

# Rechenoperationen und Elementare Algorithmen

Michael Goerz

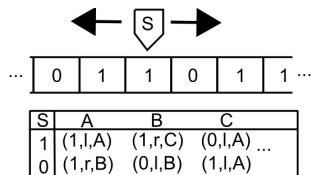
FU Berlin

Lehrseminar Quantencomputer SoSe 2007  
3. März 2007

# Gliederung

- 1 Einführung – Computermodelle
- 2 Quantencomputing
- 3 Das Quantum-Circuit-Gate-Model
  - Ein-Bit-Gatter
  - Das Dekompositionstheorem
  - Kontrollierte Operationen
  - Mehr-Bit-Gatter
  - Universelle Gatter
- 4 Klassische Algorithmen vs. Quantenalgorithmen
  - Von Logischen Gattern zu Quantengattern
  - Der Deutsch-Algorithmus
- 5 Ausblick

# Turingmaschine



Jeder klassische Algorithmus kann von einer Turingmaschine simuliert werden.

- Unendlich langes Speicherband
- Ein Zeichen aus dem Alphabet pro Zelle
- Programmgesteuerter Kopf

# Codedarstellung und Komplexitäten

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define SEED timer()

int randomwalk(int min, int max, int steps) {
    int current = start;
    int i;
    for (i=0; i<steps; i++){
        current += (drand48() - 0.5) * (max - min);
        if (current < min) current = min;
        if (current > max) current = max;
    }
    return current;
}

long timer(){
    time_t t;
    time(&t);
    return (long)t;
}

```

Algorithmen können am einfachsten durch Programmcode dargestellt werden.

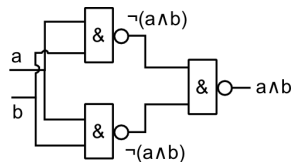
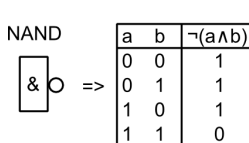
- Moderne Programmiersprachen
- Turingmaschine als Goto-Sprache
- Pseudocode

Die Effizienz von Algorithmen wird anhand von Laufzeit und Speicherverbrauch gemessen

- $O(n)$ ,  $O(n \log n)$ ,  $O(2^n)$ , ...
- P und NP

# Funktionale Theorie – Logische Gatter

- Jeder Algorithmus kann auch als Funktion von Eingabe zu Ausgabe beschrieben werden.  
Boolesche Funktion:  $\{0, 1\}^n \rightarrow \{0, 1\}^m$
- Funktionen werden über Tabellen beschrieben
- Es können 1:1 logische Gatter zugewiesen werden (Hardwareimplementierung)



# Qbits und Operatoren

- 1-Bit-System:  $\alpha_0 |0\rangle + \alpha_1 |1\rangle$   
2-Bit-System:  $\alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle$   
3-Bit-System:  $\alpha_{000} |000\rangle + \alpha_{001} |001\rangle + \alpha_{010} |010\rangle + \dots$
- $|\alpha_i^2|$  gibt die Wahrscheinlichkeit an, dass das Qbit im entsprechenden Eigenzustand gemessen wird.
- Bei einer Messung wird sich das Qbit auf jeden Fall in *irgendeinem* Eigenzustand befinden:

$$\sum_i |\alpha_i^2| = 1$$

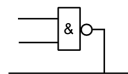
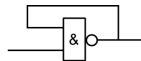
- Qbit-Systeme werden über Matrizen transformiert. Es gibt unendlich viele Transformationen. Klassisch gibt es nur endlich viele Boolesche Funktionen.

# Reversibilität

- Unitäre Transformationen: bijektive lineare Abbildung  $U$ , die Längen- und Winkelerhaltend ist (Drehspiegelung).  
 $UU^\dagger = I$
- Erhalt von  $\sum_j |\alpha_j|^2 = 1$  ist gewährleistet.
- Transformation kann die Anzahl der Qbits nicht verändern.
- Für unitäre Matrizen gilt  $UU^\dagger = I \rightarrow U^\dagger = U^{-1}$   
Alle Operationen auf Qbits sind reversibel.







# Einschränkungen für Quantengatter

- Keine Schleifen (irreversibel)
- Kein FanIn – Zusammenführung (irreversibel)
- Kein FanOut – Abzweigung (No-Cloning-Theorem)





# Matrizendarstellung der Einbitgatter

Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Pauli- $X$		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli- $Y$		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli- $Z$		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Phase		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$

# Das Dekompositionstheorem

Jede unitäre Matrix kann aus drei Drehungen und einem Phasenfaktor zusammengesetzt werden

$$U = e^{i\alpha} \underbrace{\begin{pmatrix} e^{-i\frac{\beta}{2}} & 0 \\ 0 & e^{i\frac{\beta}{2}} \end{pmatrix}}_{R_z(\beta)} \underbrace{\begin{pmatrix} \cos \frac{\gamma}{2} & -\sin \frac{\gamma}{2} \\ \sin \frac{\gamma}{2} & \cos \frac{\gamma}{2} \end{pmatrix}}_{R_y(\gamma)} \underbrace{\begin{pmatrix} e^{-i\frac{\delta}{2}} & 0 \\ 0 & e^{i\frac{\delta}{2}} \end{pmatrix}}_{R_z(\delta)}$$

Die einzelnen Drehmatrizen werden wir später durch die uns bekannten Matrizen nähern können.

# Folgerung

Aus dem Theorem ergibt sich auch die folgende Zerlegung

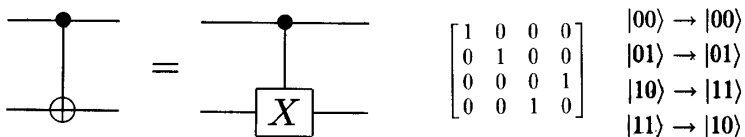
$$U = e^{i\alpha}AXBXC$$

mit

$$A = R_z(\beta)R_y(\gamma/2), \quad B = R_y(\gamma/2)R_z(-(\delta + \beta)/2),$$

$$C = R_z((\delta - \beta)/2); \quad ABC = 1$$

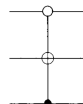
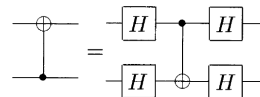
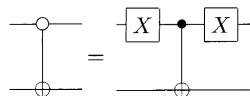
# Das CNOT Gatter



- Controlled-Not-Gatter, oder
- reversibles XOR-Gatter
- wichtiges Element der universellen Gatter (einziges 2-Bit-Gatter)

# CNOT Variationen

- invertiertes Control-Bit
- Control- und Target-Bit vertauscht
- Mehrere Control-Bits / mehrere Target-Bits



# Allgemeine Kontrollierte Operationen

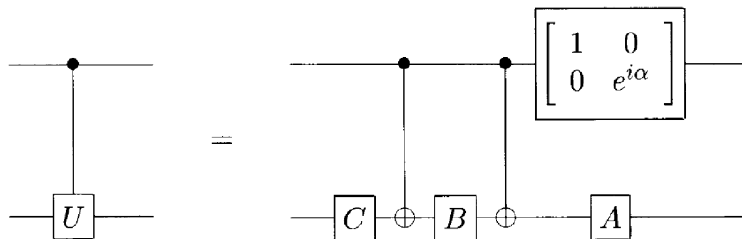
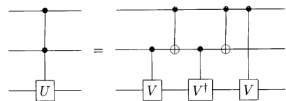


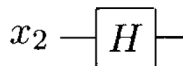
Figure 4.6. Circuit implementing the controlled- $U$  operation for single qubit  $U$ .  $\alpha$ ,  $A$ ,  $B$  and  $C$  satisfy  $U = \exp(i\alpha)AXBXC$ ,  $ABC = I$ .

Benutze  $U = e^{i\alpha}AXBXC$  mit  $ABC = I$ .  
 Mehrere Control-Bits können mit weiteren  
 einfachen Zerlegungen realisiert werden.



# Mehr-Bit-Gatter aus Ein-Bit-Gattern

- $n$ -Teilchen Hilbertraum entsteht durch Tensorprodukt  $\longrightarrow$  neue Basis
- Bei der Analyse muss dies gedanklich rückgängig gemacht werden:  
 $(0100) \Rightarrow (10), (01)$



$x_1$  —————

$$(1000) \Rightarrow (10), (10) \rightarrow (10), 1/\sqrt{2}(11) \Rightarrow 1/\sqrt{2}(1100)$$

$$(0100) \Rightarrow (10), (01) \rightarrow (10), 1/\sqrt{2}(1\bar{1}) \Rightarrow 1/\sqrt{2}(1\bar{1}00)$$

$$(0010) \Rightarrow (01), (10) \rightarrow (01), 1/\sqrt{2}(11) \Rightarrow 1/\sqrt{2}(0011)$$

$$(0001) \Rightarrow (01), (01) \rightarrow (01), 1/\sqrt{2}(1\bar{1}) \Rightarrow 1/\sqrt{2}(001\bar{1})$$

# Universelle Systeme

Universalität: Eine Menge von Gattern ist universell, wenn sich jede unitäre Transformation mit beliebiger Genauigkeit durch eine System nur dieser Gatter darstellen lässt.

- Die Menge aller Ein-Bit-Gatter mit CNOT ist universell
- Die Menge aus Hadamard, Phase, CNOT, und  $\pi/8$  Gatter ist universell (Ein-Bit-Gatter können damit genähert werden)

Man muss nur eine begrenzte Menge an Gattern physisch implementieren, um jede beliebige Transformation durchzuführen.



# Beweis für das Ein-Bit-Gatter–CNOT System

Schritt 1: Zerlegung in Zwei-Level-Gatter

- Zwei-Level-Gatter = Einheitsmatrix bis auf unitäre  $2 \times 2$ -Untermatrix
- Jede unitäre  $n \times n$ -Matrix  $U$ :  
 $U = U_1^\dagger \dots U_m^\dagger$  mit  $U_m \dots U_1 U = I$
- In jedem Schritt wird eine Null am Anfang der nächsten Zeile erzeugt

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 & 0 & 0 & c \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & 0 & d \end{bmatrix}$$

$$U = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & j \end{bmatrix}$$

$$U_1 U = \begin{bmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{bmatrix}$$

$$U_2 U_1 U = \begin{bmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{bmatrix}$$

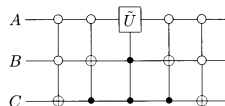
# Beweis für das Ein-Bit-Gatter–CNOT System

## Schritt 2: Zwei-Level-Gatter aus CNOT und Ein-Bit-Gattern

- Suche die beiden Eigenvektoren, die das Gatter verändert
- Schreibe den sog. „Gray-Code“ zwischen beiden Eigenvektoren aus
- Setze den Gray-Code mit erweiterten CNOT-Gattern um
- Im letzten Schritt  $\tilde{U}$ -Gatter an die Stelle des sich ändernden Bits setzen
- Gray-Code rückwärts implementieren

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

$$\tilde{U} \equiv \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \begin{array}{ccc} A & B & C \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array}$$

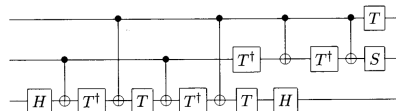
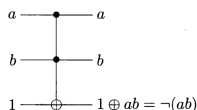


# Zusammenfassung Quantengatter

- Durch die Dekomposition von Ein-Bit-Gattern können kontrollierte Ein-Bit-Gatter implementiert werden.
- Mehrbitgatter können in 2-Level-Gatter zerlegt werden.
- 2-Level-Gatter können durch CNOT und Ein-Bit-Gatter implementiert werden.
- Ein-Bit-Gatter können mit Hadamard, Phase, CNOT, und  $\pi/8$  Gattern genähert werden.

# NAND- und Tofflogatter

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



Mit Quantengattern können alle klassischen Schaltnetze nachgebildet werden.

- Das klassische NAND-Gatter ist universell
- Toffoli-Gatter:  $(a, b, 1) \longrightarrow (a, b, \neg(a \wedge b))$
- Auch FanOut möglich:  $(1, a, 0) \longrightarrow (1, a, a)$
- Keine Verletzung des No-Cloning-Theorems!

# Der Deutsch-Algorithmus

- Mit Hilfe des Hadamard-Gatters (Superpositionszustände) kann eine Funktion mit *einer* Transformation an allen Stellen gleichzeitig berechnet werden
- Das Ergebnis liegt wieder in einem Superpositionszustand vor → nicht nützlich.
- Deutsch: Kann auch eine globale Eigenschaft der Funktion in einem Schritt berechnet werden?
- Ja: durch Interferenz kann das XOR der Funktionswerte ermittelt werden.
- Variante: Deutsch-Jozsa ermittelt, ob die Funktion konstant ist.

# Gatter-Implementierung

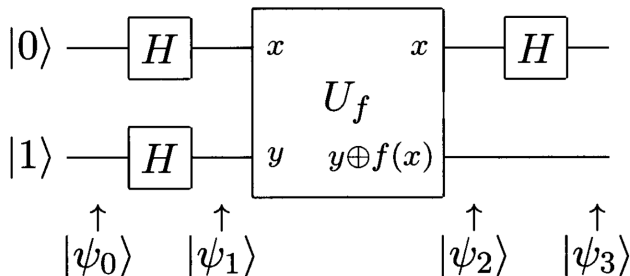
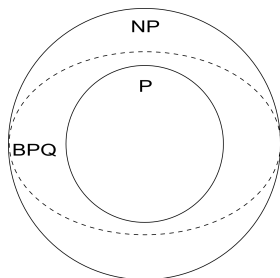


Figure 1.19. Quantum circuit implementing Deutsch's algorithm.

$$|\psi_1\rangle = \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \quad |\psi_2\rangle = \begin{cases} \pm \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) = f(1) \\ \pm \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) \neq f(1). \end{cases} \quad |\psi_3\rangle = \begin{cases} \pm|0\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) = f(1) \\ \pm|1\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) \neq f(1). \end{cases}$$

# „The Power of Quantum Computation“



- Alles, was ein Computer effizient lösen kann, kann auch ein Quantencomputer effizient lösen.
- Einiges von dem wir denken, dass es ein Computer nicht effizient lösen kann, kann der Quantencomputer effizient lösen.
- Aber: Es ist nicht bewiesen, dass Computer nicht doch die „schweren“ Probleme effizient lösen können.
- Auch für den Quantencomputer gibt es noch schwere Probleme.

# Quantensimulation

- Anzahl der Koeffizienten (der Eigenvektoren) steigt exponentiell.
- Quantensysteme klassisch schlecht simulierbar.
- Quantencomputer benötigt nur linear viele Qbits.
- Aber: Ergebnisse einer Simulation können nicht beliebig ausgelesen werden.



# Fourier- und Such-Algorithmen

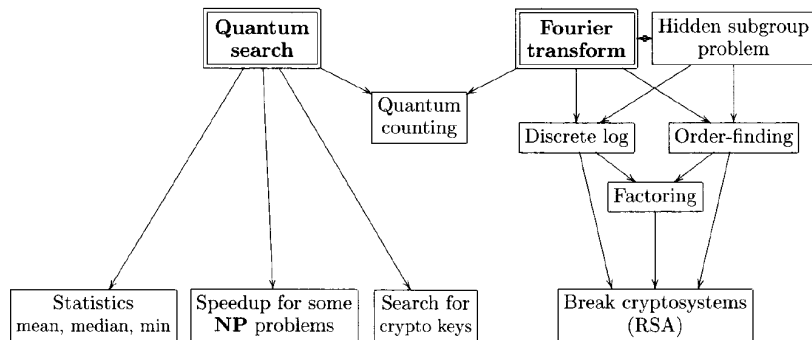


Figure 4.1. The main quantum algorithms and their relationships, including some notable applications.

# Zusammenfassung

- Qbits können als Vektoren dargestellt werden, die mit Matrizen manipuliert werden. Für jedes  $n$ -Bit System wird eine Basis aus  $2^n$  Eigenvektoren eingeführt.
- Ein-Bit-Gatter können in Rotationen zerlegt werden, welche wiederum von einer endlichen Anzahl Gatter genähert werden können.
- Alle Gatter können aus einem endlichen Satz von Universalgattern hergestellt werden.
- Für bestimmte, aber nicht alle Probleme ist der Quantencomputer mit speziellen Algorithmen fundamental effizienter als klassische Computer.

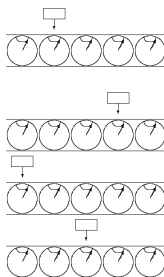
# Beweis des Dekompositionstheorems

Jede unitäre Matrix lässt sich aufgrund ihrer Orthonormalitätseigenschaften schreiben als

$$U = \begin{pmatrix} e^{i(\alpha-\beta/2-\delta/2)} \cos(\gamma/2) & -e^{i(\alpha-\beta/2+\delta/2)} \sin(\gamma/2) \\ e^{i(\alpha+\beta/2-\delta/2)} \sin(\gamma/2) & e^{i(\alpha+\beta/2+\delta/2)} \cos(\gamma/2) \end{pmatrix}$$

Man kann sich leicht überzeugen dass sich dies in die drei gewünschten Matrizen zerlegen lässt.

# Die Quantenturingmaschine und QCL-Quantencode



```

operator dft(ureg q) { // n
  const n=#q; // n
  int i; int j; // n
  for i=0 to n-1 {
    for j=0 to i-1 { // n
      CPhase(2*pi/2^(i-j+1));
    }
    Mix(q[n-i-1]); // q
  }
  flip(q); // n
}

```

Es existiert das Modell einer  
Quantenturingmaschine

- Kopf und Band bestehen aus Qbits
- Der Kopf kann in viele Richtungen gleichzeitig gehen (Quantenparallelismus)

Es existieren bereits  
Programmiersprachen für  
Quantencomputer

- QCL an der TU Wien
- Mischt klassische Kontrollstrukturen mit Quantenoperationen