LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

MNM TEAM

Video-Hausarbeit

Einführung in Quantencomputing

Adrian Uffmann[1]

adrian.uffmann@campus.lmu.de

LMU München

Wednesday 5th August, 2020

Aufgabe

Geben Sie einen QuBit-Zustand zur Basis $\{|0\rangle, |1\rangle\}$ an, bei welchem die Wahrscheinlichkeit $|+\rangle$ zu messen 20% beträgt.

Zustand in Hadamard Basis

Beispiel für solch einen Zustand in der Basis $\{|+\rangle, |-\rangle\}$ (Hadamard Basis).

$$|\Psi\rangle = \hat{\alpha} \cdot |+\rangle + \hat{\beta} \cdot |-\rangle$$

$$\hat{\alpha}^2 = 20\% = 0,2 \Rightarrow \hat{\alpha} = \sqrt{0,2} = \frac{1}{\sqrt{5}}$$

$$|\hat{\beta}|^2 = 1 - |\hat{\alpha}|^2 \Rightarrow |\hat{\beta}| = \pm\sqrt{1 - |\hat{\alpha}|^2} = \pm\sqrt{1 - |\frac{1}{\sqrt{5}}|^2} = \pm\sqrt{1 - \frac{1}{5}} = \pm\frac{2}{\sqrt{5}}$$

$\Rightarrow$ Ein Beispiel für $\hat{\beta}$ ist $\frac{2}{\sqrt{5}}$

$\Rightarrow$ Ein Beispiel für Ψ ist $\frac{1}{\sqrt{5}}|+\rangle + \frac{2}{\sqrt{5}}|-\rangle$

Zustand in normaler Basis

Für $|\Phi\rangle = \frac{1}{\sqrt{5}}|+\rangle + \frac{2}{\sqrt{5}}|-\rangle$:

$$
\begin{aligned}
H|\Phi\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \cdot \begin{pmatrix}\frac{1}{\sqrt{5}}\\ \frac{2}{\sqrt{5}}\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{5}} + \frac{1}{\sqrt{2}}\cdot\frac{2}{\sqrt{5}}\\ \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{5}} - \frac{1}{\sqrt{2}}\cdot\frac{2}{\sqrt{5}}\end{pmatrix} = \begin{pmatrix}\frac{3}{\sqrt{10}}\\ -\frac{1}{\sqrt{10}}\end{pmatrix}\\
&= \frac{3}{\sqrt{10}}|+\rangle - \frac{1}{\sqrt{10}}|-\rangle\\
\Rightarrow |\Phi\rangle &= \frac{3}{\sqrt{10}}|0\rangle - \frac{1}{\sqrt{10}}|1\rangle
\end{aligned}
$$

Aufgabe

Diskutieren Sie folgende Aussage: *"Quantencomputer sind einfach schnellere Rechner, die Probleme beliebiger Komplexität im Handumdrehen lösen."*

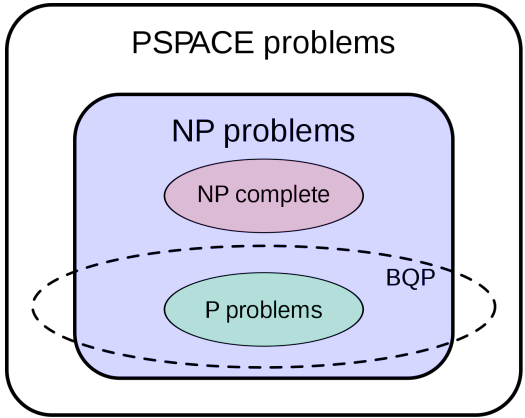
Aufgabe 1b

Das stimmt nicht.

Quantencomputer können alles berechnen, was normale Computer auch berechnen können, **aber** sie sind nicht bei jedem Problem schneller.

Für Quantenalgorithmen gibt es eine eigene Komplexitätsklasse.

BQP (*bounded error quantum polynomial time*)

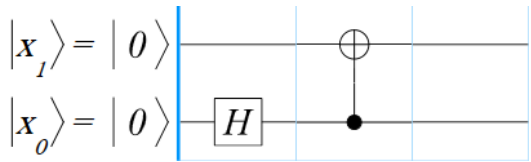


Aufgabe

Welche Rolle spielen die vier nach Bell benannten Zustände von 2-QuBit- Registern im Bereich Quantencomputing?

Bell Zustände

- Die Bell-Zustände sind verschränkte Zustände.
- lassen sich mit einem sehr einfachen Schaltkreis erzeugen.



Die vier Bell Zustände

- $|\beta_{00}\rangle = |\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\beta_{10}\rangle = |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\beta_{01}\rangle = |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\beta_{11}\rangle = |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

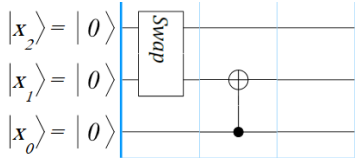
Aufgabe

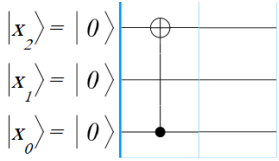
Beschreiben Sie das SWAP-Gatter. Warum spielt es auf den aktuell verfügbaren Quantencomputern (z.B. IBMq) eine besondere Rolle?

SWAP-Gatter logisch

- SWAP-Gatter tauschen die Zustände zweier QuBits.

$\Rightarrow$ SWAP-Gatter sind logisch gesehen nie nötig, denn man könnte auch mit nicht-vertauschten QUBits weiter rechnen.

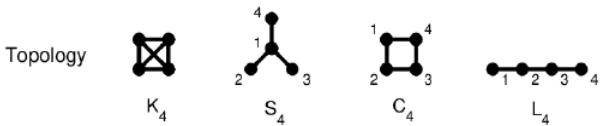




SWAP-Gatter physikalisch

Echte Quantencomputer ...

- ... können kontrollierte Operationen nicht zwischen beliebigen QuBits machen.
- ... haben eine Netzwerktopologie.



Beispiel für Netzwerktopologien

Aufgabe

Sie sind sich sicher, dass nur einer der folgenden Quantenalgorithmen zur Lösung des Problems in Frage kommt.

- Grover's Algorithmus
- Shor's Algorithmus
- Quanten Fourier Transformation
- Simon's Algorithmus

Gehen Sie auf jeden dieser Algorithmen kurz ein und wählen Sie einen zur Lösung des genannten Problems aus. Erklären Sie dabei insbesondere auch, warum die übrigen Algorithmen nicht in Frage kommen.

Grover's Algorithmus

- untersucht Funktionen der Form $f : \{0,1\}^n \to \{0,1\}$.
- Ziel ist es, eine Eingabe zu finden, bei der die Funktion eine Ausgabe von 1 liefert.
- Funktioniert besser, wenn es weniger mögliche Ergebnisse gibt.
- Eignet sich für unser Problem, wenn wir eine Funktion finden, die nur für den kürzesten Pfad 1 liefert.

Shor's Algorithmus

- Kann effizient Zahlen faktorisieren.
- Eignet sich um RSA zu brechen.
- Eignet sich nicht für unser Problem.

Quanten Fourier Transformation

- die Quanten-Variante der DFT (*diskrete Fourier-Transformation*).
- Eignet sich um die Periode einer Funktion zu finden.
- Eignet sich nicht für unser Problem.

Simon's Algorithmus

- untersucht Funktionen der Form $f : \{0,1\}^n \to \{0,1\}^n$.
- Ziel ist es herauszufinden, ob die Funktion bijektiv ist, oder
- ob es einen Vektor s gibt, sodass $f(x) = f(\hat{x}) \Leftrightarrow x \oplus s = \hat{x}$.
  ($\oplus$ steht hier für element-weises XOR)
- Beispiel: Wenn $f(001_2) = f(111_2)$ ist der gesuchte Vektor
  $s = 110_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$, denn $001_2 \oplus 110_2 = \begin{pmatrix} 1 \oplus 0 \\ 0 \oplus 1 \\ 0 \oplus 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 111_2$
- Eignet sich nicht für unser Problem.

Aufgabe

Überlegen und erklären Sie, wie Sie den oben abgebildeten Graphen und den zugehörigen kürzesten Weg (bzw. die kürzesten Wege) in eine Funktion f abbilden können, die zu Ihrem gewählten Algorithmus passt. Im oben abgebildeten Labyrinth ist die Lösung $(S \to a \to Z)$ durch scharfes Hinsehen ersichtlich; Sie dürfen diese verwenden.

Überlegung: Optimale Funktion

- Mit Grover's Algorithmus erhalten wir für eine Funktion der Form $f : \{0,1\}^n \to \{0,1\}$ eine Eingabe, die eine Ausgabe von 1 erzeugt.

$\Rightarrow$ Optimal wäre eine Funktion, der man den Pfad als Eingabe gibt und die 1 liefert, genau dann, wenn es der kürzeste Pfad von S zu Z ist.

- nur leicht zu implementieren, wenn kürzester Pfad bereits bekannt.
- Dazu mehr in Aufgabe 2f.

Modellierung der Eingabe

Die Eingabe muss einen Pfad repräsentieren. Hier gibt es verschiedene Varianten:

1. Jedes Bit steht für einen Knoten (gesetzt bedeutet Teil des Pfades).
   $\Rightarrow$ Pfad nicht eindeutig, Knoten-Reihenfolge nicht klar.
2. Jedes Bit steht für eine Kante (gesetzt bedeutet Teil des Pfades).
   $\Rightarrow$ Pfad schwer zu validieren, da Kanten-Reihenfolge implizit.
3. Mehrere Bits kodieren einen Knoten, mehrere kodierte Knoten ergeben den Pfad.
   $\Rightarrow$ Leicht zu validieren, da Knote-Reihenfolge Teil der Eingabe.
4. Mehrere Bits kodieren eine Kante, mehrere kodierte Kanten ergeben den Pfad.
   $\Rightarrow$ Wie 3., aber es gibt mehr Kanten als Knoten, daher ineffizienter.

Modellierung der Eingabe

Wir verwenden Variante 3.

Bei vier Knoten $\{a, b, c, d\}$ reichten 2 Bit um einen Knoten zu kodieren:
$a = |00\rangle$, $b = |01\rangle$, $c = |10\rangle$ und $d = |11\rangle$.

Wir schreiben Knoten, die weiter vorne im Pfad sind auf die QBits mit niedrigerem Index (also weiter rechts).

Z. B. wird der Pfad $b \to c \to a$ als $|001001\rangle$ dargestellt.

Modellierung von Startknoten und Zielknoten

- Ein valider Pfad muss immer bei S beginnen und bei Z enden.

$\Rightarrow$ Wir müssen nur den Teilpfad zwischen S und Z betrachten.

- Es macht keinen Sinn, wenn S oder Z an anderer Stelle im Pfad auftauchen.

$\Rightarrow$ Wir brauchen keine Zahlen um S und Z kodieren zu können.

Funktionsdefinition

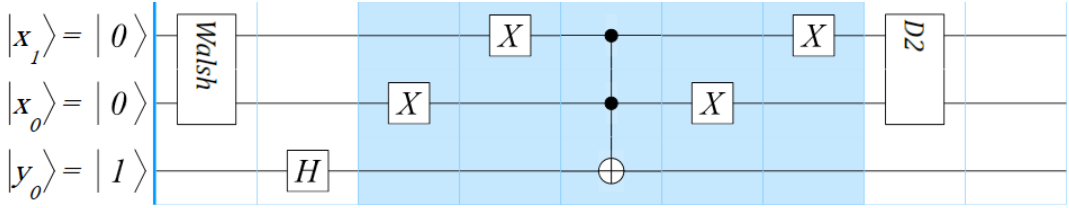
- Der Graph hat nur einen kürzesten Pfad: $S \to a \to Z$.
- Dieser wird als $|00\rangle$ dargestellt.

$$\Rightarrow f(x) = \begin{cases} 1 & x = |00\rangle \\ 0 & \text{otherwise.} \end{cases}$$

Aufgabe

Erstellen Sie den Schaltkreis für die Analyse des obigen Problems inklusive der Orakelfunktion Uf. Erläutern Sie, wie man aus Ihrer Funktion f das Gatter Uf konstruiert oder errechnet.

Schaltkreis



U_f ist blau hinterlegt

Als Erinnerung $f(x) = \begin{cases} 1 & x = |00\rangle \\ 0 & \text{otherwise.} \end{cases}$

Aufgabe

Diskutieren Sie Ihren Schaltkreis und das Ergebnis.

Zustände

Zustand vor U_f:

Value	Qubits	Probability	Amplitude
|0>	|000>	P = 0.125	0.35 + 0.00i
|1>	|001>	P = 0.125	-0.35 + 0.00i
|2>	|010>	P = 0.125	0.35 + 0.00i
|3>	|011>	P = 0.125	-0.35 + 0.00i
|4>	|100>	P = 0.125	0.35 + 0.00i
|5>	|101>	P = 0.125	-0.35 + 0.00i
|6>	|110>	P = 0.125	0.35 + 0.00i
|7>	|111>	P = 0.125	-0.35 + 0.00i

Zustand nach U_f:

Value	Qubits	Probability	Amplitude
|0>	|000>	P = 0.125	-0.35 + 0.00i
|1>	|001>	P = 0.125	0.35 + 0.00i
|2>	|010>	P = 0.125	0.35 + 0.00i
|3>	|011>	P = 0.125	-0.35 + 0.00i
|4>	|100>	P = 0.125	0.35 + 0.00i
|5>	|101>	P = 0.125	-0.35 + 0.00i
|6>	|110>	P = 0.125	0.35 + 0.00i
|7>	|111>	P = 0.125	-0.35 + 0.00i

Ergebniszustand:

Value	Qubits	Probability	Amplitude
|0>	|000>	P = 0.5	-0.71 + 0.00i
|1>	|001>	P = 0.5	0.71 + 0.00i

Aufgabe

Das Ziel ist vom Start in 4 Schritten erreichbar. Wie viele kürzeste Wege gibt es? *(Hinweis: Es sind maximal 8.)* Erläutern Sie, wie Sie Ihre Funktion f für diesen Graphen abändern müssen und welchen Zustand Sie in den Registern des angepassten Schaltkreises nach der Anwendung des angepassten U_f erwarten.

Kürzeste Wege

Es gibt 7 kürzeste Wege (alphabetisch sortiert):

1. $S \to i \to a \to d \to Z$
2. $S \to i \to a \to e \to Z$
3. $S \to i \to a \to g \to Z$
4. $S \to i \to h \to d \to Z$
5. $S \to l \to c \to d \to Z$
6. $S \to l \to c \to j \to Z$
7. $S \to l \to h \to d \to Z$

Abändern der Funktion

Dieser Graph hat 2 Unterschiede:

1. Es gibt mehr Knoten: 12 und S und Z.

$\Rightarrow$ Es werden nun $ceil(log_2(12)) = 4$, statt 2 Bits pro Knoten benötigt.

2. Es gibt mehr kürzeste Pfade.

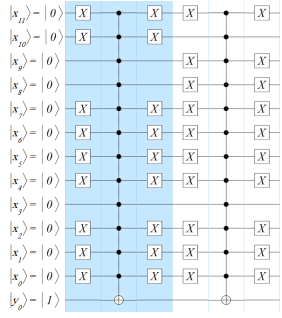
$\Rightarrow$ Bei allen muss die Funktion 1 liefern.

Abändern des Schaltkreises

Eingabedarstellung der zwei ersten kürzesten Pfade:

1. $S \to i \to a \to d \to Z \equiv |001100001000\rangle$
2. $S \to i \to a \to e \to Z \equiv |010000001000\rangle$

Die Schaltkreise für die kürzesten Pfade können einfach hintereinander ausgeführt werden.
Aufeinander folgende X-Gatter können weggelassen werden.



Teile des Orakels: blau für Pfad 1, weiß für Pfad 2.

Zustand nach U_f

- Ähnlich wie in 2d.
- Das angepasste Orakel wird bei 7 statt nur einem Pfad aktiv.

$\Rightarrow$ Es sollten bei 14 statt nur bei 2 Zuständen der Superposition die Vorzeichen negiert worden sein.

Value	Qubits	Probability	Amplitude
|0>	|000>	P = 0.125	-0.35 + 0.00i
|1>	|001>	P = 0.125	0.35 + 0.00i
|2>	|010>	P = 0.125	0.35 + 0.00i
|3>	|011>	P = 0.125	-0.35 + 0.00i
|4>	|100>	P = 0.125	0.35 + 0.00i
|5>	|101>	P = 0.125	-0.35 + 0.00i
|6>	|110>	P = 0.125	0.35 + 0.00i
|7>	|111>	P = 0.125	-0.35 + 0.00i

Zustand nach U_f aus Aufgabe 2d

Aufgabe

Skizzieren Sie, wie man die Orakelfunktion konstruieren könnte, wenn der kürzeste Weg bzw. die kürzesten Wege nicht offensichtlich sind. Erläutern Sie außerdem das Vorgehen, wenn auch dessen Länge unbekannt ist.

Hilfreiche Überlegungen:

- *Wie modellieren Sie Knoten und Kanten des Graphen?*
- *Wie modellieren Sie "gültige" Wege im Graphen?*
- *Wie modellieren Sie Länge eines Weges?*
- *Wie können Sie ohne das Wissen über den kürzesten Weg dessen Länge ermitteln?*

Verallgemeinerung des Schaltkreises

- Bisherige Annahme: Funktion liefert 1, genau dann, wenn Eingabe der kürzeste Pfad von S zu Z.
- **Aber:** Im Allgemeinen zu schwer, man kann nicht leicht prüfen, ob kürzerer Pfad existiert.
- Wie beim Max-Cut-Problem reicht es, wenn die Funktion nur für valide Pfade mit bestimmter Länge eine 1 liefert.

$\Rightarrow$ Dann kann mit binärer Suche die kürzeste Länge bestimmt werden.

Abbilden des Problems in eine analysierbare Funktion

- Unsere Funktion soll nur für valide Pfade mit bestimmter Länge eine 1 liefern.
- Die Pfadlänge ist durch die Länge der Eingabe bereits festgelegt.

$\Rightarrow$ Die Funktion muss daher nur noch den Pfad validieren.

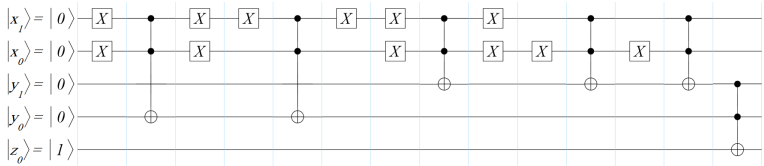
Welche Eigenschaften hat ein valider Pfad?

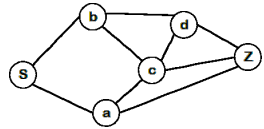
1. Der erste Pfad-Knoten muss durch eine Kante mit S verbunden sein.
2. Für aufeinander folgende Knoten muss der Graph eine Kante enthalten.
3. Der letzte Pfad-Knoten muss durch eine Kante mit Z verbunden sein.

- Die Eigenschaften lassen sich leicht prüfen, indem für jedes aufeinander folgende Knotenpaar geprüft wird, ob es so eine Kante gibt.
- Wenn ja wird der Wert in einem Ancilla QuBit gespeichert.
- Die Ancilla QBits können dann mit einem mehrfachen Toffoli verundet werden.

$\Rightarrow$ Es werden so viele Ancilla QuBits benötigt, wie der Pfad Kanten hat.

Allgemeines Orakel für Problem aus 2b





- y_0: gibt es eine Kante zwischen S und dem ersten (einzigen) Knoten des Pfades.
- y_1: gibt es eine Kante zwischen dem letzten (einzigen) Knoten des Pfades und Z.
- Anschließend werden die Ancilla QuBits wieder entschränkt, indem die Operationen (außer das letzte Toffoli) noch einmal rückwärts ausgeführt werden.

Aufgabe

Diskutieren Sie die Laufzeit Ihres Algorithmus (inkl. klassischem Teil, sofern relevant). Diskutieren Sie, ob Sie auf einem aktuellen Quantenchip (z.B. IBMq) einen Quantenvorteil herausholen können.

Laufzeit

Die Laufzeit hängt von der Anzahl der Knoten V, der Anzahl der Kanten E und der Länge des zu untersuchenden Pfades $|P|$ ab.

Das Orakel aus 2f benötigt für jede Kante des Pfades eine zu der Anzahl der Knoten logarithmische Anzahl an Operationen pro Kante des Graphes. Das Orakel hat also eine Komplexität von $O(|P| \cdot |E| \cdot log(|V|))$.

Dies muss man noch multiplizieren mit der Anzahl der Aufrufe des Orakels.

Diese hängt logarithmisch von der Größe des Suchraumes der binären Suche und von der optimalen Anzahl von Grover-Iterationen ab.

Laufzeit

Die binäre Suche liegt in $O(log(|V|))$, da der kürzeste Pfad höchstens alle Knoten besucht.

Das Register x, auf dem Grover's Algorithmus agiert hat eine Größe n in $O(|P| \cdot log(|V|))$, da $O(|P|)$ Knoten in der Eingabe sind und jeder $O(log(|V|))$ QuBits für die Kodierung braucht.

Die optimale Anzahl an Grover-Iterationen bei M mögichen Messergebnissen liegt in $O(\sqrt{\frac{2^n}{M}})$.

Es ist also nicht offensichtlich, ob der Algorithmus den klassischen Algorithmus von Dijkstra mit $O(|E| + |V|log(|V|))$ schlagen kann.

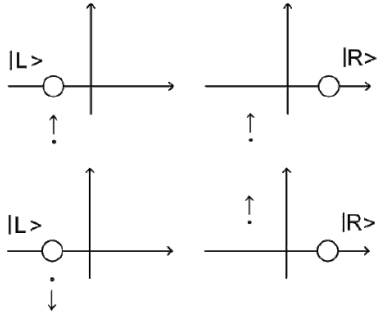
Aufgabe

Erläutern Sie anhand eines Beispiel aus der Vorlesung den Begriff Dekohärenz. Welche Maßnahmen werden bei heutigen Quantenchips eingesetzt, um Dekohärenz zu vermeiden?

Dekohärenz

- ist das Phänomen, dass QuBits mit der Umgebung interagieren können.

Beispiel aus der Vorlesung:



$$\frac{1}{\sqrt{2}}(|L\rangle + |R\rangle)\,|\uparrow\rangle = \frac{1}{\sqrt{2}}(|L\rangle\,|\uparrow\rangle + |R\rangle\,|\uparrow\rangle)$$

$$\xrightarrow{\text{Interaktion}}$$

$$\frac{1}{\sqrt{2}}(|L\rangle\,|\downarrow\rangle + |R\rangle\,|\uparrow\rangle)$$

Dekohärenzgleichung

$$
\begin{aligned}
(\alpha\,|0\rangle + \beta\,|e\rangle \xrightarrow{\text{Interaktion}} \alpha(|0\rangle\,|e_{00}\rangle + |1\rangle\,|e_{01}\rangle) + \beta(|0\rangle\,|e_{10}\rangle + |1\rangle\,|e_{11}\rangle) \\
= (\alpha\,|0\rangle + \beta\,|1\rangle) \cdot \frac{1}{2}(|e_{00}\rangle + |e_{11}\rangle) \\
+ (\alpha\,|0\rangle - \beta\,|1\rangle) \cdot \frac{1}{2}(|e_{00}\rangle - |e_{11}\rangle) \\
+ (\alpha\,|1\rangle + \beta\,|0\rangle) \cdot \frac{1}{2}(|e_{01}\rangle + |e_{10}\rangle) \\
+ (\alpha\,|1\rangle - \beta\,|0\rangle) \cdot \frac{1}{2}(|e_{01}\rangle - |e_{10}\rangle)
\end{aligned}
$$

Dekohärenz Vermeidung

Wenn man weiß, welche Art von Fehler aufgetreten ist, kann man die Kohärenz (meist) wieder herstellen.

In der Praxis schwer, da solche Fehler überall passieren können.

Um Dekohärenz zu vermeiden werden Quantencomputer stark gekühlt und von äußeren Einflüssen isoliert.

Aufgabe

Welche Gegebenheiten/Einschränkungen machen die Fehlerkorrektur in einem Quantensystem schwieriger als in einem klassischen System?

Warum ist Fehlerkorrektur in Quantencomputern schwieriger?

- No-Cloning Theorem.
- Viel mehr Arten von Fehlern.
- In einem diskreten System können sich Fehler leichter aufaddieren.

Aufgabe

Erklären Sie den Unterschied zwischen den beiden Fehlerarten *Bitflip* und *Phaseflip* und wie Sie selbige korrigieren können.

Bitflip vs. *Phaseflip*

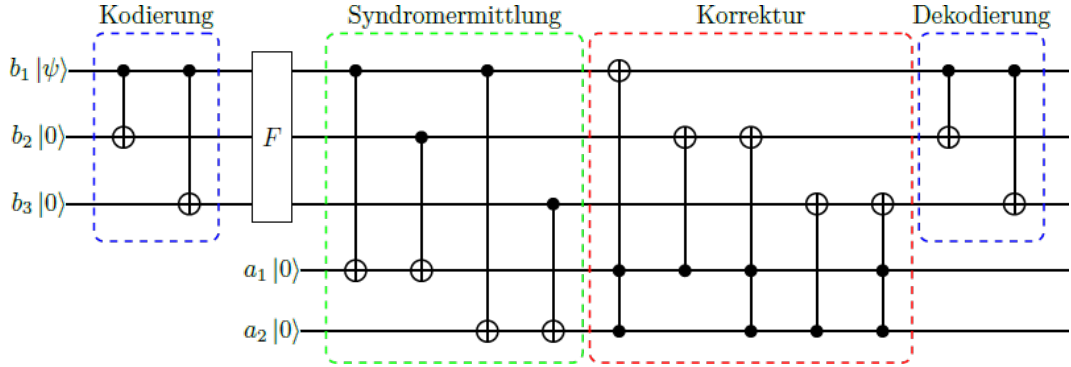
- *Bitflip*
  - kennt man auch aus der klassischen Informatik.
  - an einer Stelle, an der eine 0 sein sollte, ist eine 1 (oder anders herum).
  - entspricht der Anwendung eines X-Gatters.
- *Phaseflip*
  - Die Amplitude von $|1\rangle$ hat das falsche Vorzeichen (Minus statt Plus, oder anders herum).
  - entspricht der Anwendung eines Z-Gatters.

Da sowohl das X-Gatter, als auch das Z-Gatter ihre eigenen Inversen sind, können diese Gatter die Fehler auch wieder beheben.

Aufgabe

Vergleichen Sie die Mechanismen zur Syndromermittlung und Korrektur von *Bitflip* und *Phasenflip* mit der *Dichten Kodierung*. Überlegen Sie, wie Sie auch mit diesen Mechanismen klassische Informationen im Zustand eines QuBits bzw. Quantenregisters speichern oder übertragen können. Diskutieren Sie, ob Ihnen das einen Vorteil gegenüber einem klassischen Speichermedium oder Informationskanal verschafft.

Syndromermittlung und Korrektur von *Bitflips*



Dichte Kodierung