Introduction to Quantum Computing

Lecture notes for the summer term 2024

Jannik Hellenkamp Dominique Unruh

2024-07-27

Table of contents

Welcome **2**

Changelog . . . 2

1 Introduction **3**

1.1 Double slit experiment . . . 3

1.2 What is a quantum computer? . . . 4

2 Probabilistic systems **5**

2.1 Deterministic possibilities . . . 5

2.2 Probability distribution . . . 5

2.3 Probabilistic processes . . . 6

Applying a probabilistic process . . . 7

3 Quantum systems **8**

3.1 Quantum states . . . 8

3.2 Unitary transformation . . . 9

4 Observing probabilistic and measuring quantum systems **10**

4.1 Observing a probabilistic system . . . 10

4.2 Measuring a quantum system . . . 12

4.3 Elitzur–Vaidman bomb tester . . . 13

5 Partial observing and measuring systems **14**

5.1 Partially observing a probabilistic system . . . 14

5.2 Partially measuring a quantum system . . . 15

6 Composite Systems **15**

6.1 Constructing composite systems . . . 16

6.2 Measuring composite systems . . . 17

7 Quantum Circuits **18**

7.1 Visual language . . . 18

7.2 Important gates . . . 18

7.2.1 Single qubit gates . . . 19

7.2.2 Controlled-NOT gate . . . 19

7.3 Ket Notation . . . 20

8 Bernstein-Vazirani Algorithm **21**

9 Shor's Algorithm **25**

9.1 Discrete Fourier Transformation . . . 25

9.2 Reducing factoring to period finding 26
9.3 The quantum algorithm for period finding 27
9.4 Post processing 29
9.5 Constructing the DFT 30

10 Grover's algorithm 34
10.1 Preparations 34
10.1.1 Constructing the oracle V_f 35
10.1.2 Constructing FLIP_* 35
10.2 The algorithm for searching 36
10.2.1 Understanding the algorithm for searching 36

11 Quantum Physics 39
11.1 Wave function 39
11.2 Energy / Hamiltonian 40
11.3 Schrödinger equation 41
11.3.1 Time-dependent Schrödinger equation 41
11.3.2 Time-independent Schrödinger equation 41
11.4 Infinite square well 42

12 From Quantum Physics to a Quantum Computer 43

13 Ion-based quantum computers 46
13.1 Electron in an atom 46
13.2 Setup for the ion traps 47
13.2.1 Cooling 47
13.2.2 Unitaries 48
13.2.3 Measurements 50

14 Universal set of gates 51
14.1 Pauli gates as universal qubit gates? 51
14.2 Rotation gates 52
14.3 Clifford gates 53
14.4 Gottesmann-Knill theorem 53

15 Repetition code 54

Welcome

These are the lecture notes for the “Introduction to Quantum Computing” lecture held by Dominique Unruh at RWTH Aachen in the summer term 2024. The lecture notes are updated throughout the semester and should be viewed as an addition to the handwritten notes and the lecture recordings.

If you spot an error, please send Jannik Hellenkamp an e-mail. You can contact Jannik by sending an e-mail to firstname.lastname@rwth-aachen.de (please replace first and lastname with Jannik’s full name). If you have a question of understanding, please ask it in the Moodle forum.

Changelog

Version 0.1.8 (27.07.2024)

- added chapters 4,5,6,7,8 and 14.
- minor error corrections

Version 0.1.7 (09.07.2024)

- finished chapter 12
- added chapter 13

Version 0.1.6 (03.07.2024)

- finished chapter 11
- started chapter 12

Version 0.1.5 (27.06.2024)

- finished chapter 10
- started chapter 11

Version 0.1.4 (18.06.2024)

- finished chapter 9
- stated chapter 10

Version 0.1.3 (11.06.2024)

- added/extended 9.4 + 9.5 (Post processing and Beginning of DFT circuit)
- updated 9.3
- added chapter 3 (Quantum systems)
- error correction in chapter 9

Version 0.1.2 (31.05.2024)

- minor changes to chapter 2
- added chapter 9

Version 0.1.1 (16.05.2024)

- Started the lecture notes.

1 Introduction

1.1 Double slit experiment

This section will be updated later on, since there is quite a lot of graphical stuff.

1.2 What is a quantum computer?

To start into the topic of quantum computing and to understand the differences from classical computers, we first need to look at some of the basics of such classical computers.

In a classical computer the information is stored in *bits* which can either be in the state 0 *or* the state 1. These bits can be manipulated through different classical operations and we can look at these bits and read them, without interfering with the system or changing any states.

In a quantum computer the information is stored in a *qubit* which can be in a superposition *between* the state 0 and 1. Just as with classical computers, we can construct variables from these qubits to store bigger numbers. For example a 64-*qu*bit integer would be described by 64 qubits which are in a superposition between 0 and $2^{64} - 1$. This can be imagined best as a variable where the universe has not yet decided on its value and therefore the variable has all possible values at the same time.

We can now use this superposition and manipulate it with different quantum operations. Contrary to a classical computer, in a quantum computer these operations are "applied" at all possible input values at the same time and the result is a superposition of all possible results of the operation. We call this effect *quantum parallelism*.

Example: Quantum parallelism

Let's say you have a quantum variable x in a superposition of numbers between 0 and $2^{64} - 1$ (all possible 64-bit values) and some function $f(x)$. You program a quantum computer to compute $f(x)$.
The quantum computer would compute $f(x)$ for $x = 0, x = 1, x = 2, ...$ at the the same time and the result of this computation is a superposition of all possible values $f(x)$.

Reading this, one might be tempted to utilize quantum parallelism to run any algorithm on a quantum computer in order to optimize runtime. Unfortunately there is a big catch with quantum computers: If we try to look at the state of a qubit (also called *measuring*), the universe decides randomly on an outcome and therefore when measuring we only get the result of one computation and all the rest of the information is lost.

Example (continued): Quantum parallelism

After your quantum computer has calculated a superposition of all possible values $f(x)$, you want to get some information on the output and therefore you do a measurement on the resulting quantum state.
You will receive one random $f(x)$ and all the other possible solutions are lost.

Due to this restriction, naively running established algorithms on a quantum computer will not work. Fortunately there are some clever tricks to create some "interference" between different computations before measuring. This will give us useful information in some cases.

2 Probabilistic systems

To describe a quantum computer mathematically, we can do math similar to the known topic of probabilistic systems. We therefore first look into describing a probabilistic system.

2.1 Deterministic possibilities

At first we need to define all the different possible outcomes of our system. For example, for a coin flip this could be *heads* or *tails* and for a dice this could be the labels of the different sides. We call these possibilities *deterministic possibilities*. Note that we will only be using a *finite* number of possibilities.

Example: Random 2-bit number

Imagine you have a random number generator, which outputs 2-bit numbers. The deterministic possibilities of this generator are 00, 01, 10 and 11.

2.2 Probability distribution

Next, we need to assign each possibility a probability. We write this as $\Pr[x] = p$ where $p \in [0, 1]$ is the probability of the deterministic possibility x.

Example: Coin flip

For a coin flip the probability of heads would be $\Pr[\text{heads}] = \frac{1}{2}$ and the probability for tails would be $\Pr[\text{tails}] = \frac{1}{2}$.

If we combine all probabilities for all the possible outcomes and write them as a vector, we get a *probability distribution*.

Definition 2.1 (Probability distribution)**.** A vector $d \in \mathbb{R}^n$ is a valid probability distribution iff $\sum d_i = 1$ and $\forall i\ d_i \geq 0$.

This vector has n entries, where each entry corresponds to a deterministic possibility X and the probability of X is $\Pr[X] = d_i$. The sum over all probabilities has to be 1 and each entry needs to be nonnegative in order to be a valid probability.

Example (continued): Coin flip

For a coin flip the probability distribution would be $d_{\text{coin}} \in \mathbb{R}^2$ with $d = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$

Example (continued): Random 2-bit number

Recall your random 2-bit number generator from above. Imagine your generator outputs each deterministic possibility with equal probability, except for the possibility 00, which is never generated. The corresponding probability distribution would be

$$d_{\text{2-bit}} = \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix}$$

2.3 Probabilistic processes

With a probability distribution, we can only describe the probabilities of possibilities without any knowledge of a prior state. We therefore add another element to our toolbox of probabilistic systems called a *probabilistic process*.

A probabilistic process is a collection of n probability distributions, where for each deterministic possibility i there is a probability distribution a_i. This means that if the system is in deterministic possibility i before the process is applied, the system will afterwards be distributed according to a_i. We can write this as a matrix, where each column is a probability distribution a_i.

Definition 2.2 (Probabilistic process)**.** A matrix $A \in \mathbb{R}^{n \times n}$ is a valid probabilistic process iff for every column a of A, a is a valid probability distribution.

From Definition 2.1 we know that a valid probability distribution a has the properties $\sum a_i = 1$ and $\forall i\ a_i \geq 0$, therefore a matrix A is a probabilistic process iff $A \in \mathbb{R}^{n \times n}$ with $\sum a_i = 1$ and $\forall i\ a_i \geq 0$. Such a matrix is also called a *stochastic matrix*.

Example (continued): Random 2-bit number

Imagine a second device, which receives a 2-bit number as an input and flips both bits at the same time with a probability of $\frac{1}{3}$. The probability distributions for each of the deterministic possibility would then be

$$a_{00} = \begin{pmatrix} \frac{2}{3} \\ 0 \\ 0 \\ \frac{1}{3} \end{pmatrix}, a_{01} = \begin{pmatrix} 0 \\ \frac{2}{3} \\ \frac{1}{3} \\ 0 \end{pmatrix}, a_{10} = \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix}, a_{11} = \begin{pmatrix} \frac{1}{3} \\ 0 \\ 0 \\ \frac{2}{3} \end{pmatrix}$$

From this we can construct the process as a matrix from these processes as follows:

$$A_{\text{flip}} = \begin{pmatrix} a_{00} & a_{01} & a_{10} & a_{11} \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & 0 & 0 & \frac{1}{3} \\ 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{3} & 0 & 0 & \frac{2}{3} \end{pmatrix}$$

Applying a probabilistic process

Having defined probability distributions and probabilistic processes, we can now combine these two elements and apply a probabilistic process on a probability distribution.

Definition 2.3 (Applying a probabilistic process)**.** Given an initial probability distribution $x \in \mathbb{R}^n$ and a probabilistic process $A \in \mathbb{R}^{n \times n}$, the result $y \in \mathbb{R}^n$ of applying the process A is defined as

$$y = Ax$$

Example (continued): Random 2-bit number

Recall the 2-bit number generator and the bit flip from above. Imagine you would first draw a random 2-bit number from the generator and then run the bit flip device. We already know that the probability distribution of the generator is $d_{\text{2-bit}}$. Using A_{flip} we can calculate the final probability distribution:

$$A_{\text{flip}} \cdot d_{\text{2-bit}} = \begin{pmatrix} \frac{2}{3} & 0 & 0 & \frac{1}{3} \\ 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{3} & 0 & 0 & \frac{2}{3} \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} = \begin{pmatrix} \frac{1}{9} \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{2}{9} \end{pmatrix}$$

3 Quantum systems

With the basics for a probabilistic system defined, we now look into describing a quantum computer mathematically. In the following table you can see the analogy from the quantum world to the probabilistic world.

Probabilistic world	Quantum world
Probability distributions	Quantum states
Probabilities	Amplitudes
Deterministic possibilities	Classical possibilities
Stochastic matrix as process	Unitary matrix as process

3.1 Quantum states

One of the most important element of the quantum world is a quantum state. A quantum state describes the state of a quantum system as a vector. Each entry of the vector represents a *classical* possibility (similar to the deterministic possibilities in a probability distribution). The entries of a quantum state are called *amplitude*. In contrast to a probabilistic system, these entries can be negative and are also complex numbers.

These amplitudes tell us the probability of the quantum state being in the corresponding classical possibility. To calculate the probabilities from the amplitude, we can take the square of the absolute value of the amplitude.

This means that for the classical possibility x and a quantum state ψ the probability for x is $\Pr[x] = |\psi|^2$. To have valid probabilities, the sum of all probabilities need to sum up to 1. From this we get the formal definition of a quantum state:

Definition 3.1 (Quantum State)**.** A quantum state is a vector $\psi \in \mathbb{C}^n$ with $\sqrt{\sum_{i=1}^{n} |\psi_i|^2} = 1$.

Example: Some Quantum states

The following vectors are valid quantum states with the classical possibilities 0 and 1:

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

Note that the symbol $|\rangle$ is not yet introduced, so just understand it as some label at this point. The probabilities for each state can be calculated as follows:

$$\begin{array}{lll} |0\rangle: & \Pr[0] = |1|^2 = 1 & \Pr[1] = |0|^2 = 0 \\ |1\rangle: & \Pr[0] = |0|^2 = 0 & \Pr[1] = |1|^2 = 1 \\ |+\rangle: & \Pr[0] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} & \Pr[1] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} \\ |-\rangle: & \Pr[0] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} & \Pr[1] = |\frac{-1}{\sqrt{2}}|^2 = \frac{1}{2} \end{array}$$

We can see here that two different quantum states ($|+\rangle$ and $|-\rangle$) can have the same probabilities for all classical possibilities.

3.2 Unitary transformation

We now have defined quantum states and need a way to describe some processes, which we want to apply on the quantum states. In the probabilistic world, we have stochastic matrices for this, but unfortunately we can not use these matrices on quantum states, since the output of applying these on a quantum state is not guaranteed to be a quantum state again. We therefore look for a different property of a matrix for which the outcome of applying that matrix is guaranteed to be a quantum state. We get this property with *unitary* matrices.

Definition 3.2 (Unitary transformation)**.** Given a quantum state $\psi \in \mathbb{C}^n$ and a unitary matrix $U \in \mathbb{C}^{n \times n}$, the state after the transformation is a quantum state $U\psi$.

Lemma 3.1 (Unitary matrix)**.** *A matrix $U \in \mathbb{C}^{n \times n}$ is called* unitary *iff $U^\dagger U = I$ where I is the identity matrix and $U^\dagger$ is the complex conjugate transpose of U.*

A unitary matrix is by this lemma invertible, therefore we can undo all unitary transformations by applying $U^\dagger$.

Example: Some Unitary transformations

The following matrices are examples for unitary transformations:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

These matrices are called Pauli-matrices, we will get to know them later on.
As an example for applying a unitary on a quantum state, we apply the Pauli X matrix on the quantum state $|0\rangle$:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

4 Observing probabilistic and measuring quantum systems

So far we only talked about the description of a probabilistic and a quantum system. We now look into observing/measuring those systems.

4.1 Observing a probabilistic system

Observing a probabilistic system is the process of learning the outcome from a probability distribution. If our probability distribution for example represents a coin flip, observing this distribution is equivalent to actually flipping the coin. In the probabilistic case, an observation is just about updating our knowledge or beliefs. This will be different in the quantum case.

Definition 4.1 (Observing a probabilistic system)**.** Given a probability distribution $d \in \mathbb{R}^n$, we will get the outcome i with a probability d_i. The new distribution is then

$$e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow 1 \text{ at the } i\text{-th position}$$

The intuition for the new distribution is that we now after observing that i is the deterministic possibility for sure.

When observing a probabilistic system, the observation is just a passive process with no impact on the system. This means that there is no difference to the end result, whether we observe during the process or not. We take a look at an example to further understand this.

Example: Random 1-bit number

We use a random 1-bit number example similar to the random 2-bit example from Chapter 2. We have a distribution $d_{\text{1-bit}} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$ which represents the probability distribution of generating a 1-bit number with equal probability. We also have a process $A_{\text{flip}} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix}$ which flips the bit with a probability of $\frac{1}{3}$.
We look at two different cases: For the first case, we observe only the final distribution and for the second case we observe after the generation of the 1-bit number and we also observe the final distribution.

4.1.0.0.1 * Observing the final distribution

From Section 2.3 we know that the final distribution d is

$$d = A_{\text{flip}} \cdot d_{\text{1-bit}} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

We observe this distribution and will get outcome 0 and the new distribution $d = e_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ with a probability of $\Pr[0] = d_0 = \frac{1}{2}$ and the outcome 1 and the new distribution $d = e_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ with a probability of $\Pr[1] = d_1 = \frac{1}{2}$.

4.1.0.0.2 * Observing after generation and the final distribution

We now observe the system after the generation of the 1-bit number and also observe the final distribution. After the generation, we will get outcome 0 and the new distribution $d = e_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ with a probability of $\Pr[0] = d_0 = \frac{1}{2}$ and the outcome 1 and the new distribution $d = e_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ with a probability of $\Pr[1] = d_1 = \frac{1}{2}$.

We now apply in each case the matrix A_{flip}. This will give us the outcome $A_{\text{flip}} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}$ for the case of the outcome 0 and the outcome $A_{\text{flip}} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \end{pmatrix}$ for the case of the outcome 1. If we observe the distribution $\begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}$, we will get the outcome 0 and the new distribution $d = e_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ with a probability of $\Pr[0] = \frac{2}{3}$ and the outcome 1 and the new distribution $d = e_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ with a probability of $\Pr[1] = \frac{1}{3}$. If we observe the

distribution $\begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \end{pmatrix}$, we will get the outcome 0 and the new distribution $d = e_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ with a probability of $\Pr[0] = \frac{1}{3}$ and the outcome 1 and the new distribution $d = e_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ with a probability of $\Pr[1] = \frac{2}{3}$.
Combining these probabilities, we get the total probability $\Pr[0] = \frac{1}{2}\frac{2}{3} + \frac{1}{2}\frac{1}{3} = \frac{1}{2}$ for the outcome 0 and the probability $\Pr[1] = \frac{1}{2}\frac{1}{3} + \frac{1}{2}\frac{2}{3} = \frac{1}{2}$ for the outcome 1. This is the same as observing the final distribution.

4.2 Measuring a quantum system

Unlike in the probabilistic system, the "observation" of a quantum system is called *measuring*. The definition is similar to the observation of a probabilistic system, except that we need to take the absolute square of the amplitude to get the probability and that the state after measuring is called *post-measurement-state* (p.m.s.).

Definition 4.2 (Measuring a quantum system)**.** Given a quantum State $\psi \in \mathbb{C}^n$, we will get the outcome i with a probability $|\psi_i|^2$. The post-measurement-state (p.m.s.) is then

$$e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow 1 \text{ at the } i\text{-th position}$$

This is called a complete measurement in the computational basis.

With this similarity to the probabilistic observation in the definition, one might assume that measuring a quantum state has also no impact on the system. **This is not the case, measuring a quantum state changes the system!** We can see this effect with an example:

Example: Measuring a quantum system

Let $\psi = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ be a quantum state and $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ be a unitary transformation.
We look at two different cases: First we apply H immediately and then measure the system. As a second case, we do a measurement before the application of the H unitary and then a measurement after applying it.

4.2.0.0.1 * Measure the final state
We first calculate the state after applying H:

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

Measuring this state will get the outcome 0 with probability $\Pr[0] = |\psi_0|^2 = 1$ and have the post-measurement-state $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

4.2.0.0.2 * Measure the initial and the final state
Measuring ψ with no further unitary matrices applied can have the outcome 0 or 1. We will look at the final measurement for each case:
The first measurement will have outcome 0 with probability $\Pr[0] = |\psi_0|^2 = \frac{1}{2}$ and the post-measurement-state will be $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$. H applied to this post-measurement-state will be $H \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$. When measuring this state, we will get the outcome 0 with probability $\Pr[0] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$ and outcome 1 with with probability $\Pr[1] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$.
The outcome 1 will appear at the initial state with probability $\Pr[1] = |\psi_1|^2 = \frac{1}{2}$ and the post-measurement-state will be $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. H applied to this post-measurement-state will be $H \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$. When measuring this state, we will get the outcome 0 with probability $\Pr[0] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$ and outcome 1 with with probability $\Pr[1] = |-\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$.
So independent of the outcome of the first measurement, at the second measurement the outcome 0 and 1 have a probability of $\frac{1}{2}$. This shows that when measuring before applying H, we will receive different probabilities for the second measurement, then when measuring only at the end. This proves that measurements can change the system.

4.3 Elitzur–Vaidman bomb tester

This section will be updated later on.

5 Partial observing and measuring systems

In the previous chapter, we looked into observing a probabilistic and measuring a quantum system. In this approach, we alway looked at the full system. This means that we either have no measurement at all or we know the exact possibility, in which our system is.

For larger systems, this can become quite complicated, as we might not need the full measurement, but only some partial information. For example if we consider a dice throw, we might not need the final number of the dice, but we are only interested if it is an even or an odd number. To archive this, we can do a partial observation on a probabilistic system.

5.1 Partially observing a probabilistic system

To perform a partial observation on a probabilistic system, we first decide on which *alternatives* we want to distinguish. Each alternative is described by a set A of deterministic possibilities. By performing the partial observation, we will get for each alternative A the probability that the system is in a state in A.

Definition 5.1 (Partially observing a probabilistic system)**.** Given a distribution $M \in \mathbb{R}^N$ and a family of alternatives $A_1, \dots, A_m \subseteq \{1, \dots, N\}$ with $A_i \cap A_j = \emptyset$ and $\bigcup_i A_i = \{1, \dots, N\}$, the probability of observing the alternative k is given by

$$\Pr[\text{outcome} = k] = \sum_{i \in A_k} M_i = \|v^{(k)}\|_1$$

$\|\|_1$ denotes the 1-norm here. The distribution after the observation of the outcome k is given by the normalized conditional distribution. This is computed by

1. Computing the non-normalized conditional distribution $v^{(k)}$ denoted by $v^{(k)} := (v_1, \dots, v_N)$ with
$$v_i := \begin{cases} M_i & \text{if } i \in A_k \\ 0 & \text{else} \end{cases}$$
2. Computing the normalized conditional distribution by calculating
$$\frac{v^{(k)}}{\Pr[\text{outcome} = k]}$$

Note that similar to the full observation of a probabilistic system a partial observation has no impact on the system. We only get some new knowledge but do not influence the actual system by our observation.

5.2 Partially measuring a quantum system

Similar to the partial observation of a probabilistic system, we can perform a partial measurement on a quantum system.

Definition 5.2 (Partially measuring a quantum system)**.** Given a quantum state $\psi \in \mathbb{C}^N$ and a family of alternatives $A_1, \dots, A_m \subseteq \{1, \dots, N\}$ with $A_i \cap A_j = \emptyset$ and $\bigcup_i A_i = \{1, \dots, N\}$, the probability of the alternative k is given by

$$\Pr[\text{outcome} = k] = \sum_{i \in A_k} |\psi_i|^2 = \|\phi^{(k)}\|^2$$

$\|\|$ denotes the Euclidean norm here. The post-measurement-state (p.m.s.) of the outcome k is computed as follows:

1. Computing the non-normalized post-measurement-state $\phi^{(k)}$ denoted by $\phi^{(k)} := (\phi_1, \dots, \phi_N)$ with
$$\phi_i := \begin{cases} \psi_i & \text{if } i \in A_k \\ 0 & \text{else} \end{cases}$$
2. Computing the normalized post-measurement-state by calculating
$$\frac{\phi^{(k)}}{\sqrt{\Pr[\text{outcome} = k]}} = \frac{\phi^{(k)}}{\|\phi^{(k)}\|}$$

As with the complete measurement for quantum systems, the measurement can change the system. Note that there exists other types of definitions for a measurement e.g. projective measurement, generalized measurements, POVMs, … The variant above can best be described as a “projective measurement in the computational basis”.

6 Composite Systems

So far our probabilistic and quantum systems consist of only one single distribution/state. In the real world, quantum computers often have several different registers (variables).

In theory, we could use a single very big distribution/state to model multiple qubits. For example a 10 qubit system could be modeled with the classical possibilities $0000000000, 0000000001, \dots, 1111111110, 1111111111$.

Unfortunately the vector for these states gets really big, for 10 qubits, the vector would have the dimension of 1024. Since this is very inconvenient to write down, we need to look at a different solution. For this, we compose different probabilistic or quantum systems with each other.

6.1 Constructing composite systems

Definition 6.1 (Composite systems / Tensor product)**.** Given two probabilistic or quantum systems A and B with the possibilities of A given by $1, \dots, N$[1] and a distribution/state a and with the possibilities of B given by $1, \dots, M$ and a distribution/state b, the *composite* system called AB has the possibilities

$$11, 12, \dots, 1M, 21, 22, \dots, 2M, \dots, N1, N2, \dots, NM$$

and the distribution/state ab of AB is given by the *tensor product*

$$ab = a \otimes b = \begin{pmatrix} a_1 \cdot b \\ \vdots \\ a_N \cdot b \end{pmatrix}$$

This vector has the size NM.

Note that the definition of combining a probabilistic and a quantum system are the same.

Example: Tensor product

Given two vectors a and b, with

$$a = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \quad b = \begin{pmatrix} 10 \\ 100 \end{pmatrix}$$

the tensor product of those vectors is given by

$$a \otimes b = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \otimes \begin{pmatrix} 10 \\ 100 \end{pmatrix} = \begin{pmatrix} 1 \cdot 10 \\ 1 \cdot 100 \\ 2 \cdot 10 \\ 2 \cdot 100 \\ 4 \cdot 10 \\ 4 \cdot 100 \end{pmatrix} = \begin{pmatrix} 10 \\ 100 \\ 20 \\ 200 \\ 40 \\ 400 \end{pmatrix}$$

[1] Note that the possibilities are labels $1, \dots, N$ and do not have be the numbers $1, \dots, N$. We could also label these e.g. *red*, *green* and *blue* (or any other label).

We now need a way to apply operators on these combined systems. For this we can also construct the tensor product of either two probabilistic processes or two unitary transformations by using the tensor product of two matrices.

Definition 6.2 (Composite matrices / Tensor product)**.** Given two matrices S and T with S of the size $N \times N$ and T of the size $M \times M$. The tensor product $S \otimes T$ of is given by

$$S \otimes T = \begin{pmatrix} S_{11}T & \dots & S_{1N}T \\ \vdots & \ddots & \vdots \\ S_{N1}T & \dots & S_{NN}T \end{pmatrix}$$

Overall we can say: If we apply S to the system A and T to the system B, we apply $S \otimes T$ to the composite system AB.

If the distribution d_{AB} of a given probabilistic system AB can be written as a composite of two distributions d_A and d_B, we know that A and B are independent of each other. If we can not write d_{AB} as two septate systems, the probabilities are depended on each other.

If the quantum state ψ_{AB} of a given quantum system AB can be written as a composite of two different quantum states ψ_A and ψ_B, the quantum states of A and B are independent of each other. If we can not write ψ_{AB} as a tensor product of two quantum systems, the quantum states depend on each other. We call this *entangled*.

6.2 Measuring composite systems

To perform a (partial) observation or (partial) measurement on a composite system AB, we can compose two separate measurements on the systems A and B similar as we constructed the tensor product.

Definition 6.3 (Composite measurements)**.** Given two systems A and B with possibilities $1, \dots, N$ and $1, \dots, M$ and two partial measurements M_A and M_B on systems A and B with alternatives $A_1, \dots, A_N \subseteq \{1, \dots, N\}$ and $B_1, \dots, B_M \subseteq \{1, \dots, M\}$. The measurement $M_A \otimes M_B$ on AB is a measurement with the alternatives $C_{11}, C_{12}, \dots, C_{NM}$ where $C_{ij} = A_i \times B_j$.
If we only have a set of alternatives for system A, we can do a measurement $M_A \otimes I$ with alternatives $C_1, \dots, C_N := A \otimes \{1, \dots, M\}$.

7 Quantum Circuits

In the previous chapters, we learned the basics on how to construct a quantum computer. We will now start constructing quantum circuits from these. Note that we will no longer look into probabilistic system.

The quantum systems which we consider in the following sections consist of qubits, unless specified otherwise. A qubit is a quantum state ψ with $\psi \in \mathbb{C}^2$.

7.1 Visual language

So far we have only seen the elements of quantum computers in a mathematical form. When constructing quantum circuits, this can get very unreadable very fast. Therefore we can draw quantum circuits as a picture, which also helps us to get a better intuition for these circuits. You can see a very simple example here:

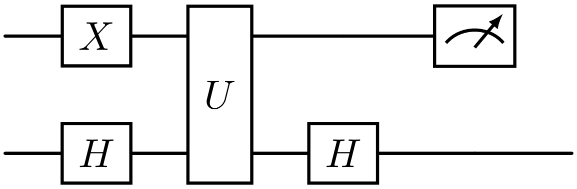

Figure 7.1: A basic quantum circuit

In this circuit we have two qubits, which are drawn as separate wires. We first apply the unitary X on the top wire and at the same time we apply the unitary H at the bottom wire. Mathematically this can be written as $X \otimes H$. Next we apply the unitary U, which operates on both qubits. After this, we apply a unitary H on the bottom wire. Since we do not apply anything on the top wire, we can write this mathematically as $I \otimes H$. Finally we measure the top qubit. This means a complete measurement in the computational basis of the qubit as described in Section 4.2.

7.2 Important gates

When working with quantum computers, we encounter some of the same unitaries very often. We distinguish between single qubit gates (unitary transformations $\in \mathbb{C}^{2\times 2}$) and gates on multiple qubits.

7.2.1 Single qubit gates

The following gates are relevant single qubit gates:

Definition 7.1 (Pauli matrices)**.** The Pauli matrices X, Y and Z are defined as

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Note that X is also called bit-flip.

Definition 7.2 (Hadamard gate)**.** The Hadamard gate H is defined as

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

The Hadamard gate is useful for introducing superpositions as it takes a classical bit $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and transforms it into a superposition $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$.

7.2.2 Controlled-NOT gate

The gates introduced above only operate on a single qubit. To connect two different qubits, we need gates which operate on multiple qubits. For this we introduce the controlled-not:

Definition 7.3 (Controlled-NOT gate)**.** The controlled-NOT gate CNOT $\in \mathbb{C}^{4\times 4}$ is defined as

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

The CNOT gate flips the qubit of the second qubit if the first qubit is 1. We call the first wire the controlling wire and the second wire the target wire. It can be drawn in a quantum circuit as follows:

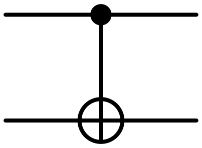

Figure 7.2: Controlled-NOT in a quantum circuit

If the second qubit should be the controlling wire and the first qubit the target wire, we can use CNOT$'$ denoted as

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

7.3 Ket Notation

So far we have only seen vectors as a way to mathematically describe a quantum state. This can get quite inconvenient if the vector get bigger and also often contains not that much useful information (e.g. a lot of 0 entries). We therefore introduce a new form of writing quantum states called the *ket* notation.

The idea works as follows: We can rewrite a quantum state ψ in the following way

$$\psi = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_N \end{pmatrix} = \psi_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \psi_2 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \psi_N \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = \sum_{i=1}^{N} \psi_i \cdot e_i$$

The vector e_i denotes the vector with 0 entries at every position except the i-th position, where the entry is 1.

From this notation we already get an advantage, since we can drop out all 0 entries. But we still have no intuitive mapping from the vector e_i to the classical possibility represented by e_i. For this we use a $|\rangle$ symbol. More precise this means for a classical possibility x, which is the

i-th possibility and is represented by e_i, we write

$$|x\rangle := e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow 1 \text{ at the } i\text{-th position}$$

Example: Ket notation

Given a quantum system with the classical possibilities 00, 01, 10 and 11, the quantum state $\psi = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}^T$ can be written as

$$\psi = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$$

Written like this, we can see at first glance that this is a superposition of the classical possibilities 00 and 11.

Note that the ket notation can also be used in a few other ways. We can use it as described above to describe the classical possibility x as $|x\rangle$, but we also use it to emphasize that ψ is a quantum state by writing $|\psi\rangle$ (here ψ is not a classical possibility). We also have two special cases $|+\rangle$ and $|-\rangle$ which are defined as follows:

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

Which of the meanings of the symbol $|\rangle$ is meant has to be deduced from the context.

8 Bernstein-Vazirani Algorithm

With all the quantum basics from the previous chapters, we now can start with the first quantum algorithm. This algorithm is called the Bernstein-Vazirani algorithm.

This algorithm tackles the following problem: Given a secret $s \in \{0,1\}^n$ and the function $f : \{0,1\}^n \to \{0,1\}$, defined as $f(x) := x \cdot s$. $\cdot$ denotes the inner product of two bitstrings here. This means that for bitstrings x and y of length n, the inner product is $x \cdot y = x_1 y_1 + \cdots + x_n y_n \bmod 2$.

The goal is to find the secret s using as little queries of f as possible. By "query" we mean an evaluation of f. The word query stems from the fact that we often think of the algorithm having access to a so-called "oracle" which we can "query" to get $f(x)$.

Classically we will need at least n queries to f to get s definitely. A classical algorithm with only $m \le n$ queries will get s with a probability of 2^{m-n} if s is uniformly random.

We will now look at a quantum algorithm which will find s with only one evaluation of f. This algorithm is sketched in the following circuit:

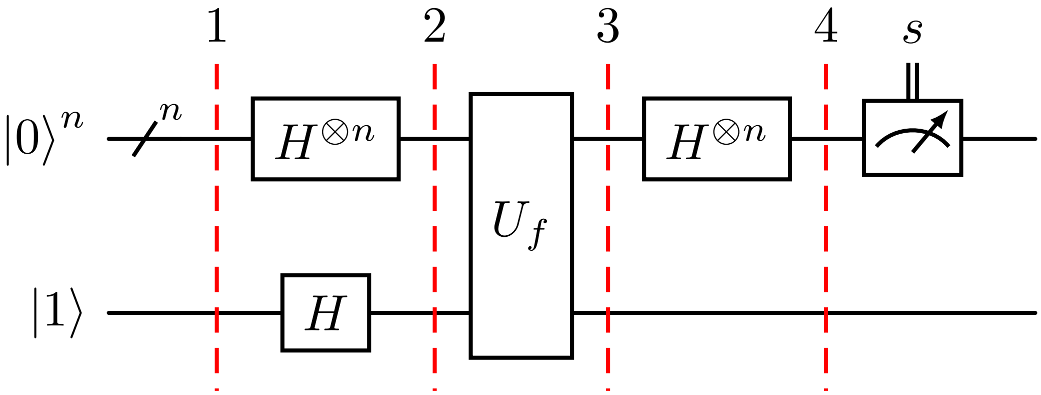

Figure 8.1: The quantum circuit for Bernstein-Vazirani

Note that U_f is defined with the explanation below.

We start with n qubits on the top wire. All of these qubits are in the state $|0\rangle$, which we write $|0\rangle^n = |0\rangle \otimes \cdots \otimes |0\rangle$. The bottom wire is in the state $|1\rangle$. Both wires composed together can be written as $\psi_1 = |0\rangle^n \otimes |1\rangle$, which is the overall starting state of our algorithm (the state at slice 1). We now perform the following steps

1. First we apply a Hadamard gate on all qubits. This is denoted for the first n qubits by the $H^{\otimes n}$ gate and for the last qubit by the H gate on the bottom wire. The resulting

quantum state is calculated as follows:

$$\begin{aligned}
\psi_2 &= (H^{\otimes n} \otimes H) \big(|0\rangle^n \otimes |1\rangle \big) \\
&= \big(H^{\otimes n} |0\rangle^n \big) \otimes H |1\rangle \\
&= (|+\rangle)^{\otimes n} \otimes |-\rangle \\
&= \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right)^{\otimes n} \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |-\rangle
\end{aligned}$$

Roughly speaking, we are now in the superposition over all classical possibilities on the top wire and in $|-\rangle$ on the bottom wire at slice 2.

2. Next, we apply the unitary U_f on both wires. This unitary is defined as

$$U_f \; |x, y\rangle = |x, y \oplus f(x)\rangle$$

This unitary represents the function f and combines the output of $f(x)$ with the bottom wire y. For our quantum states, this means that the state after U_f at slice 3 can be calculated as follows:

$$\begin{aligned}
\psi_3 &= U_f \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} U_f \Big(|x\rangle \otimes |-\rangle \Big) \\
&\overset{*}{=} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \otimes |-\rangle \\
&= \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \right) \otimes |-\rangle
\end{aligned}$$

Note that the $*$ holds since we can rewrite $U_f(|x\rangle \otimes |-\rangle)$ as

$$\begin{aligned}
&U_f\Big(|x\rangle \otimes |-\rangle\Big) \\
&= \frac{1}{\sqrt{2}} U_f |x, 0\rangle - \frac{1}{\sqrt{2}} |x, 1\rangle \\
&= \frac{1}{\sqrt{2}} |x, f(x)\rangle - \frac{1}{\sqrt{2}} |x, \overline{f(x)}\rangle \\
&= \begin{cases} \frac{1}{\sqrt{2}} |x, 0\rangle - \frac{1}{\sqrt{2}} |x, 1\rangle & f(x) = 0 \\ \frac{1}{\sqrt{2}} |x, 1\rangle - \frac{1}{\sqrt{2}} |x, 0\rangle & f(x) = 1 \end{cases} \\
&= \begin{cases} |x\rangle \otimes |-\rangle & f(x) = 0 \\ -|x\rangle \otimes |-\rangle & f(x) = 1 \end{cases} \\
&= (-1)^{f(x)} |x\rangle \otimes |-\rangle
\end{aligned}$$

The bottom wire has not changed and is still $|-\rangle$. But on the top wire, we now have $f(x)$ somehow encoded into our quantum state. The phenomenon that the output of f is encoded as a -1 in the input register is called phase kickback. Measuring this quantum state would not give us any advantage, since we would just get one random x. We therefore perform one final step before measuring.

3. As the final unitary, we perform another $H^{\otimes n}$ on the top wire. We hope that the result of this unitary transformation is the state $|s\rangle$. To check, whether our hopes become reality, we can calculate $(H^{\otimes n})^\dagger |s\rangle \otimes |-\rangle$ and check if it is equal to ψ_3. We do it in this direction,

since these calculations are a bit simpler:

$$\begin{aligned}
(H^{\otimes n})^{\dagger} |s\rangle \otimes |-\rangle &= H^{\otimes n} |s\rangle \otimes |-\rangle \\
&= H^{\otimes n}(|s_1\rangle \otimes \dots \otimes |s_n\rangle) \otimes |-\rangle \\
&= H |s_1\rangle \otimes \dots \otimes H |s_n\rangle \otimes |-\rangle \\
&= \bigotimes_{i=1}^{n} \left(\frac{1}{\sqrt{2}} |0\rangle + (-1)^{s_i} \frac{1}{\sqrt{2}} |1\rangle \right) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{i=1}^{n} \Big(|0\rangle + (-1)^{s_i} |1\rangle \Big) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \Big((-1)^{x_1 s_1} (-1)^{x_2 s_2} \dots (-1)^{x_n s_n} |x\rangle \Big) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \Big((-1)^{\sum_i x_i s_i \bmod 2} |x\rangle \Big) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x} |x\rangle \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \otimes |-\rangle \\
&= \psi_3
\end{aligned}$$

This calculation shows that we have the quantum state $|s\rangle \otimes |-\rangle$ at slice 4.

4. We now perform a measurement on the top wire and measure s as a result.

This concludes the Bernstein-Vazirani algorithm.

9 Shor's Algorithm

One of the best known quantum algorithm is Shor's algorithm for finding the prime factors of an integer. It was developed by Peter Shor in 1994.

9.1 Discrete Fourier Transformation

One of the tools required for Shor's algorithm is the Discrete Fourier Transformation (DFT). Generally, a Fourier transformation is a mathematical technique that decomposes a function into its constituent frequencies. We use the DFT to find the period of a vector.

The DFT is defined as follows:

Definition 9.1 (Discrete Fourier Transformation (DFT))**.** The discrete Fourier transform (DFT) is a linear transformation on $\mathbb{C}^M$ represented by the matrix

$$\mathrm{DFT}_M = \frac{1}{\sqrt{M}}(\omega^{kl})_{kl} \in \mathbb{C}^{M\times M}$$

with $\omega = e^{2i\pi/M}$, which is the M-th root of unity.

This transformation is best imagined as a process, which takes a periodic vector as an input and outputs the period of that vector. The DFT has some important properties, which help us later on.

Theorem 9.1 (Properties of the DFT)**.** *Here are some properties of the DFT which can be used without further proof.*

1. *The DFT is unitary.*
2. $\omega^t = \omega^{t \mod M}$ *for all* $t \in \mathbb{Z}$*.*
3. *Given a quantum state* $\psi \in \mathbb{C}^M$ *which is* r*-periodic and where* $r \mid M$, $\mathrm{DFT}_M\,\psi$ *will compute a quantum state* $\phi \in \mathbb{C}^M$*, which has non-zero values on the multiples of* $\frac{M}{r}$*. Note that* $\frac{M}{r}$ *intuitively represents the frequency of* ψ*. This means, that*

$$|\phi_i| = \begin{cases} \frac{1}{\sqrt{r}}, & \textit{if } \frac{M}{r} \mid i \\ 0, & \textit{otherwise} \end{cases}$$

9.2 Reducing factoring to period finding

With the DFT, we have seen, that we can use a unitary to find the period of a quantum state. We now look into using period finding to factor integers. We first look at the definition of the two problems:

Definition 9.2 (Factoring problem)**.** Given integer N with two prime factors p, q such that $pq = N$ and $p \neq q$, find p and q.

Note that this definition of the factoring problem is a simplified version of the factoring problem, where N has only 2 prime factors.

Definition 9.3 (Period finding problem)**.** Given $f : \mathbb{Z} \to X$ with $f(x) = f(y)$ iff $x \equiv y \bmod r$ for some fixed secret r. r is called the *period* of f. Find r.

To start the reduction, we need a special case of the period finding problem called order finding:

Definition 9.4 (Order finding problem)**.** For known a and N which are relatively prime, find the period r of $f(i) = a^i \bmod N$. We call r the order of a written $r = \operatorname{ord} a$. (This is similar to finding the smallest $i > 0$ with $f(i) = 1$).

Since the order finding problem is just the period finding problem for a specific $f(x)$, we know that if we can solve the period finding problem within reasonable runtime, we can also solve the order finding problem within reasonable runtime. We now reduce the factoring problem to the order finding problem:

We have a integer N as an input for the factoring problem.

1. Pick an $a \in \{1, \dots, N-1\}$ with a relatively prime to N.
2. Compute the order of a, so that $r = \operatorname{ord} a$ (using the solver for the order finding problem).
3. If the order r is odd, we abort.
4. Calculate $x := a^{\frac{r}{2}} + 1 \bmod N$ and $y := a^{\frac{r}{2}} - 1 \bmod N$.
5. If $\gcd(x, N) \in \{1, N\}$, we abort.
6. We compute $p = \gcd(x, N)$ and $q = \gcd(y, N)$.

The output of the reduction are p, q, such that $pq = N$. This holds, since

$$xy = (a^{\frac{r}{2}} + 1)(a^{\frac{r}{2}} - 1) = a^r - 1 \equiv 1 - 1 = 0 \pmod N$$

Theorem 9.2 (Probability of an abort)**.** *If N has at least two different prime factors and N is odd, then the probability to abort is $\leq \frac{1}{2}$.*

All in all this reduction shows, that if we have an oracle which can solve the period finding problem within reasonable runtime, we can also solve the factoring problem within reasonable runtime (since all other operations are classically fast to compute).

9.3 The quantum algorithm for period finding

We now look into an quantum algorithm that solves the period finding problem within reasonable runtime. The quantum circuit for Shor's algorithm requires a $f : \mathbb{Z} \to X$ which is r-periodic. X denotes an arbitrary set here. We choose a number m which needs to be big enough to encode the values of X and choose a number n under the condition of $n \geq 2\log_2(r)$ for the post processing to work. Note that when using this algorithm for factoring, we choose n to be $n := 2|N|$, since $r \leq N$. $|N|$ denotes the number of bits needed to encode N here.

The quantum algorithm for period finding is shown in this figure:

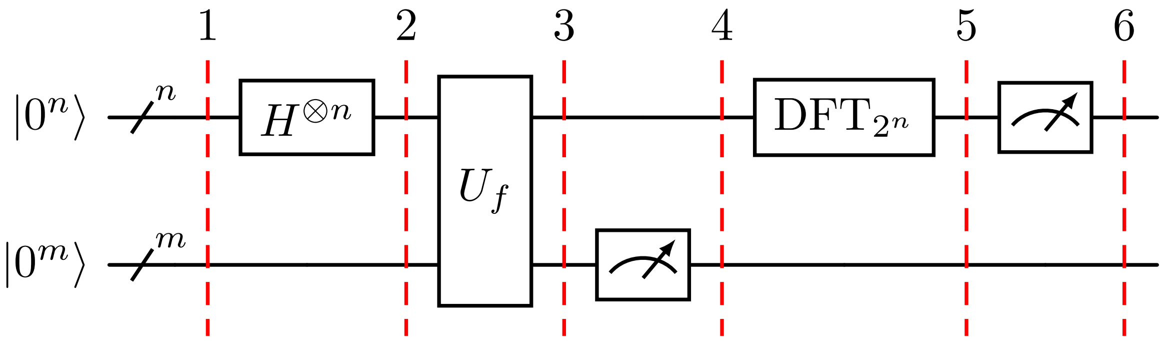

Figure 9.1: Shor's algorithm (quantum part)

The algorithm works as follows:

1. We start with a $|0\rangle$ entry on every wire.
2. We bring the top wire into the superposition over all entries. The quantum state is then $2^{\frac{-n}{2}} \sum_x |x\rangle \otimes |0^m\rangle$.
3. We apply U_f, which is the unitary of $f : \{0,1\}^n \to \{0,1\}^m$. This calculates the superposition over all possible values $f(x)$ on the bottom wire. The resulting quantum state is $2^{\frac{-n}{2}} \sum_x |x, f(x)\rangle$.
4. To understand the algorithm better, we measure the bottom wire at this point. This will give us one random value $f(x_0)$ for some x_0. The top wire will then contain a superposition over all values x where $f(x) = f(x_0)$. Since f is known to be r-periodic, we know, that $f(x) = f(x_0)$ iff $x \equiv x_0 \bmod r$. This means, that the resulting quantum state on the top wire is periodic and can be written as $\frac{\sqrt{r}}{\sqrt{2^n}} \sum_{x \equiv x_0 \bmod r} |x\rangle \otimes |f(x_0)\rangle$. For simplicity we assume, that $r \mid 2^n$ holds.
5. We apply the Discrete Fourier Transform on the top wire. This will "analyze" the top wire for the period and output a vector with entries at multiples of $\frac{2^n}{r}$ as seen in Theorem 9.1. For simplicity we still assume, that $r \mid 2^n$ holds.
6. We measure the top wire and get one random multiple of $\frac{2^n}{r}$, which we can denote as $a \cdot \frac{2^n}{r}$

Since we get a multiple of $\frac{2^n}{r}$ on each run, we can simply run the algorithm multiple times to get different multiples and then compute $\frac{2^n}{r}$ by taking the gcd of those multiples. From that we compute r. Unfortunately this only works because we assumed $r \mid 2^n$. Since this does usually not hold, we only get approximate multiples of $\frac{2^n}{r}$ (which is not even an integer) and thus post processing is a bit more complex.

9.4 Post processing

So far we have seen the DFT to analyze the period of a quantum state, we have seen a way to reduce the factoring problem to the period finding and we have seen a quantum algorithm for finding an approximate multiple of such a period of a function. We just need one final step to find r. For this we start with a theorem:

Theorem 9.3. *Iff $f : \mathbb{Z} \to X$ is r-periodic, the following holds with probability $\Omega(1/\log\log r)$:*

$$\frac{-r}{2} \leq rc \bmod 2^n \leq \frac{r}{2}$$

where c is the output of the second measurement of the quantum circuit described in Section 9.3 and n is the number of qubits on the upper wire of the quantum circuit.

We assume that the theorem holds for our outcome c of the second measurement (if that is not the case, our result will be wrong and we can just run the quantum algorithm again to get a different outcome):

Then exists an integer d such that:

$$\begin{aligned} & |rc - d2^n| \leq \frac{r}{2} \\ \iff & |\frac{c}{2^n} - \frac{d}{r}| \leq \frac{1}{2^{n+1}} \quad \mid \text{division by } r \cdot 2^n \end{aligned}$$

The fraction $\frac{c}{2^n}$ is known, so the goal is to find a fraction $\frac{d}{r}$ that is $\frac{1}{2^{n+1}}$-close to $\frac{c}{2^n}$.

Since n is the number of qubits used in the quantum circuit and was chosen, such that $n \geq 2\log_2(r)$ and thus $2^n \geq r^2$ holds and from this we know that $\frac{1}{2^{n+1}} \leq \frac{1}{2r^2}$ holds as well.

So if Theorem 9.3 holds, we now $|\frac{c}{2^n} - \frac{d}{r}| \leq \frac{1}{2r^2}$ also holds. Our task is now rewritten to find $\frac{d}{r}$ under this condition. For this we use another theorem:

Theorem 9.4. *For a given real number $\varphi \geq 0$ and integer $q > 0$ there is at most one fraction $\frac{d}{r}$ with $r \leq q$ and $|\varphi - \frac{d}{r}| \leq \frac{1}{2q}$. In this case, this $\frac{d}{r}$ is a convergent of the continued fraction expansion of φ.*

This theorem uses the convergent of a continued fraction expansion. A continued fraction expansion of a number t is the number rewritten as a fraction in the form

$$t = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots}}}$$

where a_i always has to be the biggest possible integer. We call $[a_0, a_1, a_2, a_3, ...]$ the continued expansion of t. The expansion is finite iff t is rational. For a given continued expansion, a prefix $[a_0, \ldots, a_i]$ is called a convergent. Writing this convergent as a normal fraction will give us an approximation of the number t.

Example: continued expansion of a fraction

The number 2.3 can be written as

$$2.3 = 2 + \frac{1}{3 + \frac{1}{3+0}}$$

and the continued fraction expansion of 2.3 is $[2, 3, 3]$. The expansions $[2]$ and $[2, 3]$ are convergents of the expansion of 2.3 and written as a fraction will give us the approximations 2 and $2 + \frac{1}{3} = 2.\bar{3}$.

The number 0.99 can be written as

$$0.99 = 0 + \frac{1}{1 + \frac{1}{99+0}}$$

and the continued fraction expansion of 0.99 is $[0, 1, 99]$. The expansions $[0]$ and $[0, 1]$ are convergents of the expansion of 0.99 and written as a fraction will give us the approximations 0 and $0 + \frac{1}{1} = 1$.

Using Theorem 9.4 (with $\varphi := \frac{c}{2^n}$ and $q := 2^n$) we can find $\frac{d}{r}$ and from this r which is the period of our function using the following steps:

For each convergent γ of φ do the following:

1. Compute γ as fraction $\frac{d}{r}$.
2. Stop if $r \leq 2^n$ and this $\frac{d}{r}$ is $\frac{1}{2^{n+1}}$-close to $\frac{c}{2^n}$ and return r.

Note: It can happen, that the resulting fraction does not have the right r in the denominator, because $\frac{d}{r}$ was simplified (if numerator and denominator shared a common factor). But the probability of this happening is sufficiently small and already included in the probability in Theorem 9.3.

This completes the postprocessing of Shor's algorithm.

9.5 Constructing the DFT

So far we have described everything necessary for Shor's algorithm, but only described the matrix representation of the DFT_M. We will now take a closer look into implementing the DFT_M as a quantum circuit. Since we only use the DFT_M for Shor's algorithm so far, we will only look at $M = 2^n$, which is the DFT applied on n qubits.

To start the circuit, we recall the definition of the DFT_{2^n} from Definition 9.1: $\mathrm{DFT}_{2^n} := \frac{1}{\sqrt{2^n}}(\omega^{kl})_{kl}$ with $\omega := e^{2\pi i/2^n}$. To apply the DFT_{2^n} to a quantum state $|j\rangle$ we calculate

$$\mathrm{DFT}_{2^n}|j\rangle = \frac{1}{\sqrt{2^n}}\sum_k e^{2\pi ijk2^{-n}}|k\rangle$$

We can rewrite this as follows:

$$\begin{aligned}\mathrm{DFT}_{2^n}|j\rangle =& \frac{1}{\sqrt{2^n}}\sum_k e^{2\pi ijk2^{-n}}|k\rangle\\ =& \frac{1}{\sqrt{2^n}}\sum_{k_1}\dots\sum_{k_n} e^{2\pi ij(\sum_l k_l 2^{-l})}|k_1 \dots k_n\rangle\\ =& \frac{1}{\sqrt{2^n}}\sum_{k_1}\dots\sum_{k_n}\bigotimes_{l=1}^{n} e^{2\pi ij(k_l 2^{-l})}|k_l\rangle\\ =& \frac{1}{\sqrt{2^n}}\bigotimes_{l=1}^{n}\sum_{k_l} e^{2\pi ij(k_l 2^{-l})}|k_l\rangle\\ =& \frac{1}{\sqrt{2^n}}\bigotimes_{l=1}^{n}(|0\rangle + e^{2\pi ij2^{-l}}|1\rangle)\\ =& \frac{1}{\sqrt{2^n}}\bigotimes_{l=1}^{n}(|0\rangle + e^{2\pi i0.j_{n-(l-1)}\cdots j_n}|1\rangle)\\ =& \bigotimes_{l=1}^{n}\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i0.j_{n-(l-1)}\cdots j_n}|1\rangle)\end{aligned}$$

The expression $0.j$ expresses a binary fraction (e.g. $0.101 = \frac{1}{2} + \frac{1}{8} = \frac{5}{8}$).

With this we have shown, that we can write $\mathrm{DFT}_{2^n}|j\rangle$ as the following tensor product of quantum states

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i0.j_n}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i0.j_{n-1}j_n}|1\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i0.j_1\cdots j_n}|1\rangle)$$

From this rewritten tensor product, we can get an idea on how to construct the quantum circuit for the DFT_{2^n}. Namely, we can construct a quantum circuit for each element of the tensor product and from this build the general circuit.

For this, we segment the tensor product into different elements ψ as follows:

$$\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i0.j_n}|1\rangle)}_{\psi_1} \otimes \underbrace{\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i0.j_{n-1}j_n}|1\rangle)}_{\psi_2} \otimes \cdots \otimes \underbrace{\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i0.j_1\cdots j_n}|1\rangle)}_{\psi_n}$$

We also introduce a new gate R_k which is defined by the following matrix:

$$R_k := \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

To understand the construction of the circuit from these elements, we will look at an example for $n = 3$ first:

Example: Construction of the DFT circuit for $n = 3$

We start by building the tensor product for $n = 3$. The input for the DFT circuit is $|j\rangle = |j_1 j_2 j_3\rangle$. Using the formula from above, we can write the result of $\mathrm{DFT}_{2^3}|j\rangle$ as the following tensor product:

$$\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_3}|1\rangle)}_{\psi_1} \otimes \underbrace{\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle)}_{\psi_2} \otimes \underbrace{\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle)}_{\psi_3}$$

First we construct the ψ_3 element. Contrary to the intuition, we use the top wire containing $|j_1\rangle$ for this. We use a Hadamad-gate to bring $|j_1\rangle$ into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{j_1}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1}|1\rangle)$. This looks close to ψ_3 already, we now need to add the last two decimal places $j_2 j_3$ to the state. For this we use R_2 and R_3. We apply R_2 controlled by the wire j_2 and R_3 controlled by the wire j_3. This means, that we only apply the R-gate, if the corresponding wire contains a 1. You can see this written as a quantum circuit at the figure below. After applying R_2 we have the state $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle)$ and after applying R_3 we have the state $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle)$ on the top wire. This is the same as ψ_3, so we are done on the first wire (We are at slice 3 in the figure below at this point).
The next step is to construct the ψ_2 state on the middle wire. We again use a Hadamad-gate to bring $|j_2\rangle$ into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2}|1\rangle)$. We now need to include the last decimal point j_3, for which we use R_2 again, this time controlled by j_3. The resulting superposition is now $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle)$, which is ψ_2.
On the bottom wire, we can just do a Hadamad-gate to bring $|j_3\rangle$ into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_3}|1\rangle)$. We then have ψ_1 on the bottom wire. The full circuit is described in this figure:

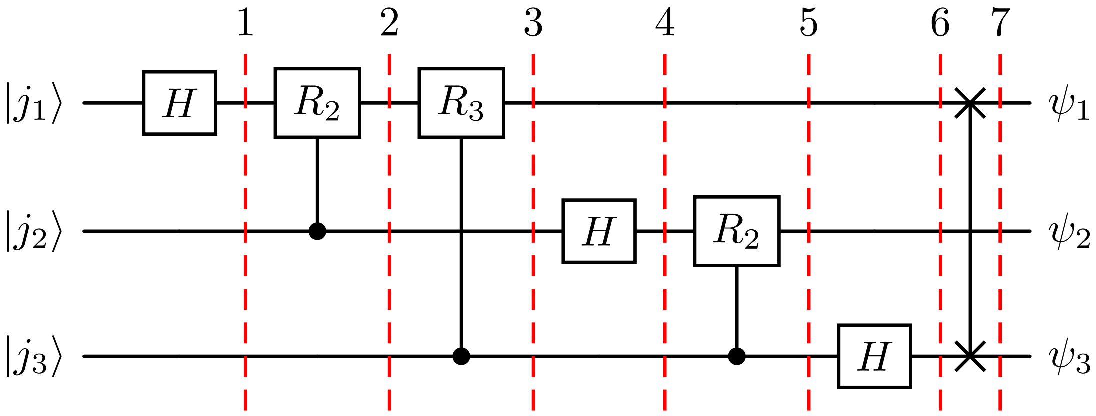

Figure 9.2: The DTF for three qubits

When applying this circuit, we get the state $\psi_3 \otimes \psi_2 \otimes \psi_1$ as a result. This very close to our desired state $\psi_1 \otimes \psi_2 \otimes \psi_3$, just the order of the wires is flipped. To solve this, we apply a SWAP onto all wires, which flips the order of the wires an delivers the correct output for DFT_{2^3}.

The more general approach to construct the DFT_{2^n} as a quantum circuit with n qubits works as follows:

1. Initialize wires with input $|j\rangle$, so that $|j_1\rangle$ is on the top wire and $|j_n\rangle$ is on the bottom wire. Note, that this is not part of the circuit yet.
2. Start with the top wire. For each wire j_i do the following:
    1. Apply a Hadamad-gate on the wire j_i.
    2. For each wire j_k below the current wire j_i (with $i < k \leq n$), add a R_{k-i+1}-gate controlled by j_k. Start with $k = i + 1$ (if $i < n$, else stop).
3. Perform a SWAP to flip all the wires. This means, that the first wire is swapped with the last wire, the second wire is swapped with the second to last wire and so on.

Note: If the the output of the DFT circuit is measured right after applying it (as in Shor's algorithm) or if the rest of the algorithm allows for it, it is more efficient to perform the SWAP classically, since this is considered to be the cheaper operation.

The more general layout of the quantum circuit for the DFT_{2^n} with the SWAP is shown in this figure.

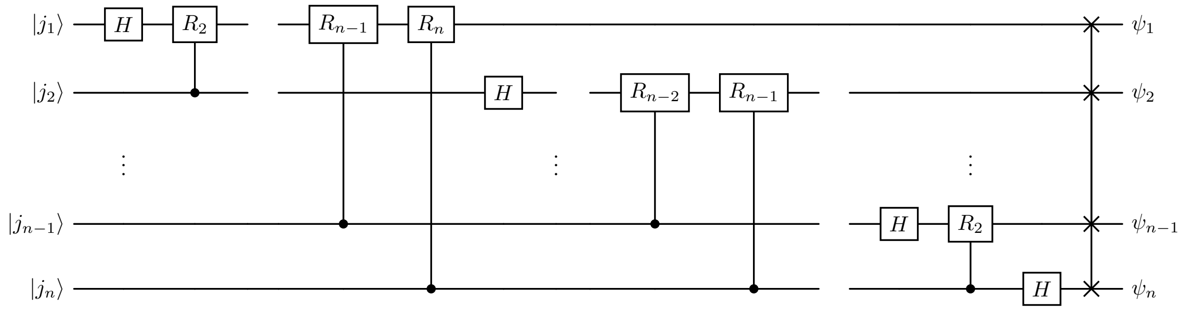

Figure 9.3: The DTF for n qubits

10 Grover's algorithm

Another well known quantum algorithm is Grover's algorithm for searching. It was developed by Lov Grover in 1996.

Grover's algorithm takes a function $f : \{0,1\}^n \to \{0,1\}$, where exactly one x_0 exists, such that $f(x_0) = 1$. The goal is to find x_0.

There are a number of interesting problems, which can be reduced to this general definition. One of these problems is the breaking of a (symmetric) encryption. The function f would take a key as an input and output a 1, if the decryption is successful.

Classically, finding this x_0 takes approximately 2^n steps (when simply bruteforcing the function). Using Grover's algorithm, we can reduce this runtime to approximately $2^{\frac{n}{2}}$ steps. As an example, a 128-bit encryption would only take about 2^{64} steps to break it, instead of about 2^{128} steps for the classical bruteforce.

10.1 Preparations

To construct Grover's algorithm, we first need to introduce two new gates V_f and FLIP_*.

10.1.1 Constructing the oracle V_f

In the previous algorithms, we have learned that we can implement a function f as a unitary U_f with $U_f |x, y\rangle = |x, y \oplus f(x)\rangle$. We construct a different unitary called V_f from this, which has the following behavior:

$$V_f |x\rangle = \begin{cases} -|x\rangle & \text{if } f(x) = 1 \\ |x\rangle & \text{else} \end{cases}$$

We can construct V_f from U_f using the following circuit:

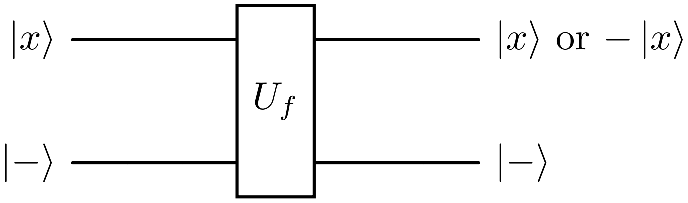

Figure 10.1: The circuit for V_f

The bottom wire can be discarded, since it always contains a $|-\rangle$ and thus is not entangled with the upper wire.

10.1.2 Constructing FLIP_*

As a second ingredient for Grover's algorithm, we define a unitary called FLIP_*. This unitary does nothing, if it is applied on the uniform superposition $|*\rangle$. For any other quantum state ψ with ψ orthogonal to $|*\rangle$ it maps ψ to $-\psi$. The uniform superposition $|*\rangle$ simply denotes the superposition over all classical possibilities $2^{-\frac{n}{2}} \sum_x |x\rangle$. We can construct this FLIP_* by the following quantum circuit:

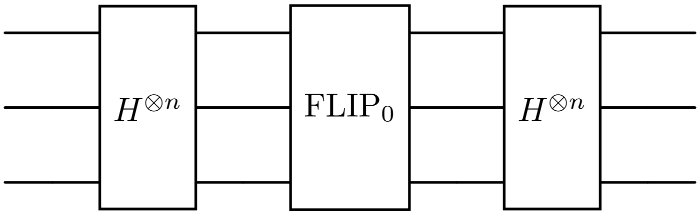

Figure 10.2: The circuit for FLIP_*

FLIP_0 is here definied by the unitary

$$\text{FLIP}_0 |x\rangle = \begin{cases} |0\rangle & \text{if } x = 0 \\ -|x\rangle & \text{else} \end{cases}$$

10.2 The algorithm for searching

The actual algorithm takes a function $f : \{0,1\}^n \to \{0,1\}$ and outputs an x_0 with $f(x_0) = 1$. For simplicity, we assume that there is only one x_0 for which $f(x_0) = 1$ holds and for each other $x \neq x_0$ it holds that $f(x) = 0$.

With the two new unitaries V_f and FLIP_* defined, we can construct the circuit for Grover's algorithm, which is shown in the following figure:

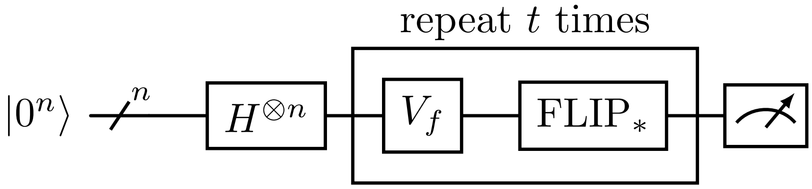

Figure 10.3: The quantum circuit for Grover's algorithm

The algorithm works as follows:

1. We start with a $|0\rangle$ entry on every qubit.
2. We bring the system into the superposition over all entries by applying $H^{\otimes n}$. The quantum state is then $2^{\frac{-n}{2}} \sum_x |x\rangle$ which we also call $|*\rangle$.
3. We apply the unitary V_f.
4. We apply the unitary FLIP_*.
5. We repeat steps 3 and 4 t times, and then do a measurement.

The measurement in step 5 will then give us x_0 with high probability.

10.2.1 Understanding the algorithm for searching

When looking at the quantum circuit, it is not completely intuitive why the algorithm gives the correct result. We therefore now look into what is happening in each step.

The desired quantum state after the algorithm finishes is $|x_0\rangle$. At the beginning of the algorithm, we bring the system into the uniform superposition $|*\rangle = 2^{\frac{-n}{2}} \sum_x |x\rangle$. We know that

$|x_0\rangle$ is part of this superposition, therefore we can rewrite $|*\rangle$ as follows

$$|*\rangle = 2^{-\frac{n}{2}} \sum_x |x\rangle = \frac{1}{\sqrt{2^n}} \underbrace{|x_0\rangle}_{good} + \sqrt{\frac{2^n - 1}{2^n}} \underbrace{\sum_{x \neq x_0} \frac{1}{\sqrt{2^n - 1}} |x\rangle}_{bad}$$

So the current state can be seen as a superposition of a “good” state *good* and a “bad” state *bad*.

The geometric interpretation of this superposition can be drawn as follows:

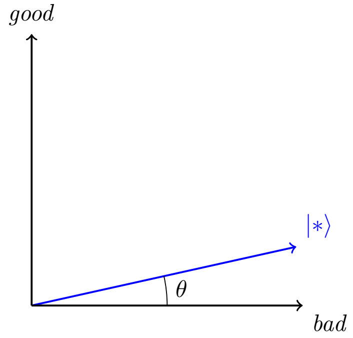

Figure 10.4: Geometric interpretation of $|*\rangle$

The angel θ denotes, how “good” the resulting outcome will be. If $\theta = 0$, the state is completely bad, if $\theta = \frac{\pi}{2}$, the state is completely good.

We can calculate $\cos\theta = |\langle * | bad\rangle| = \frac{\sqrt{2^n - 1}}{\sqrt{2^n}}$. From this we can derivate that the angle θ is $\cos^{-1}\sqrt{\frac{2^n-1}{2^n}}$ at the beginning, which is approximately $\sqrt{\frac{1}{2^n}}$.

We now apply V_f on this quantum state. This will negate the amplitude off our desired $|x_0\rangle$ and not change the amplitude to the rest of the state.

$$V_f\,|*\rangle = -\frac{1}{\sqrt{2^n}} \underbrace{|x_0\rangle}_{\text{good}} + \sqrt{\frac{2^n - 1}{2^n}} \underbrace{\sum_{x \neq x_0}}_{\text{bad}} |x\rangle$$

This looks like this in the geometric interpretation:

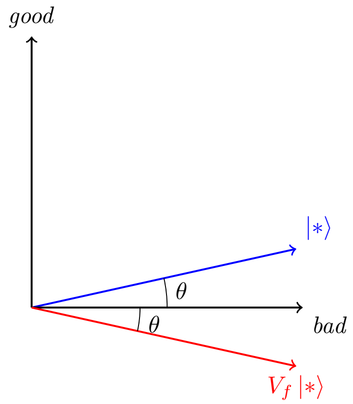

Figure 10.5: Geometric interpretation after V_f

We can see that by applying V_f, we mirror the vector across the *bad* axis.

After V_f, we apply the FLIP_* operation on the quantum state. Since FLIP_* does nothing on the $|*\rangle$ entries and negates the amplitude of any vector orthogonal to it, FLIP_* mirrors the vector across $|*\rangle$. This can be seen in the following figure:

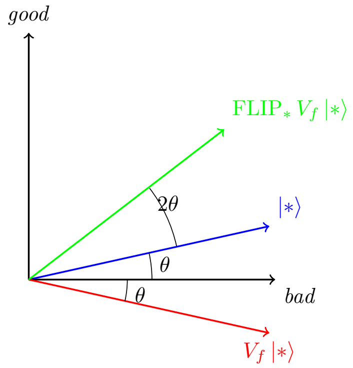

Figure 10.6: Geometric interpretation after FLIP_*

All in all, we have seen that by applying V_f and FLIP_*, we can increase the angle of the quantum state in relation to the "good" and "bad" states by 2θ. Therefore two reflection give rotation. By repeating this step often enough, we can get the amplitude of $|x_0\rangle$ close to 1.

To be more precise: Since we know θ and we know that we will increase the *good*-ness of our quantum state by 2θ each time, we can calculate that only t iterations are necessary with

$$t \approx \frac{\frac{\pi/2}{\theta} - 1}{2} \approx \frac{\pi}{4\theta} \approx \frac{\pi}{4} \cdot \sqrt{2^n}$$

Grover's algorithm therefore takes $O(\sqrt{2^n})$ steps, where an evaluation of the circuit counts as one step.

11 Quantum Physics

To further understand, how quantum computers work, we take a look at the basics of quantum physics.

11.1 Wave function

The first concept we look into is the wave function of quantum mechanics. For this we will look at the experiment "particle in a well". To keep the math simple, we assume the space to be 1-dimensional, so the position of the particle is confined to a line.

In this experiment, we have one particle and a potential, which is denoted by a function $V : \mathbb{R} \to \mathbb{R}$. This function maps a position of a particle in $\mathbb{R}$ to the energy which is needed to hold the particle in that position.

Classically a state of a system at time t is described by the position $x(t) \in \mathbb{R}$ and the momentum $p(t) \in \mathbb{R}$.

In the quantum world, we have a wave function $\psi_t(x)$ with $\psi_t : \mathbb{R} \to \mathbb{C}$ under $t \in \mathbb{R}$, which takes the position of a particle as an input and outputs the amplitude of that particle, with t as the time.

To calculate the probability of a particle being in the interval $[a, b]$ at time t_0, we can use the integral over the wave function:

$$\Pr[\text{Particle is in } [a, b] \text{ a time } t_0] = \int_a^b |\psi_{t_0}(x)|^2 dx$$

From this we can see that the integral $\int |\psi_{t_0}|^2 dx = 1$. The momentum is not needed for the wave function.

The inner product of two wave functions $\mathbb{R} \to \mathbb{C}$ is given by

$$\langle \psi | \phi \rangle := \int \overline{\psi(x)} \cdot \phi(x) dx$$

The norm of a wave function is given by

$$\|\psi\|^2 := \langle\psi|\psi\rangle = \int |\psi(x)|^2 dx$$

In general, the wave function can have a different domain, e.g. $\psi_t : \mathbb{R}^3 \to \mathbb{C}$ for a particle in 3D-space. Everything below works analogously in that case.

11.2 Energy / Hamiltonian

Given a particle in a state, this particle has some energy. Classically this energy is calculated from the potential and the momentum, with

$$\text{Energy} := \underbrace{V(x)}_{\text{potential energy}} + \underbrace{\frac{p^2}{2m}}_{\text{kinetic energy}}$$

In the quantum world, we can calculate the potential energy using the wave function as follows:

$$\text{potential energy} := \int \underbrace{|\psi(x)|^2}_{\text{probability at pos. } x} \cdot V(x) dx$$

The kinetic energy can be calculated as follows:

$$\text{kinetic energy} := \int -\overline{\psi(x)} \cdot \frac{\hbar}{2m} \cdot \frac{\partial^2 \psi(x)}{\partial x^2} dx$$

$\hbar$ is the reduced Planck constant. We will not go further into reasoning for this definition.

From this we can calculate the energy in the quantum context:

$$\begin{aligned}\text{energy} &:= \int \overline{\psi(x)} \left(-\frac{\hbar}{2m} \frac{\partial^2 \psi(x)}{\partial x^2} + V(x)\psi(x) \right) dx \\ &= \int \overline{\psi(x)} (H\psi)(x) dx \\ &= \langle \psi, H\psi \rangle\end{aligned}$$

The operator H is called a Hamiltonian and is an operator that maps wave functions to wave functions, such that $\langle \psi, H\psi \rangle$ is the energy. H maps ψ to $H\psi$ which is defined as:

$$(H\psi)(x) = -\frac{\hbar}{2m} \frac{\partial^2 \psi(x)}{\partial x^2} + V(x)\psi(x) \tag{11.1}$$

Note that $\hbar$ is the reduced plank constant and m is the mass. Also note that other physical systems (e.g. when there are more particles, or when there is not just a simple potential) may have a different definition of the Hamiltonian.

11.3 Schrödinger equation

11.3.1 Time-dependent Schrödinger equation

The time-dependent Schrödinger equation is denoted by:

$$i\hbar \frac{\partial \psi_t(x)}{\partial t} = (H\psi_t)(x)$$

From this we see that the time development of ψ_t is determined by ψ_t at that moment (via $H\psi_t$).

11.3.2 Time-independent Schrödinger equation

For the time-independent Schrödinger equation, we try to find a wave function $\psi : \mathbb{R} \to \mathbb{C}, \psi \neq 0$ and an energy E such that

$$H\psi = E\psi$$

This means roughly that we try to find ψ that has the same energy everywhere. That is that we try to find a ψ with energy E (since the energy is $\langle \psi | H\psi \rangle = \langle \psi | E\psi \rangle = E \langle \psi | \psi \rangle = 1$) and not a superposition of different energies.

Such a ψ is useful, since by the time-dependent Schrödinger equation, if $H\psi = E\psi$, then

$$i\hbar \frac{\partial \psi_t(x)}{\partial t} = H\psi = E\psi_{t_0}(x)$$

Solving this differential equation gives us ψ with

$$\psi_t(x) = e^{-iEt/\hbar}\psi_{t_0}(x)$$

So all in all: If ψ_0 is a solution of the time-independent Schrödinger equation, then $\psi_t(x) = e^{-iEt/\hbar}\psi_0(x)$ is the solution to the time-dependent Schrödinger equation with initial condition ψ_0 (at time $t = 0$).

Given any ψ_0, we can try to rewrite ψ_0 as $\psi_0 = \sum_k a_k \psi^k$, where ψ^i, E_i are solutions of the time-independent Schrödinger equation and then the solution of the time-dependent Schrödinger equation is

$$\psi_t(x) = \sum_k a_k e^{-iEt/\hbar}\psi^k(x)$$

Note that with ψ^k, k is an index and not an exponent.

11.4 Infinite square well

Our goal is to solve the time-independent Schrödinger equation for the infinite square well. The infinite square well is defined to have a potential of $V(x) = 0$ in the range $[0, 1]$ and $V(x) = \infty$ otherwise.

We now try to solve the time-independent Schrödinger equation $H\psi = E\psi$, which for the Hamiltonian from Equation 11.1 becomes:

$$-\frac{\hbar}{2m}\frac{\partial^2\psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi$$

Since the potential of the particle is ∞ outside of the range $[0, 1]$, we know that $\psi(x) = 0$ outside of $[0, 1]$, because there is no infinite energy. From this we also know that $\psi(0) = 0$ and $\psi(1) = 0$ by continuity.

Since either $\psi(x)$ or $V(x)$ are always 0 for any x, we need to solve the term

$$-\frac{\hbar}{2m}\frac{\partial^2\psi(x)}{\partial x^2} = E\psi$$

We assume for simplicity that $m = 1$ and $\hbar = 1$ and we don't specify units like meter, seconds, joule etc. All solutions to $-\frac{1}{2}\frac{\partial^2\psi(x)}{\partial x^2} = E\psi$ have the form

$$\psi = A\sin(\gamma x) + B\cos(\gamma x)$$

with $E = \frac{\gamma^2}{2}$ for any $A, B \in \mathbb{C}$ and $\gamma \in \mathbb{R}$. Since $\psi(0) = 0$, we know that $B = 0$. Therefore $\psi = A\sin(\gamma x)$. The scaling factor A has to fulfill $A \neq 0$ since we only look for non-zero wave functions.

Since $\psi(1) = 0$, we know that $A\sin(\gamma) = 0$. Therefore γ needs to be a multiple k of π and also $k > 0$ because we want non-zero solutions $\psi \neq 0$. So we set $\gamma := (k + 1)\pi$ for integers $k \geq 0$. With $E = \frac{\gamma^2}{2}$ we get:

$$E_k = \frac{(k+1)^2\pi^2}{2}$$

and also that

$$\psi_k(x) = A\sin((k + 1)\pi x)$$

For the rest of our lecture, we will set A to be $A = \sqrt{2}$ because we are only interested in normalized wave functions.

So all in all: In the infinite square well the wave functions with no energy-superposition are $\sin((k + 1)\pi x)$ with $E_k = \frac{(k+1)^2\pi^2}{2}$.

There is one important thing to note here: With the infinite square well, there are discrete energy levels, so for each energy level k, we have an energy $E_k = (k + 1)^2\frac{\pi^2}{2}$ and no other energy levels (e.g $\frac{1.4\pi^2}{2}$) exist. $E_k > 0$ also holds, so we can never have 0-energy. This means that, different from the classical case, the particle can never be fully at rest. We will write $|k\rangle := \psi_k$.

We can now write each state as

$$\sum_{k\in\mathbb{N}} a_k |m\rangle \tag{11.2}$$

If we ignore energies above a certain level, we get a system $|1\rangle, |2\rangle, |3\rangle, ..., |N\rangle$, where any ψ can be written as $\sum a_k |k\rangle$. That is useful if we do not want to think about infinite dimensional systems.

12 From Quantum Physics to a Quantum Computer

In the previous chapter, we have learned about the fundamentals of quantum physics. We now relate this to quantum computers.

So far we have seen solutions of the time-independent Schrödinger equation $\psi_k(x) = \sqrt{2}\sin((k+1)\pi x)$ and $E_k = \frac{(k+1)^2\pi^2}{2}$. Note that we still assume $\hbar = 1$ and $m = 1$.

We can now combine physics and quantum computer science by setting

$$\begin{gathered} |0\rangle := \psi_0 \\ |1\rangle := \psi_1 \\ ... \end{gathered}$$

For one qubit, we just look at ψ_0 and ψ_1 and ignore all other wave functions. Note that this can lead to errors, since those other wave functions still exists and interact with our system, even though they might have a very small probability.

To fully construct our one qubit quantum computer, we need to be able to perform three basic operations:

1. We need to initialize our qubit with $|0\rangle$. This is also called cooling.
2. We need to be able to apply a unitary on the qubit.
3. We need to be able to measure the qubit.

We look into how to construct a unitary on our particle. The cooling and measuring operations are out of scope od this chapter. We know that the time evolution of our quantum computer is $|0\rangle \mapsto e^{-i\frac{\pi^2}{2}t}|0\rangle$ and $|1\rangle \mapsto e^{-i2\pi^2 t}|1\rangle$.

This can be seen as a unitary U_t, which is depending on t written as the matrix:

$$U_t = \begin{pmatrix} e^{-i\frac{\pi^2}{2}t} & 0 \\ 0 & e^{-i2\pi^2 t} \end{pmatrix}$$

Using $t = \frac{6}{\pi}$, we get the unitary

$$U_{\frac{6}{\pi}} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

which is equal to the unitary $-Z$. This means that the unitary $-Z$ is applied every $\frac{6}{\pi}$ steps "automatically". The factor -1 does not make a physical difference as it is a global phase factor, so the $-Z$ is physically equal to the Z gate.

But how do we get new unitaries which are not Z? Since U_t is dependent on the evolution of ψ, which is dependent of H, which is dependent of the potential $V(x)$, we can change the potential $V(x)$ to get a different U_t.

The problem is that when changing the potential $V(x)$, we need to solve the differential equation again. Luckily, there is a trick for that which avoids thinking about wave functions to much: If we have wave functions $|k\rangle$ with $k \in \{1, \dots, N\}$ and $\| |k\rangle \| = 1$ and also $\langle k|l\rangle = 0$ for $k \neq l$ so $|k\rangle$ and $|l\rangle$ are orthogonal to each other, we can rewrite H as a linear operator on the wave functions written as $\operatorname{span}\{|k\rangle : k = 1 \dots N\}$, therefore we can write H as a matrix:

$$H = \begin{pmatrix} E_1 & 0 & 0 \\ 0 & E_2 & 0 \\ 0 & 0 & \ddots \end{pmatrix} = \begin{pmatrix} \frac{\pi^2}{2} & 0 & 0 \\ 0 & 2\pi^2 & 0 \\ 0 & 0 & \ddots \end{pmatrix}$$

For this representation of H, we immediately get $H|0\rangle = \frac{\pi^2}{2}$, $H|1\rangle = 2\pi^2$ and so on, so nothing has changed except that H is represented more nicely.

We can now use a helpful theorem to get a solution for the differential equation.

Theorem 12.1. *The differential equation*

$$i\hbar\frac{\partial\psi_t}{\partial t} = H\psi_t$$

with H as a $N \times N$ matrix and initial state ψ_0 as an N-dim vector has the solution

$$\psi_t = e^{-iHt/\hbar}\psi_0$$

We use this theorem for our one qubit computer. The goal is to change the potential by some δV and from this get a different U_t.

We try this by changing the potential to $\delta V = \frac{9\pi^2}{16}(\frac{1}{2} - x) \cdot 1000$ for $x \in [0, 1]$ and $\delta V = 0$ for $x \notin [0, 1]$.

We rewrite δV as a matrix. To do so, we try to find a matrix in the base $|0\rangle, |1\rangle$. If $\delta V |0\rangle = a|0\rangle + b|1\rangle$ and $\delta V |1\rangle = c|0\rangle + d|1\rangle$, then

$$\delta V = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

Since $|0\rangle$ and $|1\rangle$ are orthonormal, we know that $\langle 0|\delta V|0\rangle = \langle 0|a|0\rangle + \langle 0|b|1\rangle = a + 0 = a$. We get b, c and d similar and from this the following matrix:

$$\delta V = \begin{pmatrix} \langle 0|\delta V|0\rangle & \langle 0|\delta V|1\rangle \\ \langle 1|\delta V|0\rangle & \langle 1|\delta V|1\rangle \end{pmatrix}$$

We calculate each of the entries of the matrix separately:

$$\begin{aligned}
\langle 0|\delta V|0\rangle &= \int_0^1 \sqrt{2}\sin(\pi x) \cdot \delta V(x) \cdot \sqrt{2}\sin(\pi x) dx = 0 \\
\langle 1|\delta V|1\rangle &= \int_0^1 \sqrt{2}\sin(2\pi x) \cdot \delta V(x) \cdot \sqrt{2}\sin(2\pi x) dx = 0 \\
\langle 1|\delta V|0\rangle &= \int_0^1 \sqrt{2}\sin(2\pi x) \cdot \delta V(x) \cdot \sqrt{2}\sin(\pi x) dx = 1000 \\
\langle 0|\delta V|1\rangle &= \int_0^1 \sqrt{2}\sin(\pi x) \cdot \delta V(x) \cdot \sqrt{2}\sin(2\pi x) dx = 1000
\end{aligned}$$

From this we get δV written as a matrix with readable numbers:

$$\delta V = \begin{pmatrix} 0 & 1000 \\ 1000 & 0 \end{pmatrix}$$

We can now add this matrix to the Hamiltonian H to get the Hamiltonian H' which is the Hamiltonian under the changed potential by calculating

$$H' = H + \delta V = \begin{pmatrix} \frac{\pi^2}{2} & 1000 \\ 1000 & 2\pi^2 \end{pmatrix}$$

We now need to solve Schrodinger equation with this new H'.

If we would solve the Schrodinger equation with δV as a Hamiltonian using Theorem 12.1, we would get the unitary $U'_t = e^{-i\delta V t}$. After $t = \frac{\pi}{2000}$ this would be the unitary $\begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix} = -iX$, essentially an X gate.

If we now apply $H' = H + \delta V$ as a Hamiltonian, we would get $U'_t = e^{-iH't}$ and this is

$$U'_t = \begin{pmatrix} e^{-i\frac{\pi^2}{2}t} & e^{-i1000t} \\ e^{-i1000t} & e^{-i2\pi^2 t} \end{pmatrix}$$

so approximately the unitary $\begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix} = -iX$ up to about 2% error.

13 Ion-based quantum computers

So far we have looked at the principles of quantum mechanics and how to transfer these principles to our mathematical description of quantum computing. While there are many different approaches on how to actually build a quantum computer, which are researched at the moment, we will only look at one approach. This approach is based on trapped ions.

13.1 Electron in an atom

We look at a single atom with a nucleus with a positive charge and a single electron with negative charge “orbiting” the nucleus.

The electromagnetic field generated by the nucleus is essentially a potential well for the electron, since the electron is drawn to the nucleus and the potential of the electron rises with bigger distance from the nucleus. We simplify a lot here and ignore ,e.g., the spin.

We can solve the time-independent Schroedinger equation for this setup and by solving this, we will get the wave functions that are the energy eigenstates of the Hamiltonian. These wave functions are called *orbitals*.

We can use a single atom as a qubit, where we define one of the energy eigenstates as $|0\rangle$ and a different eigenstate as $|1\rangle$.

In the following, we will specifically use electrically charged atoms, called ions, because they are easier to capture.

13.2 Setup for the ion traps

The setup for our quantum computer looks as follows:

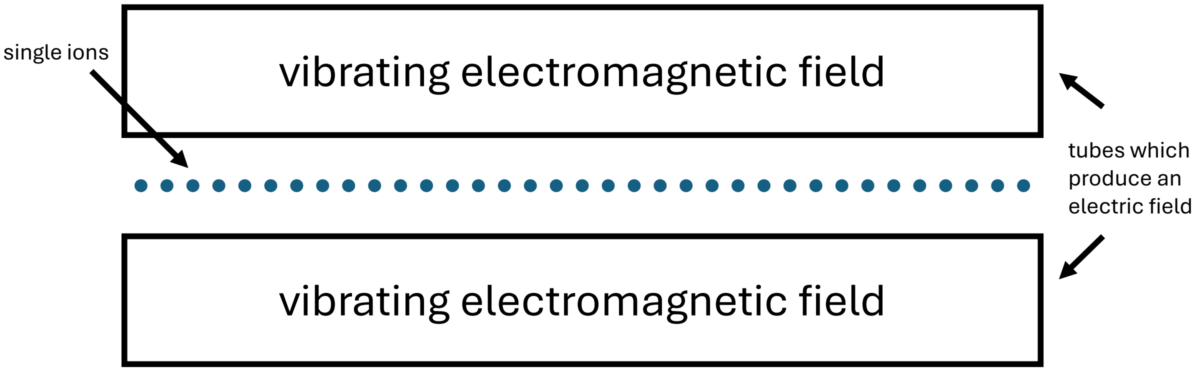
single ions
vibrating electromagnetic field
tubes which produce an electric field
vibrating electromagnetic field

Figure 13.1: Setup for the ion based quantum computer

In the previous chapter, we have learned that we need to be able to perform three different operations to build a quantum computer:

1. We need to initialize (cool) our qubit.
2. We need to be able to apply a unitary on the qubit.
3. We need to be able to measure the qubit.

13.2.1 Cooling

We first look into cooling our system. For cooling, we use a useful fact: If $E_i < E_j$ are different possible energy levels, an ion is in the energy level E_i and then hit by a photon that has the energy $E_j - E_i$, the ion will go to energy level E_j.

13.2.1.1 Doppler cooling

In our initial setup, we have an ion vibrating back and forth because it has too much energy. We shine a laser on it with slightly less energy than what is need for a transition. The energy of the photon of this laser is denoted by $E = \hbar \cdot \omega$. When the ion moves towards the photon, the photon has a higher frequency from the point of view of the ion (Doppler effect). This means that the photon has a higher energy and therefore is more likely to be absorbed.

So by shining a laser on the ion, the photons of the laser “push” the ion, when it “swings” *towards* the laser similar to a pendulum, where the pendulum gets a pushback with just enough energy so it stops. This reduces the vibrations energy down to a certain level.

13.2.1.2 Sideband cooling

Using the doppler cooling, we have reduced the vibration energy, but the electrons may still be excited. We now look at another technique called sideband cooling, which will set the energy of the electrons to a specific energy level E_0.

The electron can have any energy level E_i. If this energy level is pretty low, the possibility of a spontaneous emission of a photon, which would reduce the energy to a lower level is also quite low. So an electron with energy level E_1 or E_2 will probably not change to level E_0. If the energy level is big enough (we will call this energy level E_{big}), the probability of a spontaneous emission of a photon, which would reduce the energy to a lower level is quite high. So when this spontaneous emission happens, the electron will reach an energy level of ,e.g., E_0, E_1 or E_2.

Our goal is to get the energy level to E_0. We know that the energy level of the electrons with E_{big} will come down eventually, so we shine a laser with the energy per photon of $E_{\text{big}} - E_1, E_{\text{big}} - E_2, ...$ but not with $E_{\text{big}} - E_0$. This will "shoot" all the low energy electrons from $E_1, E_2, ...$ to a higher lever where they will either fall down to $E_1, E_2, ...$ where they will be energized again and the process is repeated, or they fall into E_0 which is our desired energy level.

Using the sideband and the doppler cooling together, we can cool the vibration and electron excitation. This means that all the ions are in the state $|0\rangle$ and the vibrations energy is also in the state $|0\rangle$.

13.2.2 Unitaries

Next we look into applying unitaries, also called gates. We first show how to create single qubit gates and then look into creating multi qubit gates.

13.2.2.1 Single qubit gates

We have already discussed in Chapter 12 that single qubit gates can be created using a different Hamiltonian. In the ion-trap, if we don't do anything, we have some Hamiltonian $H \approx 0$ (For our purposes we can assume that nothing happens until we apply a laser). If there is a gap of energy ΔE between E_0 and E_1 with $\Delta E = E_1 - E_0 = \hbar\omega$ and we shine a laser with an angular frequency of $\omega + \varphi$ with φ close to ω on it, then the Hamiltonian changes to

$$H_\varphi = C \cdot \begin{pmatrix} 0 & e^{-i\varphi} \\ e^{i\varphi} & 0 \end{pmatrix}$$

$C > 0$ denotes a constant here. Applying the laser for time t will get us the unitary $U_{\varphi,t}$ with

$$U_{\varphi,t} = e^{-iH_\varphi t}$$

By choosing the right frequency, we can get all sorts of single qubit gates, especially the H, X, Y, Z, $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ and $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$ gates.

13.2.2.2 Multi qubit gates

After we have successfully constructed a single qubit gate, we look into creating multi qubit gates. We specifically look into two qubit gates, since we can create bigger gates from these.

We remember that the initial state of the k-th ion is described by some quantum system with basis $|0\rangle, |1\rangle, ...$ and the overall vibration is described by another quantum system $|0\rangle, |1\rangle, ...$

In this case, we focus only on the vibrational states $|0\rangle, |1\rangle, |2\rangle, ...$ and internal states $|0\rangle, |1\rangle$. This means that we have different states (Here we write the vibrational state first)

$$\begin{array}{ccc} |00\rangle & |10\rangle & |20\rangle \\ |01\rangle & |11\rangle & |21\rangle \end{array}$$

Consider the energy gap $\Delta E = E_{20} - E_{11}$. The angular frequency corresponding to that gap is $\omega = \frac{\Delta E}{\hbar}$. If we shine a laser with frequency $\omega + \varphi$ then some Hamiltonian is applied to $|20\rangle, |11\rangle$ with

$$H_\varphi = \left(\begin{array}{cc:cc:cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-i\varphi} & 0 \\ \hdashline 0 & 0 & 0 & e^{i\varphi} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

With $\varphi = 0$ and $t = \pi$, we get a unitary

$$U = e^{-iH_0\pi} = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & -1 & & \\ & & & & -1 & \\ & & & & & 1 \end{pmatrix}$$

If we only consider the first 4 entries of this unitary, it represents a C-Z (controlled Z) gate. The other entries can be ignored, since we only "use" the first two energy levels.

So all in all we have created a controlled Z gate by applying a laser to the $|20\rangle, |11\rangle$ gap.

We now apply a laser to the $|01\rangle, |10\rangle$ energy gap. The resulting hamiltonian has the form

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & e^{-i\varphi} & 0 \\ 0 & e^{i\varphi} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

With $\varphi = \frac{3}{2}\pi$ and $t = \frac{\pi}{2}$, we get the unitary SWAPPY denoted by

$$\text{SWAPPY} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

We call it SWAPPY because it is almost the SWAP gate, except for a minus sign. We can get SWAPPY† with $\varphi = \frac{3}{2}\pi$ and $t = \frac{3}{2}\pi$

So we have created the unitaries C-Z, SWAPPY, SWAPPY†. From the single qubit setup, we can also retrieve the H gate.

From these gates, we construct the following circuit, which is equal to CNOT:

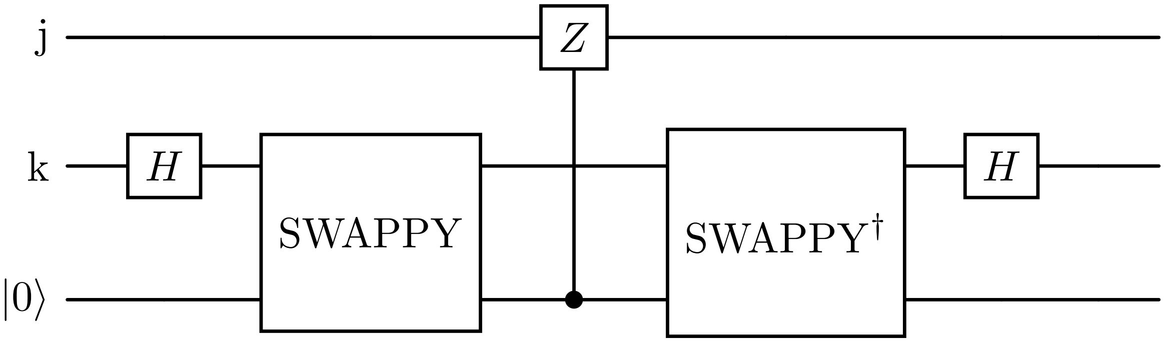

Figure 13.2: Circuit for CNOT

So all in all, we can construct the unitaries CNOT, H, X, Y, Z S and T using ion based quantum computers.

13.2.3 Measurements

Finally we now look into performing a measurement on an ion-based quantum computer. So far we have defined that the quantum state $|0\rangle$ is represented by the energy level E_0 and the quantum state $|1\rangle$ is represented by the energy level E_1.

To perform a measurement, we also use an auxiliary energy level $E_{\text{aux}} > E_0, E_1$. Let ω be the frequency of a photon with energy $E_{\text{aux}} - E_0 (E_{\text{aux}} - E_0 = \Delta\omega)$

We now shine a laser with the frequency ω on the ion. If the state is $|0\rangle$, the photon gets absorbed and the electron jumps to E_{aux} and from there spontaneously back to E_0.This process then repeats over and over emitting many photons. This will create fluorescence, which can be measured by light detectors. If the state is $|1\rangle$, no ions get absorbed and no fluorescence can be seen.

So all in all, we measure an ion by shining a photon of frequency ω onto it and then look whether it lights up.

14 Universal set of gates

So far we have seen a wide variety of different quantum gates, depending on the quantum computer architecture. We now look further more into which gates are needed to create arbitrary circuits on a quantum computer.

14.1 Pauli gates as universal qubit gates?

As a warmup recall the Pauli matrices X, Y and Z. If we only have access to a X and Y gate, could we construct the Z gate from this as well?

We could combine Y and X gate, which gives us the gate

$$XY = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} = iZ$$

So we can get the Pauli Z gate from the X and Y gate up to a global phase factor. With simple math, one can show that it is also possible to get a X gate from Y and Z and also the Y gate from X and Z.

So with this, we might assume that we can create all gates on a quantum computer only by using two of the Pauli matrices.

Unfortunately this is not true, since we can only manipulate a single qubit using the Pauli matrices. Therefore a e.g. CNOT is not possible.

But maybe we can create all single qubit gates from the Pauli matrices? Unfortunately this is also not true, since the product of a Pauli matrix is always either another Pauli matrix or the identity matrix. So we won't get a universal set of gates from the Pauli matrix and we have to try a different set of gates.

14.2 Rotation gates

We look at a different type of gates, the rotational gates R_X,R_Y and R_Z. These gates are defined by

$$R_X(\theta) = e^{-iX\theta/2} = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$
$$R_Y(\theta) = e^{-iY\theta/2} = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ -\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$
$$R_Z(\theta) = e^{-iZ\theta/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

We can obviously not create any multiple qubit gates from this, but is possible to get all single qubit gates from this. We call this universal for single qubit gates.

Definition 14.1 (Universal set of gates)**.** A set G of gates is **universal**, iff any unitary can be approximated to an arbitrary precision.
This means that for all unitary $U \in \mathbb{C}^{n\times n}$, for all n and for all ϵ there exists a circuit C consisting of elements of G without auxillary qubits such that for all quantum states ψ it holds that $\|U_\psi - C_\psi\| \leq \epsilon$.
We call G universal for single qubit gates when this holds for $n = 1$.

Using this definition, we can formalize what is stated above:

Theorem 14.1 (Rotation gates)**.** *The set* $\{R_X, R_Y, R_Z : \theta \in [0, 2\pi]\}$ *is universal for single qubit gates.*

When adding a CNOT to the set of universal single qubit gates, we get a general set of universal gates.

Theorem 14.2 (Single qubit gates and CNOT)**.** *The set of* $\{\textit{all single qubit gates}, \text{CNOT}\}$ *is universal.*

Corollary 14.1 (Rotation gates and CNOT)**.** *The set of* $\{R_X, R_Y, R_Z, \text{CNOT} : \theta \in [0, 2\pi]\}$ *is universal for all* $\theta \in [0, 2\pi]$.

14.3 Clifford gates

We look at another set of gates called *Clifford* gates. This set consists of CNOT, H and S.

While there are many gates, which can be construct from the set of Clifford gates, the set is not universal. For example, there is a gate called T, which can not be constructed from Clifford gates. T is denoted by

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

Note that $T^2 = S$ and $S^2 = Z$ and $Z^2 = I$.

But while the Clifford gates are not universal, there is a different useful fact, which we can especially use without access to a quantum computer:

Theorem 14.3 (Gottesmann-Knill theorem)**.** *A quantum circuit only using Clifford gates can be efficiently simulated on a classical computer.*

However if we add T, we get a universal set:

Theorem 14.4 (Clifford gates and T)**.** *The set of all Clifford gates and T is universal.*

There is also one more example of a universal set:

Theorem 14.5 (Toffoli and H)**.** *The set of* $\{\text{Toffoli}, H\}$ *is universal.*

14.4 Gottesmann-Knill theorem

We take a closer look into why Theorem 14.3 works.

First we look into stabilizer states. Given a set of unitaries M, ψ is stabilized by M if for all $U \in M$ is holds that $U\psi = \psi$. The state $|0\rangle$ is for example stabilized by $\{Z\}$, since $Z|0\rangle = |0\rangle$ and it is the only state (up to global phase) stabilized by $\{Z\}$, since e.g. $Z|1\rangle = -|1\rangle$.

Let P be the set of all n-qubit tensor products of X, Y, Z, I and a factor ± 1 and $\pm i$. For $n = 4$ P contains for example $X \otimes I \otimes Z \otimes Z$ and $-iX \otimes Y \otimes I \otimes I$.

We call ψ a stabilizer state if there exists a set $M \subseteq P$ such that ψ and only ψ is stabilized by M up to a global phase.

Example: Stabilizing $|0\rangle^n$

The quantum state $|0\rangle^n = |0\rangle \otimes \cdots \otimes |0\rangle$ is stabilized by $M_0 = \{Z \otimes I \otimes \cdots \otimes I, I \otimes Z \otimes \cdots \otimes I, I \otimes I \otimes \cdots \otimes Z\}$ and therefore the quantum state $|0\rangle^n$ is a stabilizer state and $|M_0| = n$.

The big trick is now that if ψ is stabilized by M, it follows that $U\psi$ is stabilized by M' for some M'. But what is M'?

We know that if $A \in M$ it holds that $A\psi = \psi$. So it follows that $UAU^\dagger = U\psi$ and therefore $UAU^\dagger$ is a stabilizer of $U\psi$. The converse also holds. So $M' = \{UAU^\dagger : A \in M\}$.

Now we can use a useful fact: For any Clifford gate G, if M contains "Pauli tensors", then M^G also contains "Pauli tensors".

The simulation algorithm to simulate Clifford gates basically works as follows:

1. $M := M_0$, which is the stabilizer for $|0\rangle^n$.
2. For each gate G:
$$M := M^G$$
3. To calculate the measurement outcomes, do linear algebra (not further specified here).

Since M will always be a set of n tensor products of n Pauli matrices, this gives a polynomial time algorithm for simulation Clifford circuits.

15 Repetition code