

Der Deutsch-Jozsa-Algorithmus und das Penny-Flip-Spiel

Kurztext im Rahmen von „Quanten auf Reisen“

Physikdidaktik, Albert-Ludwigs-Universität Freiburg



Weitere Kurztexte hier: <https://physikdidaktik.uni-freiburg.de/kurztexte/>

Gefördert durch:



Bundesministerium
für Forschung, Technologie
und Raumfahrt

universität freiburg



Inhaltsverzeichnis

1	Der Deutsch-Jozsa-Algorithmus und das Penny-Flip-Spiel	3
1.1	Konstante und balancierte Funktionen	3
1.2	Der Deutsch-Algorithmus für $N = 1$	5
1.3	Der Deutsch-Jozsa-Algorithmus für $N > 1$	7
1.4	Das Penny-Flip-Spiel	9
1.5	Anhang	10
1.5.1	Die unitäre Implementation des Oracles	10
1.5.2	f_4	11

Kapitel 1

Der Deutsch-Jozsa-Algorithmus und das Penny-Flip-Spiel

Autor: Thomas Filk, Version vom: 30.11.2025

Der Deutsch-Jozsa-Algorithmus ist ein vergleichsweise einfacher Quantenalgorithmus, mit dem man in einer Abfrage entscheiden kann, ob eine Bit-wertige Funktion auf N -Bit-Folgen konstant ist (also allen Folgen entweder den konstanten Wert 0 oder allen Folgen den Wert 1 zuweist) oder balanciert (in diesem Fall ordnet die Funktion der Hälfte der Folgen den Wert 0 und der anderen Hälfte den Wert 1 zu). Möchte man dieses Problem mit einem klassischen Algorithmus entscheiden, benötigt man im ungünstigsten Fall für eine eindeutige Antwort $2^{N-1} + 1$ Abfragen. Mit einem Quantencomputer kann man zunächst sämtliche 2^N Folgen als Superposition in einem Zustand kodieren und die gesuchte Funktion – die man auch als Oracle bezeichnet – so auf diesen Zustand anwenden, dass man anschließend mit einer einzigen Abfrage entscheiden kann, ob eine konstante oder balancierte Funktion vorliegt.

Benannt ist der Deutsch-Jozsa-Algorithmus nach David Deutsch und Richard Jozsa, die die wesentlichen Grundlagen dieses Algorithmus' entwickelt haben [1, 2] (siehe auch [3]). Die mit diesem Algorithmus gelöste Aufgabe ist zwar von keiner praktischen Relevanz, aber das Beispiel zeigt, dass ein Quantencomputer manche Probleme in wesentlich weniger Schritten lösen kann als ein klassischer Computer, im vorliegenden Fall sind es sogar exponentiell weniger Schritte.

1.1 Konstante und balancierte Funktionen

Jeder Folge $(b_0, b_1, \dots, b_{N-1})$ ($b_i \in \{0,1\}$) von N Bits kann man eine Zahl x zwischen 0 und $2^N - 1$ zuordnen, indem man die Bitfolge als binäre Darstellung dieser Zahl interpretiert:

$$x = \sum_{i=0}^{N-1} b_i 2^i .$$

Betrachten wir nun Bit-wertige Funktionen f auf solchen Folgen, können wir ihre Urbildmenge als $\{0,1\}^N$ oder $[0,2^N - 1]$ ansehen:

$$f : \{0,1\}^N \rightarrow \{0,1\} \qquad f : [0,2^N - 1] \rightarrow \{0,1\}.$$

$\{0,1\}^N$ bezeichnet das N -fache kartesische Produkt der Menge $\{0,1\}$.

Eine konstante Funktion ordnet allen Elementen dasselbe Element ihrer Bildmenge zu. Im vorliegenden Fall gibt es nur zwei konstante Funktionen, unabhängig von N : die Funktion, die allen Elementen die 0 zuordnet, und die Funktion, die allen Elementen die 1 zuordnet. Man bezeichnet eine Funktion als balanciert, wenn jedes Element der Bildmenge gleich häufig als Funktionswert auftritt. Bei einer Bit-wertigen Funktion wird also der Hälfte der Fälle die 0 zugeordnet und der Hälfte der Fälle die 1.

Insgesamt gibt es $2^{(2^N)}$ Bit-wertige Funktionen auf der Menge der N -Bit-Folgen.¹ Davon sind zwei Funktionen konstant. Von den balancierten Funktionen gibt es sehr viele (es sind $2^N!/(2^{N-1}!)^2$); die Anzahl ist jedoch nicht relevant. Wichtig ist jedoch, dass man von den 2^N Bit-Folgen mit einem klassischen Algorithmus im ungünstigsten Fall $2^{N-1} + 1$ (also eine Folge mehr als die Hälfte aller Folgen) überprüfen muss, um sicher unterscheiden zu können, ob es sich um eine konstante oder eine balancierte Funktion handelt. Im günstigsten Fall reichen zwar auch zwei Auswertungen, doch für eine eindeutige Unterscheidung müssen als Funktion von N exponentiell viele Fälle untersucht werden. Da mit dem Deutsch-Jozsa-Algorithmus nur eine einzige Auswertung auf einem Quantencomputer notwendig ist, ist der Quantencomputer exponentiell schneller als der klassische Computer.

Im Folgenden betrachten wir zunächst den Fall $N = 1$, d.h. die Bit-wertige Funktion ordnet einem einzelnen Bit wieder ein Bit zu. Von diesen Funktionen gibt es insgesamt vier:

$$\begin{array}{llll} f_1(0) = 0 & f_1(1) = 0 & \text{kurz} & f_1(x) = 0 \\ f_2(0) = 1 & f_2(1) = 1 & \text{kurz} & f_2(x) = 1 \\ f_3(0) = 0 & f_3(1) = 1 & \text{kurz} & f_3(x) = x \\ f_4(0) = 1 & f_4(1) = 0 & \text{kurz} & f_4(x) = 1 - x = \neg x \end{array}$$

Die ersten beiden Funktionen sind die konstanten Funktionen, die anderen beiden Funktionen sind balanciert. Die Aufgabe besteht darin, mit möglichst wenigen Abfragen an eine unbekannte Funktion – dem Oracle – zu entscheiden, ob es sich bei der unbekanntem Funktion um eine konstante Funktion handelt, also um f_1 oder f_2 , oder um eine balancierte Funktion (f_3 oder f_4). Klassisch muss man die unbekannte Funktion auf beiden Bitwerten testen, also zwei Abfragen vornehmen, um diese Unterscheidung treffen zu können.

¹Allgemein gibt es $|B|^{|Ω|}$ Funktionen auf einer Urbildmenge $Ω$ mit $|Ω|$ Elementen und einer Bildmenge B mit $|B|$ Elementen

1.2 Der Deutsch-Algorithmus für $N = 1$

Für den einfachsten Fall $N = 1$ betrachtet man als Ausgangszustand den 2-Qubit-Zustand $|\psi_1\rangle = |0\rangle \otimes |1\rangle$. Dieser Zustand wird in jedem seiner beiden Anteile einer Hadamard-Transformation unterworfen, die wir in diesem Fall durch

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (1.1)$$

darstellen. Wir erhalten somit folgenden Zustand

$$|\psi_1\rangle = |0\rangle \otimes |1\rangle \implies |\psi_2\rangle = H|0\rangle \otimes H|1\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle). \quad (1.2)$$

Diesen Zustand kann man auch in folgender Form schreiben:

$$|\psi_2\rangle = \frac{1}{2} \left(\sum_{x=0}^1 |x\rangle \right) \otimes (|0\rangle - |1\rangle). \quad (1.3)$$

Der nächste Schritt besteht in einer unitärer Implementierung des Oracles, also der unbekanntes Funktion f . Wir definieren für diese Implementierung U_f die Transformation

$$U_f|\psi_2\rangle = |\psi_3\rangle = \frac{1}{2} \left(\sum_{x=0}^1 |x\rangle \right) \otimes (|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle), \quad (1.4)$$

wobei \oplus im Sinne einer Addition modulo 2 (bzw. einer XOR-Operation) zu verstehen ist. Im Anhang 1.5.1 werden die vier unitären Matrizen U_i , die die Funktionen f_i repräsentieren, abgeleitet:

$$U_1 = \mathbf{1} \otimes \mathbf{1} \quad U_2 = \mathbf{1} \otimes \sigma_x \quad U_3 = \text{CNOT} \quad U_4 = (\mathbf{1} \otimes \sigma_x)\text{CNOT} \quad (1.5)$$

Hierbei sind $\mathbf{1}$ die 2×2 -Identitätsmatrix, σ_x die 2×2 Pauli-Matrix,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad (1.6)$$

und CNOT die 4×4 unitäre „controlled NOT“-Matrix, die das zweite Bit unverändert lässt, wenn das erste Bit 0 ist, andernfalls wird das zweite Bit invertiert:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (1.7)$$

In Matrix-Schreibweise folgt für die unitären Implementierungen der vier möglichen Oracles:

$$U_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad U_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad U_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad U_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1.8)$$

Wann immer $f(x) = 0$ ist, ändert sich an dem Zustand nichts. Ist $f(x) = 1$, werden die Rollen von $|0\rangle$ und $|1\rangle$ im zweiten Bit vertauscht, was aber insgesamt nur ein Minuszeichen bewirkt, da

$$|0\rangle - |1\rangle \implies |1\rangle - |0\rangle = -(|0\rangle - |1\rangle). \quad (1.9)$$

Somit können wir für $|\psi_3\rangle$ auch schreiben:

$$|\psi_3\rangle = \frac{1}{2} \left(\sum_{x=0}^1 (-1)^{f(x)} |x\rangle \right) \otimes (|0\rangle - |1\rangle). \quad (1.10)$$

Das zweite Qubit ist nun unabhängig von der Funktion f und wird im Folgenden nicht mehr benötigt, sodass wir es weglassen (einschließlich eines Normierungsfaktors $1/\sqrt{2}$). Zustände, bei denen wir das letzte Qubit weglassen, kennzeichnen wir durch einen ', z.B. $|\psi'_3\rangle$. Das erste Qubit repräsentiert den Zustand

$$|\psi'_3\rangle = \frac{1}{\sqrt{2}} \left((-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \right) = \frac{1}{\sqrt{2}} (-1)^{f(0)} \left(|0\rangle + (-1)^{f(0)+f(1)} |1\rangle \right). \quad (1.11)$$

Hierbei wurde benutzt, dass $f(0) + f(1) = 0$, wir somit immer um diese Summe erweitern können. Nun gilt:

$$f(0) + f(1) = \sigma(f) = \begin{cases} 0 & \text{falls } f \text{ konstant} \\ 1 & \text{falls } f \text{ balanciert} \end{cases} \quad (1.12)$$

Wir erhalten also letztendlich für das erste Qubit den Zustand

$$|\psi'_3\rangle = \begin{cases} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) = |+\rangle & \text{falls } f \text{ eine konstante Funktion ist} \\ \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = |-\rangle & \text{falls } f \text{ balanciert ist} \end{cases}. \quad (1.13)$$

Wenden wir auf diesen Zustand nochmals eine Hadamard-Transformation an, erhalten wir für das erste Qubit den Zustand $|0\rangle$, sofern f eine konstante Funktion ist, und $|1\rangle$, sofern f eine balancierte Funktion ist:

$$|\psi'_3\rangle \implies |\psi'_4\rangle = H|\psi'_3\rangle = \begin{cases} |0\rangle & \text{falls } f \text{ konstant} \\ |1\rangle & \text{falls } f \text{ balanciert} \end{cases} \quad (1.14)$$

Wir müssen also nur das erste Qubit bezüglich der Standardbasis $|0\rangle$ und $|1\rangle$ messen und erhalten als Ergebnis die gesuchte Antwort auf die Frage, ob es sich bei dem Oracle um eine konstante oder balancierte Funktion handelt.

Graphisch drückt man diesen Prozess häufig aus wie in Abb. 1.1.

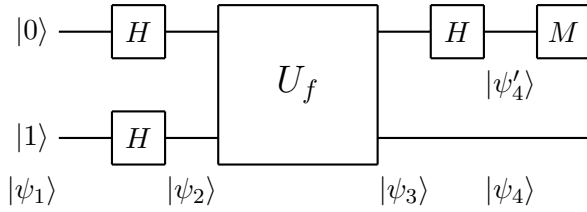


Abbildung 1.1: Graphische Darstellung des Deutsch-Algorithmus für $N = 1$. H bezeichnet Hadamard-Gatter, U_f die Implementierung des Oracles f und M die abschließende Messung.

1.3 Der Deutsch-Jozsa-Algorithmus für $N > 1$

In diesem Abschnitt werden die Verallgemeinerungen behandelt, die bei einem Oracle, das auf einer Folge von N Qubits definiert ist, auftreten. Die unitäre Implementation des Oracles wird allerdings nicht explizit angegeben. Sie ist eine Verallgemeinerung der unitären Operatoren des letzten Abschnitts.

Als Ausgangszustand wählt man nun

$$|\psi_1\rangle = |0\rangle \otimes |0\rangle \otimes \dots \otimes |0\rangle \otimes |1\rangle = |0\rangle^{\otimes N} \otimes |1\rangle, \quad (1.15)$$

wobei $|0\rangle^{\otimes N}$ das N -fache Tensorprodukt des Zustands $|0\rangle$ bezeichnet. In einem ersten Schritt wendet man auf jedes Qubit eine Hadamard-Transformation an:

$$|\psi_2\rangle = H^{\otimes N+1}|\psi_1\rangle = H|0\rangle \otimes \dots \otimes H|0\rangle \otimes H|1\rangle. \quad (1.16)$$

Diesen Zustand kann man auch folgendermaßen schreiben, wenn man das N -fache Tensorprodukt der $H|0\rangle$ -Zustände ausmultipliziert:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{N+1}}} \left(\sum_{x=0}^{2^N-1} |x\rangle \right) \otimes (|0\rangle - |1\rangle). \quad (1.17)$$

Man erkennt an dieser Darstellung deutlich, dass dieser Zustand eine Superposition aus Zuständen zu allen N -Bit-Folgen bzw. ihren Darstellungen durch natürliche Zahlen x ist. Man kann somit alle zu testenden Fälle in einem einzigen Zustand kodieren.

Die explizite unitäre Implementierung der Oracle-Funktionen ist nun etwas komplizierter. Aber das Ergebnis soll folgende Form haben:

$$|\psi_3\rangle = U_f|\psi_2\rangle = \frac{1}{\sqrt{2^{N+1}}} \left(\sum_{x=0}^{2^N-1} |x\rangle \right) \otimes (|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle). \quad (1.18)$$

Wiederum ist offensichtlich, dass sich für den Fall $f(x) = 0$ an dem Zustand überhaupt nichts ändert, und für den Fall $f(x) = 1$ werden im letzten Qubit wiederum $|0\rangle$ und $|1\rangle$ ausgetauscht, was denselben Zustand aber mit einem Minuszeichen liefert. Technisch gesprochen ist $|x\rangle \otimes (|0\rangle - |1\rangle)$ ein Eigenzustand zu allen U_f mit dem Eigenwert $(-1)^{f(x)}$ für $x \in [0, 2^N - 1]$:

$$U_f|x\rangle \otimes (|0\rangle - |1\rangle) = |x\rangle \otimes (|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) = (-1)^{f(x)}|x\rangle \otimes (|0\rangle - |1\rangle). \quad (1.19)$$

Damit können wir $|\psi_3\rangle$ auch in folgender Form schreiben:

$$|\psi_3\rangle = \frac{1}{\sqrt{2^{N+1}}} \left(\sum_{x=0}^{2^N-1} (-1)^{f(x)} |x\rangle \right) \otimes (|0\rangle - |1\rangle). \quad (1.20)$$

Das letzte Qubit ist von dem Oracle unabhängig und kann weggelassen werden. Für die ersten N Qubits gilt: Ist $f(x) = f$ eine konstante Funktion, ändert sich der Zustand nicht, der Zustandsvektor wird lediglich mit $(-1)^f$ multipliziert:

$$|\psi'_3\rangle = \frac{1}{\sqrt{2^N}} (-1)^f \left(\sum_{x=0}^{2^N-1} |x\rangle \right) \quad (f \text{ konstant}). \quad (1.21)$$

Durch eine abschließende Hadamard-Transformation auf jedem Qubit wird aus diesem Zustand bis auf den Faktor $(-1)^f$ wieder der Ausgangszustand:

$$|\psi'_4\rangle = H^{\otimes N} |\psi'_3\rangle = (-1)^f |0\rangle^{\otimes N} \quad (1.22)$$

Wir zeigen nun allgemeiner die folgende Aussage:

$$\langle \psi'_4 | 0 \rangle^{\otimes N} = \begin{cases} (-1)^f & \text{falls } f \text{ konstant} \\ 0 & \text{falls } f \text{ balanciert} \end{cases}. \quad (1.23)$$

Daraus folgt: Die Wahrscheinlichkeit $w(|0\rangle^{\otimes N})$, am Ausgang unseres Quantencomputers den Zustand $|0\rangle^{\otimes N}$ zu messen – alle N Messgeräte an den jeweiligen Ausgängen messen den Zustand $|0\rangle$ – ist gegeben durch:

$$w(|0\rangle^{\otimes N}) = \begin{cases} 1 & \text{falls } f \text{ konstant} \\ 0 & \text{falls } f \text{ balanciert} \end{cases}. \quad (1.24)$$

Zum Beweis betrachten wir das Skalarprodukt von $|\psi'_4\rangle = H^{\otimes N} |\psi'_3\rangle$ mit dem Zustand $|0\rangle^{\otimes N}$, wenden die Hadamard-Transformation nun aber auf $|0\rangle^{\otimes N}$ an. Dies liefert wie schon in Gl. 1.17 eine Summe über alle natürlichen Zahlen von 0 bis $2^N - 1$ im Binärcode:

$$\langle \psi'_4 | 0 \rangle^{\otimes N} = \frac{1}{\sqrt{2^N}} \sum_{x=0}^{2^N-1} (-1)^{f(x)} \langle x | H^{\otimes N} | 0 \rangle^{\otimes N} = \sum_{x,y=0}^{2^N-1} (-1)^{f(x)} \langle x | y \rangle. \quad (1.25)$$

Da Zustände zu verschiedenen Bitfolgen orthogonal sind, gilt $\langle x | y \rangle = \delta_{xy}$, wir erhalten also nur einen Beitrag für $x = y$:

$$\langle \psi'_4 | 0 \rangle^{\otimes N} = \frac{1}{2^N} \sum_{x=0}^{2^N-1} (-1)^{f(x)}. \quad (1.26)$$

Ist f konstant, können wir den Faktor $(-1)^f$ vor die Summe ziehen und erhalten als Ergebnis 1. Ist f balanciert, erhalten wir eine Summe von $+1$ und -1 , wobei beide Zahlen gleich häufig auftreten. Insgesamt ist in diesem Fall das Ergebnis daher 0. Damit ist unsere obige Behauptung bewiesen. Zur graphischen Darstellung des Deutsch-Jozsa-Algorithmus für beliebige N , siehe Abb. 1.2.

Man kann daher mit diesem Algorithmus mit einer einzigen Abfrage des Oracles die Information erhalten, ob die unbekannt Funktion konstant oder balanciert ist.

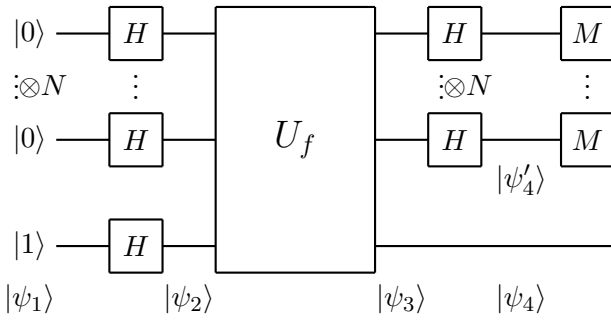


Abbildung 1.2: Graphische Darstellung des Deutsch-Algorithmus für $N > 1$. Die oberen N Qubits kodieren nach dem Hadamard-Gatter sämtliche N -Bit-Folgen. U_f bezeichnet wieder die Implementierung des Oracles und M die abschließenden Messungen an jedem Qubit.

1.4 Das Penny-Flip-Spiel

Das Penny-Flip-Spiel von David Meyer ist eine vereinfachte Version des Deutsch-Algorithmus für $N = 1$, bei dem die Quantenversion eine Gewinnstrategie bei einem einfachen Spiel ermöglicht, die klassisch nicht möglich ist [4, 5].

Die klassischen Spielregeln sind folgende: Alice darf eine Münze auf den Tisch legen, so dass „Kopf“ oder „Zahl“ oben liegen. Anschließend darf Bob die Münze von Alice entweder umdrehen oder so liegen lassen, wie Alice sie hingelegt hat. Alice sieht dabei nicht, was Bob getan hat. Alice darf nun die Münze (ohne diese anzuschauen oder abzutasten, d.h., sie weiß nicht, ob Bob die Münze gedreht hat oder nicht) nochmals drehen oder auch belassen, wie sie ist. Alice hat gewonnen, wenn die Münze abschließend „Kopf“ zeigt, Bob hat gewonnen, wenn die Münze „Zahl“ zeigt. Gibt es für Alice eine Gewinnstrategie?

Offensichtlich gibt es bei dieser klassischen Version des Spiels für Alice keine Gewinnstrategie, da sie nicht wissen kann, ob Bob die Münze gedreht hat oder nicht. In einer Quantenversion dieses Spiels hat Alice jedoch eine 100%-ige Gewinnchance. In der Quantenversion kann Alice die Münze in einem Superpositionszustand aus „Kopf“ und „Zahl“ präparieren. Sie kann beispielsweise die Münze in dem Zustand

$$|\psi_2\rangle = H|\text{Kopf}\rangle = \frac{1}{\sqrt{2}}(|\text{Kopf}\rangle + |\text{Zahl}\rangle) \tag{1.27}$$

präparieren, wobei H wieder ein Hadamard-Gatter für den Zustand der Münze darstellt.

Bob darf nun eine von zwei Transformationen durchführen: Entweder er lässt die Münze unverändert, also

$$|\text{Zahl}\rangle \implies |\text{Zahl}\rangle \quad \text{und} \quad |\text{Kopf}\rangle \implies |\text{Kopf}\rangle, \tag{1.28}$$

oder er dreht die Münze:

$$|\text{Zahl}\rangle \implies |\text{Kopf}\rangle \quad \text{und} \quad |\text{Kopf}\rangle \implies |\text{Zahl}\rangle. \tag{1.29}$$

Hier wird ausgenutzt, dass der von Alice präparierte Superpositionszustand ein Eigenzustand sowohl zur Identitätstransformation als auch zur Vertauschungsoperation (Pauli-X) ist:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{1.30}$$

Egal für welche der beiden Operationen sich Bob entscheidet, der Zustand wird durch seine Operation nicht verändert. Abschließend kann Alice nochmals eine Hadamard-Transformation auf den Zustand anwenden und erhält mit Sicherheit den Zustand $|\text{Kopf}\rangle$, unabhängig davon, was Bob gemacht hat. Insofern hat Alice immer gewonnen.

1.5 Anhang

1.5.1 Die unitäre Implementation des Oracles

In diesem Anhang soll gezeigt werden, wie man zu den Oracle-Funktionen die unitären Matrizen konstruiert und wie diese durch Quantengatter realisiert werden können.

Die Funktion f soll durch folgende Vorschrift realisiert werden:

$$f : |x\rangle \otimes |y\rangle \longrightarrow |x\rangle \otimes |y \oplus f(x)\rangle, \quad (1.31)$$

wobei $x, y \in \{0,1\}$ und die \oplus -Operation als „Addition modulo 2“ oder XOR-Operation zu verstehen sind. Als 4-Vektor wird $|x\rangle \oplus |y\rangle$ folgendermaßen repräsentiert:

$$|0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad |0\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad (1.32)$$

Dies folgt der allgemeinen Regeln:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \otimes \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 y_1 \\ x_1 y_2 \\ x_2 y_1 \\ x_2 y_2 \end{pmatrix} \quad (1.33)$$

Wir konstruieren nun die Darstellungen der vier Oracles (zu $N = 1$). Man beachte dabei, dass sich das erste Qubit nie ändert, sondern nur das zweite Qubit – eventuell in Abhängigkeit von dem Zustand des ersten Qubits. Der Einfachheit schreibe ich im Folgenden $|x, y\rangle$ statt $|x\rangle \otimes |y\rangle$.

f_1

Die Funktion f_1 hat immer den Wert $f_1(x) = 0$ und ändert den Zustand daher nicht. Sie wird durch die 4×4 -Identitätsmatrix repräsentiert:

$$U_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{1} \otimes \mathbf{1}. \quad (1.34)$$

f_2

Die Funktion f_2 hat immer den Wert $f_2(x) = 1$ und tauscht daher im zweiten Qubit die Werte von 0 und 1 aus. Das bedeutet:

$$|0,0\rangle \mapsto |0,1\rangle \quad |0,1\rangle \mapsto |0,0\rangle \quad |1,0\rangle \mapsto |1,1\rangle \quad |1,1\rangle \mapsto |1,0\rangle. \quad (1.35)$$

Die Matrix zu dieser Abbildung ist

$$U_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \mathbf{1} \otimes \sigma_x. \quad (1.36)$$

f_3

Die Abbildung $f_3(x) = x$ lässt das zweite Qubit unverändert, wenn $x = 0$ ist, und ersetzt es durch seine Negation, wenn $x = 1$ ist. Das bedeutet für die Standardbasisvektoren:

$$|0,0\rangle \mapsto |0,0\rangle \quad |0,1\rangle \mapsto |0,1\rangle \quad |1,0\rangle \mapsto |1,1\rangle \quad |1,1\rangle \mapsto |1,0\rangle. \quad (1.37)$$

Diese Transformation wird durch folgende Matrix repräsentiert:

$$U_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \text{CNOT}. \quad (1.38)$$

Man bezeichnet diese Matrix als „Controlled NOT“, da die NOT-Operation auf dem zweiten Qubit durch das erste Qubit kontrolliert wird.

1.5.2 f_4

Die vierte Abbildung $f_4(x) = \neg x$ ändert das zweite Qubit, sofern das erste sich im Zustand $|0\rangle$ befindet. Es handelt sich daher bei der Implementierung ebenfalls um ein CNOT-Gatter, allerdings ist der Control-Parameter nun vertauscht. Für die Basis bedeutet dies:

$$|0,0\rangle \mapsto |0,1\rangle \quad |0,1\rangle \mapsto |0,0\rangle \quad |1,0\rangle \mapsto |1,0\rangle \quad |1,1\rangle \mapsto |1,1\rangle. \quad (1.39)$$

Die zugehörige Transformationsmatrix ist:

$$U_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = (\mathbf{1} \otimes \sigma_3)\text{CNOT}. \quad (1.40)$$

Alle vier Oracles lassen sich somit durch geeignete Hintereinanderschaltung der beiden Gatter Pauli-X und CNOT darstellen.

Literaturverzeichnis

- [1] Deutsch, David (1985); *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*; Proceedings of the Royal Society of London A. 400 (1818): 97–117.
- [2] Deutsch, David, Jozsa, Richard (1992); *Rapid solutions of problems by quantum computation*. Proceedings of the Royal Society of London A. 439 (1907): 553–558.
- [3] Cleve, R., Ekert, A, Macchiavello, C., Mosca, M, (1998); *Quantum algorithms revisited*; Proceedings of the Royal Society of London A. 454 (1969): 339–354.
- [4] Meyer, David A. (1999); *Quantum Strategies*; Phys. Rev. Lett. 82, 1052
- [5] Müller, Rainer; Greinert, Franziska (2023); *Quantum Penny Flip – Spielen mit dem Quantencomputer*; plusLucis 2/2023 Quantenphysik. *Playing with a Quantum Computer*, arXiv: 2108.06271.