \pmod{131} = 16$$

$$91^{32} \pmod{131} = 16^2 \pmod{131} = 256 \pmod{131} = 125$$

Ordnungsbestimmung

Dies setze wir in unsere vorherige Rechnung ein und erhalten

$$\begin{aligned}91^{43} \pmod{131} &= 91^{32} \cdot 91^8 \cdot 91^2 \cdot 91 \pmod{131} \\&= 125 \cdot 4 \cdot 28 \cdot 91 \pmod{131} \\&= 1274000 \pmod{131} \\&= 25\end{aligned}$$

Wir müssen nun noch feststellen wie viele Rechenschritte im Allgemeinen bei der Verwendung der schnellen Exponentiation notwendig sind.

Ordnungsbestimmung

Angenommen wir wollen $a^x \pmod{N}$ für eine n -Bit Zahl binäre Zahl x bestimmen.

Wir beginnen mit a und führen $n - 1$ Quadrierungen gefolgt von Reduktion modulo N durch. Die Ergebnisse werden multipliziert, wobei bis zu $n - 1$ Multiplikationen gefolgt von erneuter Reduktion modulo N durchgeführt werden. Insgesamt werden dabei

$$(n - 1) + (n - 1) = O(n)$$

Operationen modulo N ausgeführt.

Ordnungsbestimmung

Modulare Exponentiation mittels Square-and-Multiply kann schnell durchgeführt werden.

Dies überträgt sich jedoch für großes N nicht auf das Problem der Ordnungsbestimmung. Ist N groß, müssen viele einzelne modulare Exponentiation ausgeführt werden.

Es gibt (bisher) keinen effizienten klassischen Algorithmus für das Problem der Ordnungsbestimmung.

Ordnungsbestimmung

Quantenalgorithmus: Wollen wir $a^x \pmod{N}$ auf einem Quantencomputer berechnen, benötigen wir ein entsprechendes Gatter für die modulare Exponentiation. Für die modulare Multiplikation verwenden wir

$$U|y\rangle = |ay \pmod{N}\rangle$$

Da wir bei der Berechnung Reduktion modulo N durchführen, kann $0 \leq y < N$ angenommen werden.

Ordnungsbestimmung

Wiederholte Anwendung von U auf $|1\rangle$ führt dann auf

$$U^0|1\rangle = |a^0 \pmod{N}\rangle = |1 \pmod{N}\rangle$$

$$U|1\rangle = |a \pmod{N}\rangle$$

$$U^2|1\rangle = U|a \pmod{N}\rangle = |a^2 \pmod{N}\rangle$$

⋮

$$U^r|1\rangle = |a^r \pmod{N}\rangle = |a^0 \pmod{N}\rangle$$

wobei im letzten Schritt verwendet wurde das die Ordnung r ist.

Ordnungsbestimmung

Betrachten wir zunächst einmal den Zustand

$$|\nu_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-2\pi i s \frac{k}{r}\right) |a^k \pmod{N}\rangle$$

wobei $s \in \{0, 1, \dots, r - 1\}$ gelte. Dieser Zustand enthält offensichtlich Informationen über die gesuchte Ordnung r (wie wir diesen Zustand konstruieren können, folgt später).

Wir zeigen zunächst das $|\nu_s\rangle$ ein Eigenvektor von U mit Eigenwert $\exp(2\pi i \frac{s}{r})$ ist, also $U|\nu_s\rangle = \exp(2\pi i \frac{s}{r}) |\nu_s\rangle$ gilt.

Ordnungsbestimmung

Für die Wirkung von U auf $|v_s\rangle$ gilt

$$\begin{aligned} U|v_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-2\pi i s \frac{k}{r}\right) U|a^k \pmod{N}\rangle \\ &= \frac{1}{\sqrt{r}} \left\{ \exp\left(-2\pi i s \frac{0}{r}\right) U|a^0 \pmod{N}\rangle \right. \\ &\quad + \exp\left(-2\pi i s \frac{1}{r}\right) U|a^1 \pmod{N}\rangle + \dots \\ &\quad + \exp\left(-2\pi i s \frac{r-2}{r}\right) U|a^{r-2} \pmod{N}\rangle \\ &\quad \left. + \exp\left(-2\pi i s \frac{r-1}{r}\right) U|a^{r-1} \pmod{N}\rangle \right\} \end{aligned}$$

Ordnungsbestimmung

Durch Berechnung auf jedem Summanden folgt

$$U|v_s\rangle = \frac{1}{\sqrt{r}} \left\{ \begin{aligned} & \exp\left(-2\pi i s \frac{0}{r}\right) |a^1 \pmod{N}\rangle \\ & + \exp\left(-2\pi i s \frac{1}{r}\right) |a^2 \pmod{N}\rangle + \dots \\ & + \exp\left(-2\pi i s \frac{r-2}{r}\right) |a^{r-1} \pmod{N}\rangle \\ & + \exp\left(-2\pi i s \frac{r-1}{r}\right) |a^r \pmod{N}\rangle \end{aligned} \right\}$$

Ordnungsbestimmung

Im letzten Summanden bemerken wir $a^r \pmod{N} = a^0 \pmod{N}$.
Wir sortieren die Terme dementsprechend neu und erhalten

$$U|v_s\rangle = \frac{1}{\sqrt{r}} \left\{ \exp\left(-2\pi i s \frac{r-1}{r}\right) |a^0 \pmod{N}\rangle \right. \\ \left. + \exp\left(-2\pi i s \frac{0}{r}\right) |a^1 \pmod{N}\rangle \right. \\ \left. + \exp\left(-2\pi i s \frac{1}{r}\right) |a^2 \pmod{N}\rangle + \dots \right. \\ \left. + \exp\left(-2\pi i s \frac{r-2}{r}\right) |a^{r-1} \pmod{N}\rangle \right\}$$

Ordnungsbestimmung

Multiplikation mit $1 = \exp(0) = \exp\left(2\pi i \frac{s}{r}\right) \exp\left(-2\pi i \frac{s}{r}\right)$

$$U|v_s\rangle = \exp\left(2\pi i \frac{s}{r}\right) \frac{1}{\sqrt{r}} \left\{ \begin{aligned} &\exp\left(-2\pi i s \frac{1}{r}\right) |a^0 \pmod{N}\rangle \\ &+ \exp\left(-2\pi i s \frac{1}{r}\right) |a^1 \pmod{N}\rangle \\ &+ \exp\left(-2\pi i s \frac{2}{r}\right) |a^2 \pmod{N}\rangle + \dots \\ &+ \exp\left(-2\pi i s \frac{r-1}{r}\right) |a^{r-1} \pmod{N}\rangle \end{aligned} \right\}$$

Ordnungsbestimmung

Wir berücksichtigen noch das für die Amplitude des ersten Summanden

$$\exp\left(-2\pi i s \frac{r}{r}\right) = 1 = \exp\left(-2\pi i s \frac{0}{r}\right)$$

gilt und erhalten damit

$$U|v_s\rangle = \exp\left(2\pi i \frac{s}{r}\right) |v_s\rangle$$

Also ist $|v_s\rangle$ ein Eigenvektor von U mit Eigenwert $\exp(2\pi i \frac{s}{r})$. Die Phase enthält die für uns relevante Information über die Ordnung r .

Ordnungsbestimmung

Zur Abschätzung des Eigenwertes bzw. der Phase, haben wir den Schaltkreis zur Phasenabschätzung kennengelernt. Dieser liefert uns eine Abschätzung für $\frac{s}{r}$, aus welcher wir eine Abschätzung für die Ordnung r bestimmen können.

Es gibt jedoch noch offene Fragen:

1. Wie werden die kontrollierten U -Gatter in der Phasenabschätzung konstruiert?
2. Wie wird der Eigenvektor $|v_s\rangle$ konstruiert?
3. Wie wird der Output der Phasenabschätzung, d.h. die m -Bit Approximation an $\frac{s}{r}$ weiter verarbeitet, um r zu bestimmen?

Ordnungsbestimmung

1. Frage: Wie werden die kontrollierten U -Gatter in der Phasenabschätzung konstruiert?

Wir benötigen Gatter für die kontrollierte Anwendung von $U, U^2, \dots, U^{2^{m-1}}$, wenn wir den Eigenwert bis auf $m = O(n)$ -Bit Genauigkeit approximieren wollen. Wir verwenden

$$CU^{2^j}|z\rangle|y\rangle = |z\rangle|a^{z2^j}y \pmod{N}\rangle$$

Ist im Kontrollqubit $z = 0$, dann bleibt im Zielqubit unverändert y , denn $0 \leq y < N$. Ist das Kontrollqubit $z = 1$, dann wird im Zielqubit Multiplikation mit a^{2^j} mit anschließender Reduktion modulo N durchgeführt.

Ordnungsbestimmung

Es sollte noch erwähnt werden das es für Square-and-Mutiply schnelle Verfahren in klassischen Schaltkreisen gibt. Diese können reversibel ausgelegt werden, sodass uns diese auch als Quantenschaltkreise zur Verfügung stehen.

Das benötigt Zeit $O(n^2)$.

Ordnungsbestimmung

2. Frage: Wie wird der Eigenvektor $|v_s\rangle$ konstruiert?

Anstelle eines einzigen Eigenvektors konstruieren wir die gleichgewichtete Superposition

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle$$

Gehen wir damit in die Phasenabschätzung ein, erhalten wir eine m -Bit Approximation an $\frac{s}{r}$ für ein $s \in \{0, 1, \dots, r - 1\}$, wobei jeder Wert von s mit Wahrscheinlichkeit $\frac{1}{r}$ angenommen wird.

Ordnungsbestimmung

Betrachten wir die Superposition genauer:

$$\begin{aligned}\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle &= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-2\pi i \frac{sk}{r}\right) |a^k \pmod{N}\rangle \\ &= \frac{1}{r} \sum_{k=0}^{r-1} \left(\sum_{s=0}^{r-1} \exp\left(-2\pi i \frac{sk}{r}\right) \right) |a^k \pmod{N}\rangle\end{aligned}$$

Für die innere Summe gilt

$$\sum_{s=0}^{r-1} \exp\left(-2\pi i \frac{sk}{r}\right) = \begin{cases} r & \text{falls } k = 0 \\ 0 & \text{sonst} \end{cases}$$

Ordnungsbestimmung

Für $k = 0$ ist

$$\sum_{s=0}^{r-1} \exp\left(-2\pi i \frac{sk}{r}\right) = \sum_{s=0}^{r-1} 1 = r$$

Für $k \neq 0$ und $\exp\left(-2\pi i \frac{k}{r}\right)$ ist

$$\sum_{s=0}^{r-1} \exp\left(-2\pi i \frac{sk}{r}\right) = \sum_{s=0}^{r-1} \omega^s = \frac{1 - \omega^r}{1 - \omega} = 0$$

Ordnungsbestimmung

Also

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle = \frac{1}{r} \cdot r |a^0 \pmod{N}\rangle = |1 \pmod{N}\rangle$$

Die gleichverteilte Superposition über die Eigenzustände $|v_s\rangle$ kann also leicht erzeugt werden, indem man mit $|0\dots0\rangle$ startet und dann ein X-Gatter auf das im Register ganz rechts stehende Qubit anwendet:

$$|0\dots01\rangle = |1 \pmod{N}\rangle$$

Ordnungsbestimmung

3. Frage: Wie wird der Output der Phasenabschätzung, d.h. die m -Bit Approximation an $\frac{s}{r}$ weiter verarbeitet, um r zu bestimmen?

Das ist technisch etwas aufwändiger und würde eine Analyse mittels Kettenbrüchen erfordern. Solche Kettenbrüche sind Gegenstand einer Vorlesung zur Zahlentheorie.

Ordnungsbestimmung

Wir untersuchen noch die Laufzeit des Verfahrens zur Ordnungsbestimmung:

Zur Vorbereitung des Eigenvektorregisters $|0\dots01\rangle$ wird ein X-Gatter benötigt, dann m Hadamard-Gatter und m kontrollierte U^P -Gatter, sowie die inverse QFT auf m Qubits.

Jedes der kontrollierten U^P -Gatter benötigt selbst wiederum $O(n^2)$ Gatter, insgesamt also $O(mn^2) = O(n^3)$ Gatter.

Die inverse QFT verwendet $O(m^2) = O(n^2)$ Gatter und die Kettenbruchentwicklung benötigt $O(n^3)$ Gatter.

Die Komplexität des Quantenschaltkreises zur Ordnungsbestimmung ist also $O(n^3)$. Das Verfahren ist effizient.

Algorithmus von Shor

Damit erhalten wir den Algorithmus von Shor.

Faktorisierungsalgorithmus

Eingabe: Eine ungerade ganze Zahl n , die keine Primzahlpotenz ist.

Ausgabe: Ein echter Teiler von n .

1. Wähle zufällig eine Zahl $a \in \{2, \dots, n - 1\}$.
2. $z \leftarrow \text{ggT}(a, n)$.
Falls $z > 1$: Ausgabe von z . Abbruch
3. Ermittle die Periode p von $a^x \pmod{n}$ mit dem Quantenalgorithmen zur Ordnungsbestimmung.
4. Falls p ungerade ist: Beginne erneut mit Schritt 1.
5. Ermittle $\text{ggT}(a^{p/2} - 1, n)$ und $\text{ggT}(a^{p/2} + 1, n)$. Hat sich kein echter Teiler ergeben, beginne erneut mit Schritt 1.
Sonst: Ausgabe z .

Algorithmus von Shor

Der Algorithmus von Shor zur Bestimmung der Periode einer Funktion hat noch weitere Anwendungen in der Kryptographie.

Diskreter Logarithmus

Sei G eine Gruppe und $g, h \in G$. Existiert ein $d \in \mathbb{N}$ mit $h = g^d$, dann heißt d der *diskrete Logarithmus* von h zur Basis g , geschrieben $d = \text{dlog}_g(h)$.

Die Bestimmung von $d = \text{dlog}_g(h)$ bei gegebenem g und h nennt man das *diskrete Logarithmus Problem (DLP)*.

Beispiel: In \mathbb{Z}_{11} gilt $16 = 4^2 \equiv 5 \pmod{11}$, d.h. $4 = \text{dlog}_2(5)$.

Algorithmus von Shor

Diffie-Hellman Schlüsseltausch

Alice	Bob	Öffentlich bekannt
Einigen sich auf $g \in G$	g	
Wählt $a \in \mathbb{N}$	Wählt $b \in \mathbb{N}$	A, B
Berechnet $A = g^a$	Berechnet $B = g^b$	
Sendet A an Bob	Sendet B an Alice	
$K = B^a = g^{ab}$	$K = A^b = g^{ab}$	

Algorithmus von Shor

Das ordnet sich in ein allgemeineres Problem ein

Hidden Subgroup

Sei G eine Gruppe, $H \leq G$ eine Untergruppe und S eine endliche Menge. Man sagt $f : G \rightarrow S$ versteckt die Untergruppe H , wenn für beliebige $g_1, g_2 \in G$ gilt:

$$f(g_1) = f(g_2) \iff g_1^{-1}g_2 \in H \iff g_1H = g_2H.$$

D.h. f versteckt H genau dann, wenn f konstant auf einer bel. Linksnebenklasse ist und auf verschiedenen Linksnebenklassen von H unterschiedliche Werte annimmt.

Algorithmus von Shor

Hidden Subgroup Problem

Sei f die Untergruppe H von G versteckende Funktion. Die Aufgabe der Identifikation von H mit Hilfe von f nennt man *Hidden Subgroup Problem (HSP)*.

Ist G eine abelsche Gruppe, dann spricht man vom *Abelian Hidden Subgroup Problem*.

DLP ist ein HSP / AHSP und für die Anzahl der Schritte zur Lösung des AHSP mittel Shor Algorithmus gilt

$$S_{AHSP}(|G|) = \text{poly}(\log |G|).$$

Algorithmus von Shor

Der Algorithmus von Shor hat also insbesondere Konsequenzen für die „klassische“ asymmetrische Kryptographie, denn er löst

- das Faktorisierungsproblem (RSA), bzw.
- das Problem des diskreten Logarithmus (Diffie-Hellmann oder elliptische Kurven Kryptographie)

in polynomieller Laufzeit.

Das motiviert

- Post-Quantum Kryptographie, sowie
- Quantum-Key Distribution.

Inhaltsverzeichnis

1 Grundlagen

- Einleitung & Ein-Qubit Systeme
- n -Qubit Systeme & Grundlegende Schaltkreise
- Verschränkung, Deutsch-Jozsa, Bernstein-Vazirani & Simon

2 Quantensuche

- Algorithmus von Grover
- Varianten der Quantensuche

3 Fouriertransformation & Shor

- RSA & periodische Funktionen
- Diskrete- & Quanten-Fouriertransformation
- Algorithmus von Shor

4 Ausblick: QEC & QKD

Quantenfehlerkorrektur

Quantum Error Correction

Fehler auf Qubits

In unseren bisherigen Betrachtungen haben wir einen störungsfreien Quantencomputer zugrunde gelegt.

Wir betrachten nun Auswirkungen von Störungen / Rauschen auf Qubits und in diesem Zusammenhang die notwendige *Quantenfehlerkorrektur*.

Wird ein Qubit realisiert, dann ist eine dauerhafte Abschottung nicht möglich. Es steht in Wechselwirkung mit seiner Umgebung, d.h. die *Außenwelt* übt Einfluss auf den Zustand des Qubits aus (und kann diesen verändern; den Einfluss des Qubits auf seine Umgebung wollen wir als vernachlässigbar ansehen).

Fehler auf Qubits

Eine Situation, in der Qubits korrigiert werden mussten, kennen wir bereits von der Teleportation:

Bitzustand	Fehlerart	Operation
$\alpha 0\rangle + \beta 1\rangle$	kein Fehler	I_2
$\alpha 0\rangle - \beta 1\rangle$	Phasenflip	Z
$\alpha 1\rangle + \beta 0\rangle$	Bitflip	X
$\alpha 1\rangle - \beta 0\rangle$	Phasen- & Bitflip	$X \cdot Z$

Fehler auf Qubits

Beobachtung: Mit einem Verfahren, dass den Bitflip X , den Phasenflip Z und die Kombination $X \cdot Z$ korrigiert, können ungewollte Veränderungen auf einem Qubit rückgängig gemacht werden.

Das ist keine Überraschung:

Satz

Jede komplexe (2×2) -Matrix M lässt sich als Linearkombination $M = \alpha I_2 + \beta X + \gamma XZ + \delta Z$ darstellen.

Fehler auf Qubits

Genauer:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \alpha I_2 + \beta X + \gamma XZ + \delta Z$$

für

$$\alpha = \frac{1}{2}(a + d), \quad \beta = \frac{1}{2}(b + c),$$

$$\gamma = \frac{1}{2}(c - b), \quad \delta = \frac{1}{2}(a - d).$$

Eine ähnliche Darstellung kennen wir unter Verwendung der Pauli-Matrizen.

Fehler auf Qubits

Da jeder Übergang eines Qubits $\alpha|0\rangle + \beta|1\rangle$ in einen ungewollten Zustand $\alpha'|0\rangle + \beta'|1\rangle$ den Gesetzen der Quantenmechanik folgt, ist dies durch eine unitäre Transformation darstellbar, also insbesondere als Linearkombination.

Zusammenfassung

Fehler auf einem Qubit lassen sich als Linearkombination
 $F = aI_2 + bX + cXZ + dZ$ darstellen.

Fehler auf Qubits

In der klassischen Fehlerkorrektur verwendet man Redundanz um Bits gegen Fehler zu schützen, d.h. es werden mehr Bits verwendet als mindestens notwendig wären.

Beispiel: Der Wiederholungscode

Ein Datenbit wird zu einem Codewort der Länge drei, indem es entsprechend kopiert wird: $0 \rightarrow 000$ bzw. $1 \rightarrow 111$.

Wird das Codewort 001 empfangen, kann dieses zu 000 korrigiert werden; vorausgesetzt, es ist nur an einer Stelle des Codeworts ein Fehler aufgetreten.

Natürlich gibt es bessere Codierungsverfahren als den Wiederholungscode \rightsquigarrow *Codierungstheorie*

Fehler auf Qubits

Einige Schwierigkeiten bei der Korrektur von Qubits sind bereits erkennbar:

- Der Wiederholungscode kopiert ein klassisches Bit. Nach dem No-Cloning Theorem können Qubits nicht beliebig kopiert werden.
- Im Wiederholungscode wird ein *Mehrheitsvotum* der Bits verwendet um erkannte Fehler zu korrigieren. Für Qubits würde dies eine Messung bedeuten, die das Codewort zerstört.
- Auf Qubits können beliebig viele Fehler auftreten (Bitwechsel *und* Amplitude).

Bitflip-Code mit Syndrom

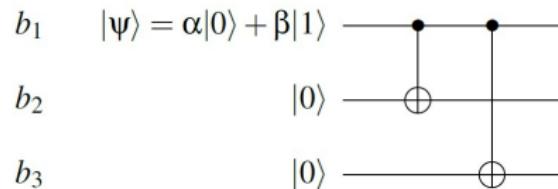
Wir wollen ausnutzen, dass sich jeder Fehler F auf einem Qubit als Linearkombination von I_2, X, Z und $X \cdot Z$ darstellen lassen.

Zuerst: Bitflip-Code mit Syndrom (Korrektur des Codeworts)

Ziel: Qubit im Zustand $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ gegen Bitflip X sichern

Bitflip-Code mit Syndrom

Kopien können nicht erzeugt werden, jedoch Verschränkung. Wir verwenden ein 3-Qubit Register.



Aus $|b_1 b_2 b_3\rangle = (\alpha|0\rangle + \beta|1\rangle)|0\rangle|0\rangle = \alpha|000\rangle + \beta|100\rangle$ wird durch zweifaches CNOT das Codewort

$$|c\rangle = \alpha|000\rangle + \beta|111\rangle.$$

Bitflip-Code mit Syndrom

Ein Bitflip auf dem ersten Bit lässt sich durch $F = X \otimes I_2 \otimes I_2$ beschreiben. Es gilt

$$F|c\rangle = \alpha|100\rangle + \beta|011\rangle.$$

Insgesamt gibt es vier Möglichkeiten:

Bitflip	Registerzustand
kein Fehler	$\alpha 000\rangle + \beta 111\rangle$
b_1	$\alpha 100\rangle + \beta 011\rangle$
b_2	$\alpha 010\rangle + \beta 101\rangle$
b_3	$\alpha 001\rangle + \beta 110\rangle$

Bitflip-Code mit Syndrom

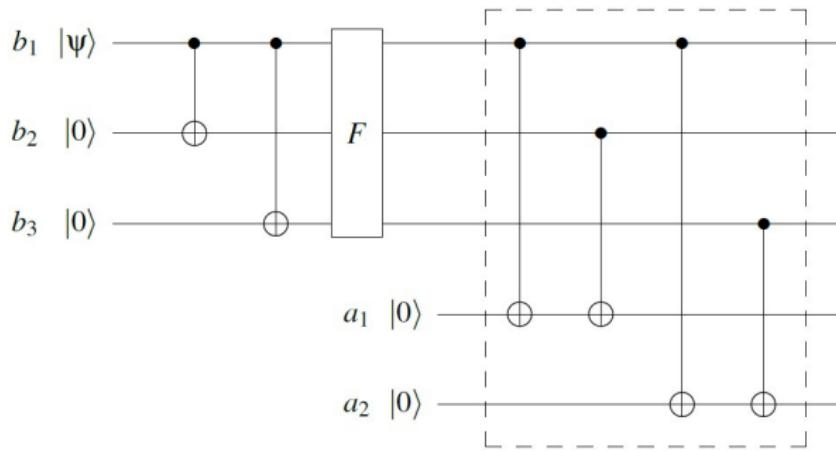
Wir beobachten: Sind alle Bits gleich, ist kein Fehler aufgetreten.

Bitflip	Eigenschaft
kein Fehler	$b_1 = b_2 = b_3$
b_1	$b_1 \neq b_2 = b_3$
b_2	$b_2 \neq b_1 = b_3$
b_3	$b_3 \neq b_1 = b_2$

Es genügt also ein Vergleich von b_1 mit b_2 und b_1 mit b_3 .

Bitflip-Code mit Syndrom

Diese Vergleiche lassen sich *ohne* Messung (*nur* mittels Schaltkreis) durchführen:



Bitflip-Code mit Syndrom

Die Hilfsbits a_1 und a_2 werden als *Syndrombits* bezeichnet, wobei $a_1 = b_1 \oplus b_2$ und $a_2 = b_1 \oplus b_3$. Es gilt

- $a_1 = 1$, wenn $b_1 \neq b_2$,
- $a_2 = 1$, wenn $b_1 \neq b_3$.

Der aufgetretene Fehler wird ermittelt, ohne das $|c\rangle$ gemessen wurde.

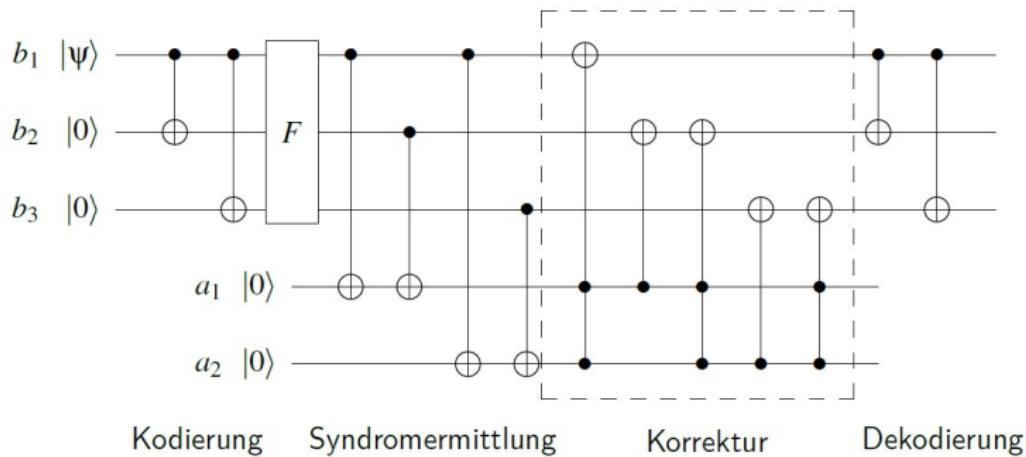
Bitflip-Code mit Syndrom

Wir beobachten:

$a_1 a_2$	Bitflip	Eigenschaft
00	keiner	$b_1 = b_2 = b_3$
10	b_2	$b_2 \neq b_1 = b_3$
01	b_3	$b_3 \neq b_1 = b_2$
11	b_1	$b_1 \neq b_2 = b_3$

Für die Korrektur werden a_1 und a_2 als Kontrollbits verwendet (d.h. eine Messung ist nicht notwendig).

Bitflip-Code mit Syndrom



Bitflip-Code mit Syndrom

Entsprechend der Tabelle werden drei Fälle unterschieden:

1. b_1 muss korrigiert werden, d.h. $a_1 = a_2 = 1$.
Dafür genügt ein Toffoli-Gatter.
2. b_2 muss korrigiert werden, d.h. $a_1 = 1, a_2 = 0$.
 $a_1 = 1$ wird zur Kontrolle eines CNOT verwendet (unabhängig vom Wert von a_2). Für $a_1 = a_2 = 1$ wird b_2 fälschlich gekippt.
Dies wird durch ein weiteres Toffoli-Gatter ausgeglichen.
Dieses kippt b_2 für $a_1 = a_2 = 1$ zurück.
3. b_3 muss gekippt werden, d.h. $a_1 = 0, a_2 = 1$.
Diese wird analog zum zweiten Fall umgesetzt.

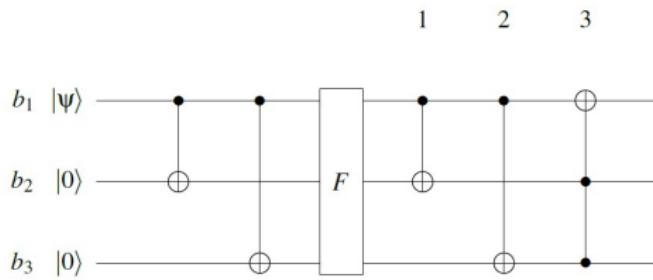
Nach der Korrektur wurde das Codewort $|c\rangle$ wieder hergestellt und kann mit zwei CNOT-Gattern decodiert werden (*uncomputing*).

Bitflip-Code ohne Syndrom

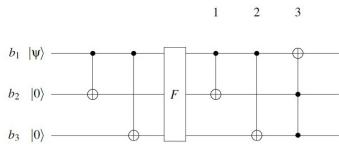
Soll nur das Ausgangsbit wiederhergestellt werden, ohne das gesamte Codewort zu korrigieren, kann auf zusätzliche Syndrombits verzichtet werden

↔ Bitflip-Code ohne Syndrom (Korrektur des Datenbits)

Übung: Untersuchen Sie die vier Fälle des vorherigen Abschnitts in dem Schaltkreis.



Übung: Untersuchen Sie die vier Fälle des vorherigen Abschnitts in dem Schaltkreis.



für $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ wird zu Beginn wieder das Codewort $\alpha|000\rangle + \beta|111\rangle$ erzeugt.

1. Fall

$$\tilde{\Gamma}_1 = I_2 \otimes I_2 \otimes I_2$$

$$(\otimes) \oplus (I \otimes I_2)$$

$$\begin{array}{ccccccc} 000 & 111 & \xrightarrow{\quad} & 100 & 011 & & \\ \swarrow 1 & \swarrow 2 & & \swarrow 1 & \swarrow 1 & & \\ 011 & 011 & & 011 & 011 & & \\ \cancel{011} & \cancel{011} & & \cancel{011} & \cancel{011} & & \\ \end{array}$$

~~011 (1, 101) (11)~~

Bitflip-Code ohne Syndrom

Wir beobachten: Sind alle drei Bits des Registers gleich, ist kein Fehler aufgetreten. Ein einzelner Bitflip kann korrigiert werden.

Für $F = X \otimes X \otimes I_2$ ist jedoch

$$\begin{aligned}\alpha|000\rangle + \beta|111\rangle &\xrightarrow{F} \alpha|110\rangle + \beta|001\rangle \\ &\xrightarrow{1} \alpha|100\rangle + \beta|001\rangle \\ &\xrightarrow{2} \alpha|101\rangle + \beta|001\rangle \\ &\xrightarrow{3} \alpha|101\rangle + \beta|001\rangle = (\alpha|1\rangle + \beta|0\rangle)|01\rangle\end{aligned}$$

Wie schon der (klassische) Wiederholungscode scheitert der Ansatz wenn mehrere Bits korrigiert werden müssen.

Korrektur von Phasenflips

Wir betrachten nun die Korrektur von Phasenflips.

Übung: Wenden Sie den Bitflip X und den Phasenflip Z auf die Zustände $|0\rangle$, $|1\rangle$, $|+\rangle$ und $|-\rangle$ an. Erkennen Sie einen Zusammenhang zwischen Bitflip und Phasenflip?

	X	Z
Computational Basis	$ 0\rangle$	$ 0\rangle$
	$ 1\rangle$	$- 1\rangle$
Hadamard Basis	$ +\rangle$	$ -\rangle$
	$ -\rangle$	$ +\rangle$

$$\begin{aligned}
 X = \text{Bitflip} \quad X|+\rangle &= X\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \\
 Z = \text{Phasenflip} \quad &= \frac{1}{\sqrt{2}}(X|0\rangle + X|1\rangle) = \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) \\
 &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle
 \end{aligned}$$

Korrektur von Phasenflips

Wir betrachten nun die Korrektur von Phasenflips.

Übung: Wenden Sie den Bitflip X und den Phasenflip Z auf die Zustände $|0\rangle, |1\rangle, |+\rangle$ und $|-\rangle$ an. Erkennen Sie einen Zusammenhang zwischen Bitflip und Phasenflip?

Bzgl. der Hadamard-Basis $\{|+\rangle, |-\rangle\}$ verhält sich der Phasenflip Z also wie der Bitflip X bzgl. der Standardbasis $\{|0\rangle, |1\rangle\}$.

Wollen wir einen 3-Bit Code für Phasenflips entwickeln, ist es günstig ein Codewort in der Form

$$\alpha|+\rangle|+\rangle|+\rangle + \beta|-\rangle|-\rangle|-\rangle$$

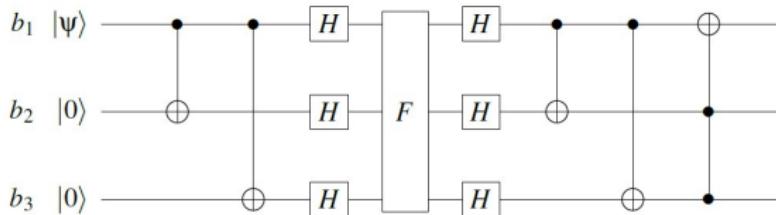
darzustellen.

Korrektur von Phasenflips

Ein Phasenflip auf dem ersten Bit würde dann

$$\alpha|-\rangle|+\rangle|+\rangle + \beta|+\rangle|-\rangle|-\rangle$$

ergeben. Nach einem Basiswechsel (durch Anwendung der Hadamard-Transformation) lässt sich der bereits bekannte Bitflip-Code übertragen:



Shors 9-Qubit Code

Allgemein ist ein Fehler auf einem Qubit als Linearkombination $F = aI_2 + bX + cXZ + dZ$ darstellbar. Nach derzeitigem Stand können Bitsflips und Phasenflips korrigiert werden.

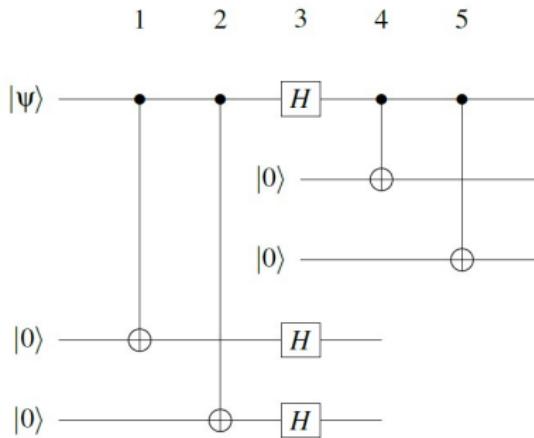
Ziel: Ein Verfahren, dass die Korrektur von Bitflip, Phasenflip und deren Kombination XZ in sich vereint. Das führt uns zu Shors 9-Qubit Code.

Idee: zwei Stufen

- Umwandlung des zu schützenden Qubits $|\psi\rangle$ mit dem Phasenflip-Code in ein Codewort der Länge drei
- Kodierung jedes der drei Qubits mit dem Bitflip-Code

Shors 9-Qubit Code

Wir betrachten zuerst einen Teil des Verfahrens:



Shors 9-Qubit Code

Nach Anwendung der **Schritte 1-3** befinden sich die drei Qubits im Zustand $\alpha|+\rangle|+\rangle|+\rangle + \beta|-\rangle|-\rangle|-\rangle$, also dem Codewort des Phasenflip-Codes.

Dann werden zwei weitere Qubits hinzugefügt, um das erste Qubit des Codeworts zu schützen:

$$\alpha|+\rangle|00\rangle|+\rangle|+\rangle + \beta|-\rangle|00\rangle|-\rangle|-\rangle.$$

Das ist der Zustand des Registers vor der Anwendung der Schritte 4 und 5.

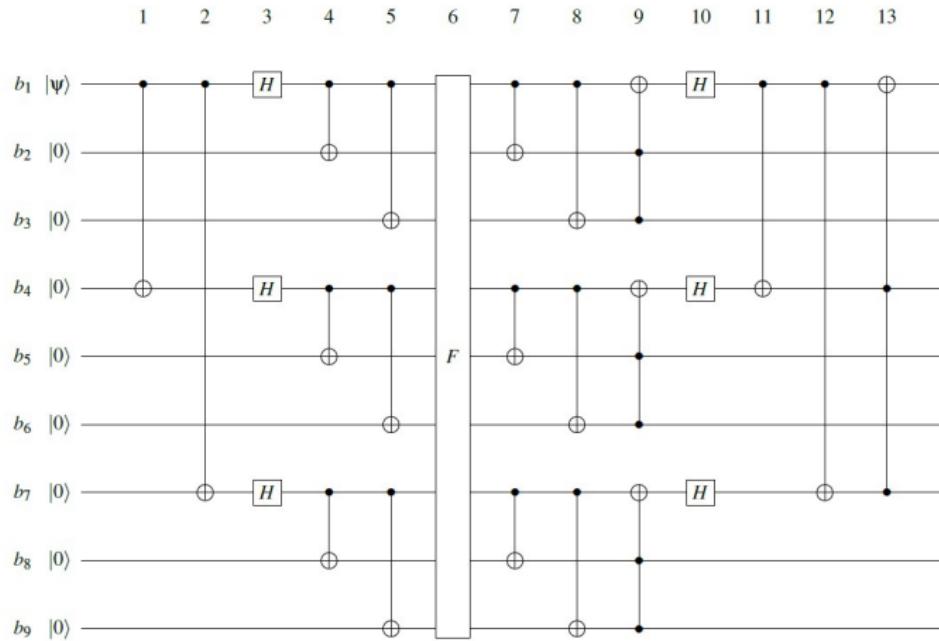
Shors 9-Qubit Code

Werden nur die ersten drei Qubits betrachtet, ergibt die Anwendung von CNOT (**Schritt 4 & 5**)

$$\begin{aligned}\alpha|+\rangle|00\rangle + \beta|-\rangle|00\rangle &= \\ &= \frac{\alpha}{\sqrt{2}}(|000\rangle + |100\rangle) + \frac{\beta}{\sqrt{2}}(|000\rangle - |100\rangle) \\ &\xrightarrow{4,5} \frac{\alpha}{\sqrt{2}}(|000\rangle + |111\rangle) + \frac{\beta}{\sqrt{2}}(|000\rangle - |111\rangle)\end{aligned}$$

Das gesamte Verfahren setzt sich entsprechend zusammen.

Shors 9-Qubit Code



Shors 9-Qubit Code

Zu Beginn befindet sich das Register $|b_1 \dots b_9\rangle$ im Zustand

$$\begin{aligned} |\phi_0\rangle &= (\alpha|0\rangle + \beta|1\rangle)|00000000\rangle \\ &= \alpha|000\rangle|000\rangle|000\rangle + \beta|100\rangle|000\rangle|000\rangle \end{aligned}$$

Anwendung von CNOT in **Schritt 1 & 2** ergibt

$$|\phi_2\rangle = \alpha|000\rangle|000\rangle|000\rangle + \beta|100\rangle|100\rangle|100\rangle$$

und Hadamard-Transformation in **Schritt 3** liefert

$$|\phi_3\rangle = \alpha|+00\rangle|+00\rangle|+00\rangle + \beta|-00\rangle|-00\rangle|-00\rangle$$

Shors 9-Qubit Code

Die Qubits $|b_1\rangle$, $|b_4\rangle$ und $|b_7\rangle$ sind verschränkt und die Qubits mit Wert $|0\rangle$ dienen der Kodierung.

Betrachtet man für einen (isolierten) Dreierblock die **Schritte 4 & 5**, so ist

$$|+00\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |100\rangle) \xrightarrow{4,5} \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) := |c^+\rangle$$

bzw.

$$|-00\rangle = \frac{1}{\sqrt{2}}(|000\rangle - |100\rangle) \xrightarrow{4,5} \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle) := |c^-\rangle$$

Shors 9-Qubit Code

Anwendung von CNOT in den Schritten 4 & 5 ergibt damit den Gesamtzustand

$$\begin{aligned} |\phi_5\rangle &= \frac{\alpha}{\sqrt{8}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \\ &\quad + \frac{\beta}{\sqrt{8}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \\ &= \alpha|c^+c^+c^+\rangle + \beta|c^-c^-c^-\rangle \end{aligned}$$

Die Bits $|b_1\rangle, |b_4\rangle$ und $|b_7\rangle$ repräsentieren den Phasenflip-Code in der Hadamard-Basis, welcher um den Bitflip-Code erweitert wurde. $|\phi_5\rangle$ ist das aus $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ entstandene Codewort.

Shors 9-Qubit Code

In Schritt 6 wirkt ein Fehler F auf $|\phi_5\rangle$. Dieser wird (analog zur Analyse des Bitflip- bzw. Phasenflip-Codes) im Übrigen Verlauf der Schaltung korrigiert.

Übung: Untersuchen Sie die Korrektur des Fehlers XZ auf das erste Qubit $|b_1\rangle$.

Shors 9-Qubit Code

In Schritt 6 wirkt ein Fehler F auf $|\phi_5\rangle$. Dieser wird (analog zur Analyse des Bitflip- bzw. Phasenflip-Codes) im Übrigen Verlauf der Schaltung korrigiert.

Übung: Untersuchen Sie die Korrektur des Fehlers XZ auf das erste Qubit $|b_1\rangle$.

Für $F = aI_2 + bX + cXZ + dZ$ ergibt sich für das Codewort $|c\rangle = \alpha|c^+c^+c^+\rangle + \beta|c^-c^-c^-\rangle$ dann

$$F|c\rangle = a|c\rangle + bX|c\rangle + cXZ|c\rangle + dZ|c\rangle$$

In jeder der vier Komponenten wird der jeweilige Fehler korrigiert.

Shors 9-Qubit Code

Unsere bisherige Betrachtung ist aber immer noch praxisfern, wird doch davon ausgegangen das auf einem Qubit ein (beliebiger) Fehler aufgetreten, während sich die übrigen Qubits nicht ändern dürfen.

Realistischer ist es doch, dass auf jedes Bit eine kleine Veränderung wirkt (Rauschen), bspw.

$$\cos(\epsilon)I_2 + i \sin(\epsilon)X$$

Das ist eine unitäre Transformation die für kleines $\epsilon > 0$ nahe der Identität liegt.

Shors 9-Qubit Code

Treten nur minimale Fehler auf (d.h. kein vollständiger Bitflip / Phasenflip), lassen sich diese durch eine Messung korrigieren oder zu einem vollständigen Bitflip / Phasenflip umwandelt. Diese lassen sich mit dann mit Shors 9-Qubit Code korrigieren.

Effizienter ist jedoch der 5-Qubit Code (von Laflamme, Miquel, Paz und Zureck). Fünf Qubits sind minimal zur Absicherung gegen einen einzelnen Bitfehler.

Mittels Gruppentheorie ist auch eine einheitliche Untersuchung von Codes zur Quantenfehlerkorrektur möglich.