

Introduction to Quantum Computing

Lecture notes for the summer term 2025

Jannik Hellenkamp

Stefan Stump

Dominique Unruh

2025-10-20

Table of contents

Welcome	2
1 Introduction	2
1.1 Double-slit experiment	2
1.2 What is a quantum computer?	4
2 Probabilistic systems	6
2.1 Deterministic possibilities	6
2.2 Probability distribution	6
2.3 Probabilistic processes	7
Applying a probabilistic process	8
3 Quantum systems	9
3.1 Classical possibilities	9
3.2 Quantum states	9
3.3 Unitary transformation	10
4 Observing probabilistic and measuring quantum systems	11
4.1 Observing a probabilistic system	12
4.2 Measuring a quantum system	13
4.3 Elitzur–Vaidman bomb tester	15
5 Partial observing and measuring systems	19
5.1 Partially observing a probabilistic system	19
5.2 Partially measuring a quantum system	21
6 Composite Systems	23
6.1 Constructing composite systems	23
6.2 Measuring composite systems	25
7 Quantum Circuits	27
7.1 Visual language	27
7.2 Important gates	28
7.2.1 Single qubit gates	28
7.2.2 Controlled-NOT gate	29
7.3 Teleportation	30
8 Ket Notation	33
8.1 Teleportation	34
8.2 Bra and Braket notation	36

9 Bernstein-Vazirani Algorithm	37
10 Discrete Fourier Transformation	40
10.1 Definition of the DFT	41
10.2 Constructing the DFT	41
11 Shor's Algorithm	45
11.1 Reducing factoring to period finding	45
11.2 The quantum algorithm for period finding	46
11.3 Post processing	47
12 Grover's algorithm	49
12.1 Preparations	50
12.1.1 Constructing the oracle V_f	50
12.1.2 Constructing FLIP_*	51
12.2 The algorithm for searching	52
12.2.1 Understanding the algorithm for searching	53
13 Quantum Physics	55
13.1 Quantum state	55
13.2 Time evolution of a quantum state	56
13.3 Energy / Hamiltonian	56
13.4 Schrödinger equation	58
14 From Quantum Physics to a Quantum Computer	59
15 Ion-based quantum computers	61
15.1 Setup	61
15.2 Operations	62
15.2.1 Cooling	62
15.2.2 Measurements	63
15.2.3 Unitaries	63
16 Universal set of gates	66
16.1 Pauli gates	66
16.2 Other known gates	67
16.3 Rotation gates	67
16.4 Clifford gates	68
16.5 Gottesmann-Knill theorem	69
17 Quantum error correction	70
17.1 Repetition code	70
17.1.1 Bit flip code	71
17.1.2 Phase flip code	72

17.2	Shor's code	73
17.3	Steane code	74
18	Fault-tolerant computation	74
18.1	Operations on logical/physical qubits	75
18.2	Fault-tolerant gates	76
18.2.1	Analysis of a circuit	77
18.2.2	Development of the error probability	80
19	Quantum complexity	81
19.1	Oracle lower bounds	82
19.1.1	Proof sketch	82
19.2	Quantum Merlin Arthur	86

Welcome

These are the lecture notes for the “Introduction to Quantum Computing” lecture held by Dominique Unruh at RWTH Aachen in the summer term 2025. They should be viewed as an addition to the [handwritten notes](#) and [the lecture recordings](#).

If you spot an error, please report them via [Gitlab](#). If you have a question of understanding, please ask it in the [Moodle forum](#).

These lecture notes are released under the [CC BY-NC 4.0 license](#).

The Jupyter notebooks created during the lectures can be found in the [JupyterHub](#) in the course “[IQC] Introduction to Quantum Computing” or in [Moodle](#). If changes are made to the files, they can be easily reset with the usual git commands using “Git” and then “Open Git Repository in Terminal”.

1 Introduction

1.1 Double-slit experiment

We start by looking at one of the most famous quantum experiments to get an idea of the surprising nature of quantum behaviour.



Figure 1.1: From [xkcd 3076](#)

In the double slit experiment, a light source is placed behind a wall with two narrow, closely spaced slits. On the other side, a photosensitive plate is positioned.

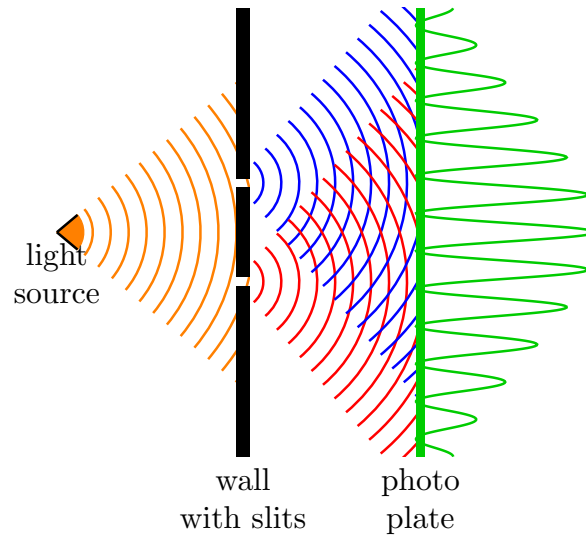


Figure 1.2: Double-slit experiment setup

An interference pattern appears on this photo plate, i.e. alternating light and dark stripes. This is due to the wave character of light. As the light waves pass through the slits, they overlap and interfere with each other. In some areas they have the same amplitude and reinforce each other, creating bright stripes. In other areas, the amplitudes have different signs and the waves cancel each other out, creating dark stripes.

This behaviour is to be expected. This is because the light travels different distances from the two slits to the same spot on the photo plate. If the difference is half a wavelength the two light waves cancel out at that spot.

Now we take individual photons. Here we would expect the interference pattern to disappear, as each photon can only pass through one of the slits. In that case, no interference can occur between photons coming from the two slits since they never meet. We would expect just two overlapping bright areas.

Surprisingly this behaviour does not occur. Even with single photons an interference pattern continues to appear. The photons do not decide to pass through one specific slit. They are in “superposition” between these two paths. This means that a single photon has two possibilities where it came from, both possibilities still happen at the same time and can cause interference with each other. For this reason, the amplitudes also add up or cancel out at the photo plate, resulting in the same interference pattern.

In later chapters, the mathematics shown will make this behaviour easier to understand.

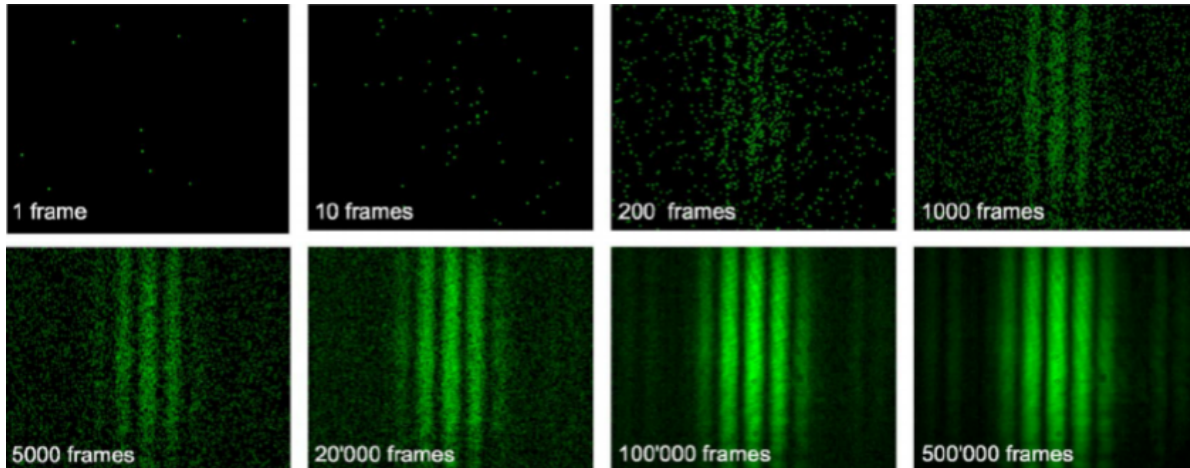


Figure 1.3: Resulting interference pattern when using single photons

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-*qubit* 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 $x = 0, x = 1, x = 2, \dots$ 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.

We will always assume the deterministic possibilities to be ordered in some way (even if it is an arbitrary one). In the example above, the deterministic possibilities are 00, 01, 10, 11, not 00, 10, 01, 11. We will need this to know the order of entries in vectors and matrices later.

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*. Here it comes in handy that we have a ordering on the deterministic possibilities. If the deterministic possibilities are $x_1, \dots, x_n$, their probabilities will be in the vector in that order.

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

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

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

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 for all i is $a_i \geq 0$, therefore a matrix A is a probabilistic process iff $A \in \mathbb{R}^{N \times N}$ with $\sum a_i = 1$ and for all i is $a_i \geq 0$. 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

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

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 $d_{2\text{-bit}}$. Using A_{flip} we can calculate the final probability distribution:

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

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 Classical possibilities

Like in the probabilistic systems we need to define all outcomes for a quantum system. For example, a photon can be in the state *up* or *down*. We call these possibilities *classical possibilities*. Note that we will only be using a *finite* number of possibilities.

Example: Random bit

Imagine you have a random bit generator, which outputs one bit. The classical possibilities of this generator are 0 and 1.

We will always assume the classical possibilities to be ordered in some way (even if it is an arbitrary one), like the deterministic possibilities. In the example above, the classical possibilities are 0, 1 not 1, 0. We will need this to know the order of entries in vectors and matrices later.

3.2 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:

$$\begin{aligned} |0\rangle : \quad & \Pr[0] = |1|^2 = 1 & \Pr[1] = |0|^2 = 0, \\ |1\rangle : \quad & \Pr[0] = |0|^2 = 0 & \Pr[1] = |1|^2 = 1, \\ |+\rangle : \quad & \Pr[0] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} & \Pr[1] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}, \\ |-\rangle : \quad & \Pr[0] = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} & \Pr[1] = |-\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}. \end{aligned}$$

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

3.3 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. The following Lemma is therefore useful.

Lemma 3.1 (Unitary matrix). *For a square matrix U , the following are equivalent:*

- U maps every quantum state to a quantum state,
- $U^\dagger U = I$ and $U U^\dagger = I$,
- $U^\dagger U = I$,
- all columns are quantum states and mutually orthogonal.

Definition 3.2 (Unitary transformation). A matrix U is called *unitary* iff $U^\dagger U = I$ and $UU^\dagger = I$.

Then the evolution of a quantum state is always described by a unitary matrix. So if the current state is ψ and we apply the transformation matrix U , the state is $U\psi$ afterwards.

A unitary matrix is by definition 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 know 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 $d_{1\text{-bit}} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$ 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.

Observing the final distribution

From Section 2.3 we know that the final distribution d is

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

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

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 $\Pr[0] = d_0 = \frac{1}{2}$. We get 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.

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 $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$. The outcome 1 can never occur, i.e. $\Pr[1] = |\psi_1|^2 = 0$

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 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 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

Next we examine an application that would not be possible with classical methods: the Elitzur–Vaidman bomb tester.

The situation is that a box is given and we want to determine whether it contains a bomb. To test this, a photon can be sent through the box. If a bomb is present, it is wired to a photon detector. If this detects a photon, the bomb explodes.

This means that if no bomb is present, the photon can pass through the box, but if a bomb is present, the photon is detected and the bomb explodes. The detector is perfect (it never misses a photon entering the box).

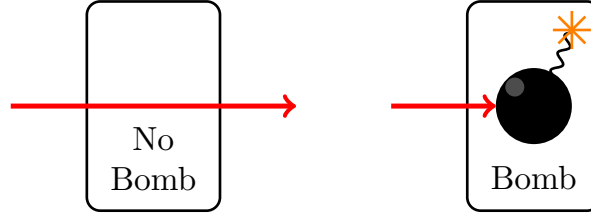


Figure 4.1: Boxes with and without bomb

Surprisingly, using quantum superposition, it is possible to detect the presence of a bomb without it exploding.

To understand how the bomb detector work, we first need to introduce a concept called a beam splitter:

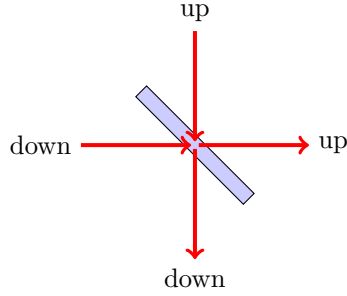


Figure 4.2: Beam splitter

A beam splitter is basically a semi-transparent mirror. In particular, when a photon comes in on the *up* path, it could come out on the *up* or *down* path. And if it comes in on the *down* path, it could come out on the *up* or *down* path as well.

Quantum mechanically, the photon could even come in in a superposition between *up* and *down*, denoted by a quantum state $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, where α is the amplitude of the *up* classical possibility, and β of the *down* classical possibility. Then it exits the beam splitter in a superposition $\begin{pmatrix} \gamma \\ \delta \end{pmatrix}$ of *up* and *down*, where:

$$\begin{pmatrix} \gamma \\ \delta \end{pmatrix} = B \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad B := \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Note that the specific matrix B depends on the precise physical setup. Here we chose one that makes the calculation more pleasant. We will later introduce B as Hadamard-gate.

Now we can describe the bomb tester:

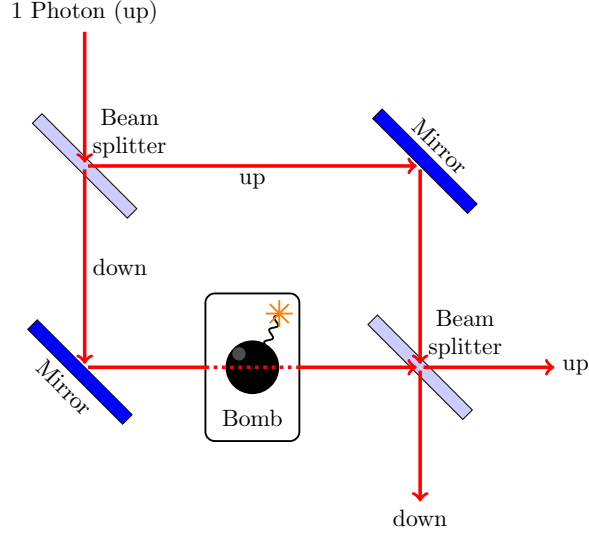


Figure 4.3: Setup for the Elitzur–Vaidman bomb tester

We begin with a photon in the *up* state, represented by $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$. The second state is called *down*, represented by $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

This photon in that *up* state is sent through a first beam splitter, which creates a superposition and sends the photon in a different direction depending on its state. Thus, after the beam splitter the photon is in the state

$$B \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Thus, the photon is in an equal superposition of the *up* state and the *down* state. The *down* path passes through the box and potentially the bomb, while the *up* state bypasses the box.

At this point, it makes a difference whether the bomb is present or not. Firstly, we consider the case where there is no bomb:

In this case the photon can pass through both paths without any measurement. It therefore reaches at a second beam splitter in any case. This is identical to the first, which means that the photon is then in the following state afterwards:

$$B \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Then we measure on which path the photon is. This results in the following probability distribution:

	Probability if no bomb
State up	1
State down	0
Bomb explodes	0

Now we consider the case where a bomb is present. In this case, by observing whether the bomb explodes, we effectively measure whether the photon took the *up/down* path. It is $\Pr[\text{down}] = \Pr[\text{up}] = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$. If the photon is in the *down* state, the photon passes the box and the bomb explodes. If the photon is in the *up* state $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ (the post-measurement state after the outcome “no explosion”), it bypasses the box and reaches the second beam splitter. After this it is in the state:

$$B \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Then we measure on which path the photon exits. This is again an equal superposition between both states, which leads to the following probability distribution:

	Probability if no bomb	Probability if bomb
State up	1	1/4
State down	0	1/4
Bomb explodes	0	1/2

Thus, if the photon is in the *down* state at the end, we know that a bomb is present without the bomb exploding. While a probability of 1/2 of exploding is not very satisfactory, we nonetheless have achieved something striking: With probability 1/4, we “observe” that there is a bomb, without the bomb ever having been touched by the photon. This is impossible classically.

Notice that the setup can be improved (details omitted) so that the probability distribution is as follows:

	Probability if no bomb	Probability if bomb
State up	1	≈ 0
State down	0	≈ 1
Bomb explodes	0	≈ 0

So it can almost certainly be found out whether a bomb is present without it exploding.

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 always 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 deterministic state in A .

Definition 5.1 (Partially observing a probabilistic system). Given a probabilistic system with deterministic possibilities $X = (x_1, \dots, x_n)$, a distribution $\mu \in \mathbb{R}^N$ and a family of alternatives $A_1, \dots, A_m$ with $A_i \cap A_j = \emptyset$ and $\bigcup_i A_i = X$, the probability of observing the alternative k is given by

$$\Pr[\text{outcome} = k] = \sum_{x_i \in A_k} \mu_{(x_i)}.$$

The distribution v after the observation of the outcome k is given by the (normalized) conditional distribution:

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} \text{ with } v_i := \begin{cases} \frac{\mu_i}{\Pr[\text{outcome}=k]} & \text{if } x_i \in A_k \\ 0 & \text{if } x_i \notin A_k \end{cases}.$$

Note: In this definition we were careful to distinguish between the names x_i of the deterministic possibilities and their number i (e.g. when writing μ_i). We will often be less precise and simply

pretend the deterministic possibilities are the number $1, \dots, N$. That is, we would write the definition as follows and pretend it means the above:

Definition 5.2 (Partially observing a probabilistic system). Given a distribution $\mu \in \mathbb{R}^N$ 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

$$\Pr[\text{outcome} = k] = \sum_{i \in A_k} \mu_i.$$

The distribution v after the observation of the outcome k is given by the conditional distribution:

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} \text{ with } v_i := \begin{cases} \frac{\mu_i}{\Pr[\text{outcome}=k]} & \text{if } i \in A_k \\ 0 & \text{if } i \notin A_k \end{cases}.$$

Note that similar to the full observation of a probabilistic system a partial observation does not actually change the system. We only get some new knowledge. In particular, a third person can never notice whether we observed the system or not.

Example: Partially observing a probabilistic system

A fair dice was rolled and it is only known that it is not a 5. Thus the distribution μ is given by

$$\mu = \begin{pmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ 0 \\ \frac{1}{5} \end{pmatrix}.$$

Now we want to observe whether the number is low (≤ 3) or high (≥ 4). This means we have two alternatives: $A_{\text{low}} = \{1, 2, 3\}$ and $A_{\text{high}} = \{4, 5, 6\}$. We therefore obtain the following probabilities for these two alternatives

$$\Pr[\text{outcome} = \text{low}] = \frac{1}{5} + \frac{1}{5} + \frac{1}{5} = \frac{3}{5},$$

$$\Pr[\text{outcome} = \text{high}] = \frac{1}{5} + 0 + \frac{1}{5} = \frac{2}{5}.$$

The conditional distribution after the outcome low is

$$\begin{pmatrix} \frac{1}{5}/\frac{3}{5} \\ \frac{1}{5}/\frac{3}{5} \\ \frac{1}{5}/\frac{3}{5} \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

That is, we know we have a uniformly random number from $\{1, 2, 3\}$, but don't know which. And after the outcome high the conditional distribution is

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{5}/\frac{2}{5} \\ 0/\frac{2}{5} \\ \frac{1}{5}/\frac{2}{5} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}.$$

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 5.3 (Partially measuring a quantum system). Given a quantum system with classical possibilities $X = (x_1, \dots, x_n)$, a quantum state $\mu \in \mathbb{C}^N$ and a family of alternatives $A_1, \dots, A_m$ with $A_i \cap A_j = \emptyset$ and $\bigcup_i A_i = X$, the probability of observing the alternative k is given by

$$\Pr[\text{outcome} = k] = \sum_{x_i \in A_k} |\psi_{(x_i)}|^2.$$

The post-measurement-state 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

$$\phi_i := \begin{cases} \psi_i & \text{if } x_i \in A_k \\ 0 & \text{if } x_i \notin A_k \end{cases}.$$

2. Computing the normalized post-measurement-state by calculating:

$$\text{post-measurement-state} := \frac{\phi^{(k)}}{\|\phi^{(k)}\|} = \frac{\phi^{(k)}}{\sqrt{\Pr[\text{outcome} = k]}}.$$

As in Definition 5.1, we were precise about the difference between the classical possibility x_i and their numbers but will not always be so precise in the future.

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

There is a slight difference between this definition and Definition 4.2, namely if you compute the post-measurement-state, you may get a different result. The two post-measurement-states can differ by a factor $c \in \mathbb{C}$ with $|c| = 1$, called a “global phase”. Such a global phase makes no observable physical difference, so this “contradiction” is not a problem.

Example: Partially measuring a quantum system

A photon is in superposition between the 4 paths left, right, top and bottom:

$$\psi = \begin{pmatrix} \frac{1}{10} \\ -\frac{3}{10} \\ \frac{9}{10}i \\ \frac{3}{10} \end{pmatrix}.$$

There are two alternatives: $A_{\text{horizontal}}$, so that the photon is in the left or right path, and A_{vertical} , so that the photon is in the top or bottom path. We therefore obtain the following probabilities for these two alternatives

$$\Pr[A_{\text{horizontal}}] = \left| \frac{1}{10} \right|^2 + \left| -\frac{3}{10} \right|^2 = \frac{1}{100} + \frac{9}{100} = \frac{1}{10},$$

$$\Pr[A_{\text{vertical}}] = \left| \frac{9}{10}i \right|^2 + \left| \frac{3}{10} \right|^2 = \frac{81}{100} + \frac{9}{100} = \frac{9}{10}.$$

The normalized post-measurement-state for the alternative $A_{\text{horizontal}}$ is

$$\begin{pmatrix} \frac{1}{10} \\ -\frac{3}{10} \\ 0 \\ 0 \end{pmatrix} / \sqrt{\frac{1}{10}} = \begin{pmatrix} \frac{1}{\sqrt{10}} \\ -\frac{3}{\sqrt{10}} \\ 0 \\ 0 \end{pmatrix}$$

For the alternative A_{vertical} the normalized post-measurement-state is

$$\begin{pmatrix} 0 \\ 0 \\ \frac{9}{10}i \\ \frac{3}{10} \end{pmatrix} / \sqrt{\frac{9}{10}} = \begin{pmatrix} 0 \\ 0 \\ \frac{3}{\sqrt{10}}i \\ \frac{1}{\sqrt{10}} \end{pmatrix}.$$

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, ..., 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 $x_1, \dots, x_N$ and a distribution/state μ_A and with the possibilities of B given by $y_1, \dots, y_M$ and a distribution/state μ_B , the *composite* system called AB has the possibilities

$$x_1y_1, x_1y_2, \dots, x_1y_M, x_2y_1, x_2y_2, \dots, x_2y_M, \dots, x_Ny_1, x_Ny_2, \dots, x_Ny_M$$

and the distribution/state μ_{AB} of AB is given by the *tensor product*

$$\mu_{AB} := \mu_A \otimes \mu_B = \begin{pmatrix} (\mu_A)_1 \cdot \mu_b \\ \vdots \\ (\mu_A)_N \cdot \mu_b \end{pmatrix}.$$

This vector has the size NM . Here $(\mu_A)_i$ stands for the i -th entry of μ_A .

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

Notice that the entry corresponding to the possibility x_iy_j in the composite system is then $(\mu_{AB})_{x_iy_j} = (\mu_A)_{x_i}(\mu_B)_{y_j}$. Here we identify the classical possibility x_i and y_j with the indices $1, \dots, N$ and $1, \dots, M$, respectively the classical possibility x_iy_j with the indices $1, \dots, NM$. (So $(\mu_{AB})_{x_iy_j}$ has just one index, namely $x_iy_j \in \{1, \dots, NM\}$.)

Example: Composite system

Let the distributions for the system A with the possibilities $1, 2$ and the system B with the possibilities a, b, c be given by

$$\mu_A = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad \mu_B = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{\sqrt{3}}{2} \end{pmatrix}.$$

Then the composite system AB has the possibilities $1a, 1b, 1c, 2a, 2b, 2c$ and the distribution

$$\mu_{AB} = \mu_A \otimes \mu_B = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{\sqrt{3}}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \cdot \frac{1}{2} \\ \frac{1}{\sqrt{2}} \cdot 0 \\ \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{3}}{2} \\ -\frac{1}{\sqrt{2}} \cdot \frac{1}{2} \\ -\frac{1}{\sqrt{2}} \cdot 0 \\ -\frac{1}{\sqrt{2}} \cdot \frac{\sqrt{3}}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2\sqrt{2}} \\ 0 \\ \frac{\sqrt{3}}{2\sqrt{2}} \\ -\frac{1}{2\sqrt{2}} \\ 0 \\ -\frac{\sqrt{3}}{2\sqrt{2}} \end{pmatrix}.$$

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 cannot write d_{AB} as two separate distributions, 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*.

Lemma 6.1. *For the (unitary) matrices A, B, C and D , the vectors (quantum states) ψ, ϕ and χ and the constant c , the following applies*

- $(A \otimes B)(\psi \otimes \phi) = A\psi \otimes B\phi$,
- $(A \otimes B)(C \otimes D) = AC \otimes BD$,
- $\psi \otimes (\phi + \chi) = (\psi \otimes \phi) + (\psi \otimes \chi)$,
- $(\psi + \phi) \otimes \chi = (\psi \otimes \chi) + (\phi \otimes \chi)$,
- $A \otimes (B + C) = (A \otimes B) + (A \otimes C)$,
- $(A + B) \otimes C = (A \otimes C) + (B \otimes C)$,
- $c\phi \otimes \psi = c(\phi \otimes \psi)$,
- $\phi \otimes c\psi = c(\phi \otimes \psi)$,
- $A \otimes cB = c(A \otimes B)$ and
- $cA \otimes B = c(A \otimes B)$.

These rules only apply if the dimensions of the matrices and vectors match.

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 $A_1, \dots, A_N \subseteq \{x_1, \dots, x_n\}$ and $B_1, \dots, B_M \subseteq \{y_1, \dots, y_M\}$. 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 $C_1, \dots, C_N := A \otimes \{y_1, \dots, y_M\}$.

Example: Composite measurement (quantum)

Let A be a system with the states $1, 2, 3$ and μ_A a measurement with the two alternatives $A_{\text{low}} = \{1, 2\}$, $A_{\text{high}} = \{3\}$. The quantum state is

$$\psi_A = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ -\frac{2}{3} \end{pmatrix}.$$

Another system B has the states a, b, c . The measurement μ_B the two alternatives $B_{\text{vocal}} = \{a\}$, $B_{\text{consonant}} = \{b, c\}$. The quantum state is

$$\psi_B = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2}i \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

So the composite system $C = AB$ has the classical possibilities $1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b$ and $3c$. The measurement $\mu_C := \mu_A \otimes \mu_B$ has the alternatives

$$\begin{aligned} C_{\text{low, vocal}} &= \{1a, 2a\}, & C_{\text{low, consonant}} &= \{1b, 1c, 2b, 2c\}, \\ C_{\text{high, vocal}} &= \{3a\}, & C_{\text{high, consonant}} &= \{3b, 3c\}. \end{aligned}$$

The quantum state is

$$\psi_C = \psi_A \otimes \psi_B = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ -\frac{2}{3} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2}i \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{2}{6} \\ \frac{2}{6}i \\ \frac{2}{3\sqrt{2}} \\ \frac{1}{6} \\ \frac{1}{6}i \\ \frac{1}{3\sqrt{2}} \\ -\frac{2}{6} \\ -\frac{2}{6}i \\ -\frac{2}{3\sqrt{2}} \end{pmatrix}.$$

We get for the alternative $C_{\text{low, vocal}}$ the probability

$$\left|\frac{2}{6}\right|^2 + \left|\frac{1}{6}\right|^2 = \frac{4}{36} + \frac{1}{36} = \frac{5}{36}$$

and thus the post-measurement-state is

$$\begin{pmatrix} \frac{2}{6} \\ 0 \\ 0 \\ \frac{1}{6} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} / \sqrt{\frac{5}{36}} = \begin{pmatrix} \frac{2}{6} \\ 0 \\ 0 \\ \frac{1}{6} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} / \frac{\sqrt{5}}{6} = \begin{pmatrix} \frac{2}{\sqrt{5}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{5}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

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 systems.

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 (i.e., as formulas). 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:

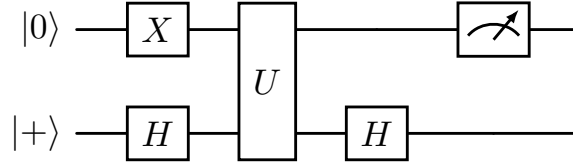


Figure 7.1: A basic quantum circuit

In this circuit we have two qubits $|0\rangle$ and $|+\rangle$, which are drawn as separate wires. Note that the symbol $|\rangle$ is introduced in the next chapter, so just understand it as a name for some state at this point. We first apply the unitary X on the top wire to $|0\rangle$ and at the same time we apply the unitary H at the bottom wire to $|+\rangle$. Mathematically this can be written as $(X \otimes H)(|0\rangle \otimes |+\rangle)$. 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. The meaning of the unitaries used is explained in the next section. A wire can contain multiple qubits, depending on the context.

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 (Identity matrix). The identity matrix I is defined as

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

This matrix is for example useful if a qubit/wire is to remain unchanged. The identity matrix also exists in other sizes.

Definition 7.2 (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 7.3 (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 7.4 (Controlled-NOT gate). The controlled-NOT gate $\text{CNOT} \in \mathbb{C}^{4 \times 4}$ 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:

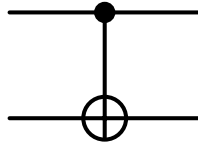


Figure 7.2: Controlled-NOT in a quantum circuit

If the second qubit should be the controlling wire and the first qubit the target wire, we

can use CNOT' denoted as

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

Accordingly, the quantum circuit is drawn the other way round.

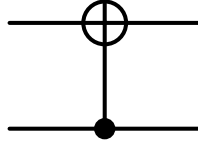


Figure 7.3: Controlled-NOT' in a quantum circuit

7.3 Teleportation

We are now looking at an example quantum circuit.

Example: Teleportation

Assumed: Alice has a qubit ψ and wants to send it to Bob. But only classic communication is possible, no quantum communication. However, they can share a state β_{00} beforehand.

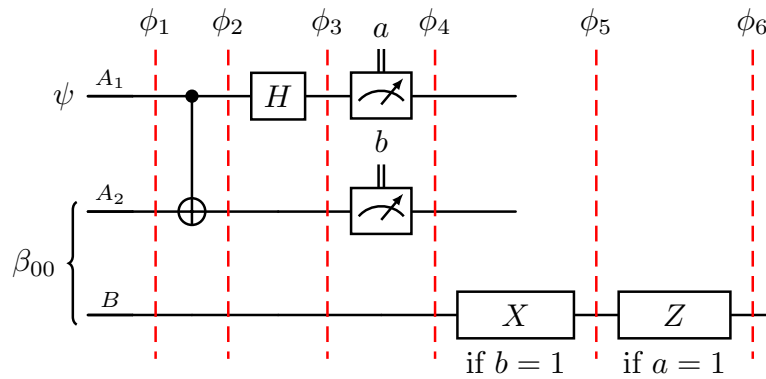


Figure 7.4: Circuit for qubit teleportation

1. Alice has the state $\psi = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ and the shared state is $\beta_{00} = \begin{pmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{pmatrix}$. This means that the entire state is

$$\phi_1 = \psi \otimes \beta_{00} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} / \sqrt{2} = \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \alpha \\ \beta \\ 0 \\ 0 \\ \beta \end{pmatrix} / \sqrt{2}.$$

2. The CNOT can be extended to $\text{CNOT} \otimes \text{I}_2$ using the identity matrix:

$$\phi_2 = (\text{CNOT} \otimes \text{I}_2) \phi_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \alpha \\ \beta \\ 0 \\ 0 \\ \beta \end{pmatrix} / \sqrt{2} = \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \alpha \\ 0 \\ \beta \\ \beta \\ 0 \end{pmatrix} / \sqrt{2}.$$

3. Identical to step 2, the Hadamard-gate can be extended with the identity matrix:

$$\phi_3 = (\text{H} \otimes \text{I}_4) \phi_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \alpha \\ 0 \\ \beta \\ \beta \\ 0 \end{pmatrix} / \sqrt{2} = \begin{pmatrix} \alpha \\ \beta \\ \beta \\ \alpha \\ \alpha \\ -\beta \\ -\beta \\ \alpha \end{pmatrix} / 2.$$

4. Here we assume that $a = 0$ and $b = 1$. It applies $|\alpha|^2 + |\beta|^2 = 1$ because ψ is a quantum state. Therefore the probability for this is

$$\left| \frac{\beta}{2} \right|^2 + \left| \frac{\alpha}{2} \right|^2 = \frac{|\alpha|^2 + |\beta|^2}{4} = \frac{1}{4}$$

and the post-measurement-state

$$\phi_4 = \begin{pmatrix} 0 \\ 0 \\ \beta \\ \alpha \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} / 2/\sqrt{\frac{1}{4}} = \begin{pmatrix} 0 \\ 0 \\ \beta \\ \alpha \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

5. Since $b = 1$, the Pauli-matrix X is used:

$$\phi_5 = (I_4 \otimes X)\phi_4 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \beta \\ \alpha \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \alpha \\ \beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

6. Since $a = 0$, nothing happens in this step:

$$\phi_6 = \phi_5 = \begin{pmatrix} 0 \\ 0 \\ \alpha \\ \beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \psi.$$

So now Bob is aware of ψ and Alice has the now useless state $(0 \ 1 \ 0 \ 0)^T$. Note that Bob would also have ψ for all other results of a and b .

8 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

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

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 example, e_{123} can represent the classical possibility “red,4,top”. 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

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

In the example, we would therefore write $|\text{red},4,\text{top}\rangle$ for e_{123} .

Example: Ket notation

Given a quantum system with the classical possibilities 00, 01, 10 and 11, the quantum state $\psi = \left(\frac{1}{\sqrt{2}} \quad 0 \quad 0 \quad \frac{1}{\sqrt{2}} \right)^T$ 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. Writing $\psi = \frac{1}{\sqrt{2}}e_1 + \frac{1}{\sqrt{2}}e_4$ would be less obvious.

Note that the ket notation can also be used in a few other ways. We can use it as described above to the state $|x\rangle$ corresponding to the classical possibility x , 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.

8.1 Teleportation

We take another look at the example from the last chapter with ket notation.

Example: Teleportation

Once again, Alice has the qubit ψ and Alice and Bob have shared the state β_{00} .

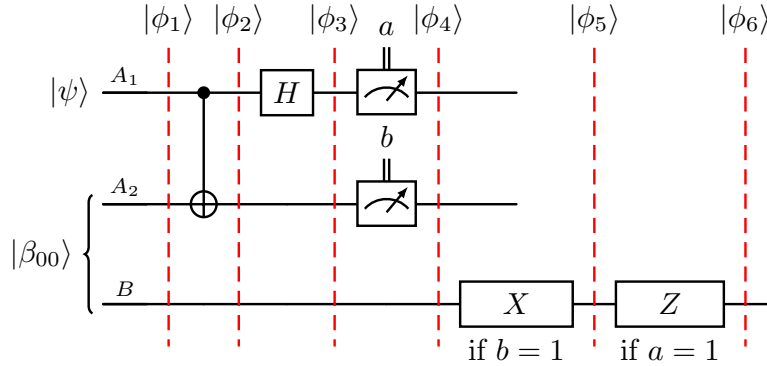


Figure 8.1: Circuit for qubit teleportation

1. Alice has the state $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle$ and the shared state is $|\beta_{00}\rangle =$

$\left(\frac{1}{\sqrt{2}} \quad 0 \quad 0 \quad \frac{1}{\sqrt{2}}\right)^T = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$. This means that the entire state is

$$\begin{aligned} |\phi_1\rangle &= |\psi\rangle \otimes |\beta_{00}\rangle \\ &= (\alpha |0\rangle + \beta |1\rangle) \otimes \left(\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle\right) \\ &= \frac{\alpha}{\sqrt{2}} |000\rangle + \frac{\alpha}{\sqrt{2}} |011\rangle + \frac{\beta}{\sqrt{2}} |100\rangle + \frac{\beta}{\sqrt{2}} |111\rangle. \end{aligned}$$

2. We can now translate each ket notation individually and get the result much simpler:

$$\begin{aligned} |\phi_2\rangle &= (\text{CNOT} \otimes \text{I}_2) |\phi_1\rangle \\ &= (\text{CNOT} \otimes \text{I}_2) \left(\frac{\alpha}{\sqrt{2}} |000\rangle + \frac{\alpha}{\sqrt{2}} |011\rangle + \frac{\beta}{\sqrt{2}} |100\rangle + \frac{\beta}{\sqrt{2}} |111\rangle \right) \\ &= \frac{\alpha}{\sqrt{2}} |000\rangle + \frac{\alpha}{\sqrt{2}} |011\rangle + \frac{\beta}{\sqrt{2}} |110\rangle + \frac{\beta}{\sqrt{2}} |101\rangle. \end{aligned}$$

3. Identical to step 2, we can look at each ket notation individually:

$$\begin{aligned} |\phi_3\rangle &= (\text{H} \otimes \text{I}_4) |\phi_2\rangle \\ &= (\text{H} \otimes \text{I}_4) \left(\frac{\alpha}{\sqrt{2}} |000\rangle + \frac{\alpha}{\sqrt{2}} |011\rangle + \frac{\beta}{\sqrt{2}} |101\rangle + \frac{\beta}{\sqrt{2}} |110\rangle \right) \\ &= \frac{\alpha}{\sqrt{2}} (\text{H} |0\rangle \otimes |00\rangle) + \frac{\alpha}{\sqrt{2}} (\text{H} |0\rangle \otimes |11\rangle) + \frac{\beta}{\sqrt{2}} (\text{H} |1\rangle \otimes |01\rangle) + \frac{\beta}{\sqrt{2}} (\text{H} |1\rangle \otimes |10\rangle) \\ &= \frac{\alpha}{\sqrt{2}} \left(\left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right) \otimes |00\rangle \right) + \frac{\alpha}{\sqrt{2}} \left(\left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right) \otimes |11\rangle \right) + \\ &\quad \frac{\beta}{\sqrt{2}} \left(\left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \right) \otimes |01\rangle \right) + \frac{\beta}{\sqrt{2}} \left(\left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \right) \otimes |10\rangle \right) \\ &= \frac{\alpha}{2} |000\rangle + \frac{\alpha}{2} |100\rangle + \frac{\alpha}{2} |011\rangle + \frac{\alpha}{2} |111\rangle + \frac{\beta}{2} |001\rangle - \frac{\beta}{2} |101\rangle + \frac{\beta}{2} |010\rangle - \frac{\beta}{2} |110\rangle. \end{aligned}$$

4. We again assume that $a = 0$ and $b = 1$ and therefore only $\frac{\alpha}{2} |011\rangle$ and $\frac{\beta}{2} |010\rangle$ are relevant. It applies $|\alpha|^2 + |\beta|^2 = 1$ because ψ is a quantum state. Therefore the probability for this is

$$\left| \frac{\beta}{2} \right|^2 + \left| \frac{\alpha}{2} \right|^2 = \frac{|\alpha|^2 + |\beta|^2}{4} = \frac{1}{4}$$

and the post-measurement-state

$$|\phi_4\rangle = \frac{\frac{\alpha}{2} |011\rangle}{\sqrt{\frac{1}{4}}} + \frac{\frac{\beta}{2} |010\rangle}{\sqrt{\frac{1}{4}}} = \alpha |011\rangle + \beta |010\rangle = |01\rangle \otimes (\alpha |1\rangle + \beta |0\rangle).$$

5. Since $b = 1$, the Pauli-matrix X is used:

$$\begin{aligned}
 |\phi_5\rangle &= (I_4 \otimes X) |\phi_4\rangle \\
 &= (I_4 \otimes X) (|01\rangle \otimes (\alpha |1\rangle + \beta |0\rangle)) \\
 &= I_4 |01\rangle \otimes (X(\alpha |1\rangle + \beta |0\rangle)) \\
 &= |01\rangle \otimes (\alpha X |1\rangle + \beta X |0\rangle) \\
 &= |01\rangle \otimes (\alpha |0\rangle + \beta |1\rangle).
 \end{aligned}$$

6. Since $a = 0$, nothing happens in this step:

$$|\phi_6\rangle = |\phi_5\rangle = |01\rangle \otimes (\alpha |0\rangle + \beta |1\rangle) = |01\rangle \otimes |\psi\rangle.$$

As expected, we get the same result as in the previous chapter.

The ket notation can save a lot of work and sources of error. (Keep in mind that the example here got a bit lengthy because we wrote out a lot of intermediate steps.)

8.2 Bra and Braket notation

In addition to the ket notation, we also use the corresponding bra notation, written as $\langle\psi|$, which denotes the conjugate transpose (Hermitian) of the state $|\psi\rangle$:

$$\langle\psi| := |\psi\rangle^\dagger.$$

As an example, consider $\psi = \begin{pmatrix} 0 \\ i \end{pmatrix}$:

$$\langle\psi| = (0 \quad -i) = |\psi\rangle^\dagger.$$

Combining the bra and the ket notation gives us the braket notation, written as $\langle\psi|\phi\rangle$, which is the inner product between the quantum state ψ and ϕ :

$$\langle\psi|\phi\rangle := \langle\psi| |\phi\rangle.$$

As an example, let $\psi = \begin{pmatrix} 0 \\ i \end{pmatrix}$ and $\phi = \begin{pmatrix} \frac{1}{\sqrt{2}}i \\ -\frac{1}{\sqrt{2}}i \end{pmatrix}$:

$$\langle\psi|\phi\rangle = \langle\psi| |\phi\rangle = (0 \quad -i) \begin{pmatrix} \frac{1}{\sqrt{2}}i \\ -\frac{1}{\sqrt{2}}i \end{pmatrix} = 0 \cdot \frac{1}{\sqrt{2}}i + (-i) \cdot \left(-\frac{1}{\sqrt{2}}i\right) = -\frac{1}{\sqrt{2}}.$$

Notice that $\langle\psi|\phi\rangle$ or $\langle\psi, \phi\rangle$ is a common notation for the inner product. In fact, the symbols $\langle\cdot|$ and $|\cdot\rangle$ were chosen exactly to make this work.

9 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 $f : \{0,1\}^n \rightarrow \{0,1\}$, 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 $x \cdot y = x_1y_1 + \dots + x_ny_n \bmod 2$.

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 $m \leq n$ 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:

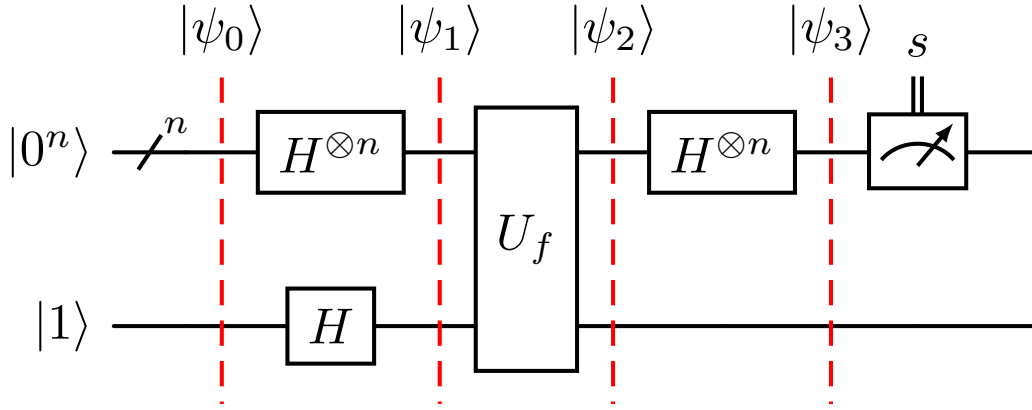


Figure 9.1: The quantum circuit for Bernstein-Vazirani

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 $|0\rangle^n = |0\rangle \otimes \dots \otimes |0\rangle$. The bottom wire is in the state $|1\rangle$. Both wires composed together can be written as $|\psi_0\rangle = |0^n 1\rangle = |0\rangle^n \otimes |1\rangle$, which is the overall starting state of our algorithm. 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:

$$\begin{aligned}
|\psi_1\rangle &= (H^{\otimes n} \otimes H) (|\psi_0\rangle) \\
&= (H^{\otimes n} \otimes H) (|0\rangle^n \otimes |1\rangle) \\
&= (H^{\otimes n} |0\rangle^n) \otimes H |1\rangle \\
&= |+\rangle^{\otimes n} \otimes |-\rangle \\
&= \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right)^{\otimes n} \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |-\rangle
\end{aligned}$$

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

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 can be calculated as follows:

$$\begin{aligned}
|\psi_2\rangle &= U_f |\psi_1\rangle \\
&= U_f \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |-\rangle \\
&= U_f \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} U_f (|x\rangle \otimes |-\rangle) \\
&\stackrel{(*)}{=} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \otimes |-\rangle \\
&= \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \right) \otimes |-\rangle
\end{aligned}$$

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

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

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 $|\psi_3\rangle = |s\rangle \otimes |-\rangle$. 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 $|\psi_2\rangle$. We do

it in this direction, since these calculations are a bit simpler:

$$\begin{aligned}
\left((H^{\otimes n})^\dagger \otimes I\right) |\psi_3\rangle &= (H^{\otimes n})^\dagger |s\rangle \otimes |-\rangle \\
&= H^{\otimes n} |s\rangle \otimes |-\rangle \\
&= H^{\otimes n} (|s_1\rangle \otimes \cdots \otimes |s_n\rangle) \otimes |-\rangle \\
&= H |s_1\rangle \otimes \cdots \otimes H |s_n\rangle \otimes |-\rangle \\
&= \bigotimes_{i=1}^n \left(\frac{1}{\sqrt{2}} |0\rangle + (-1)^{s_i} \frac{1}{\sqrt{2}} |1\rangle \right) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{i=1}^n \left(|0\rangle + (-1)^{s_i} |1\rangle \right) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \left((-1)^{x_1 s_1} (-1)^{x_2 s_2} \cdots (-1)^{x_n s_n} |x\rangle \right) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \left((-1)^{\sum_i x_i s_i \bmod 2} |x\rangle \right) \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x} |x\rangle \otimes |-\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \otimes |-\rangle \\
&= |\psi_2\rangle
\end{aligned}$$

This calculation shows that we have the quantum state $|s\rangle \otimes |-\rangle$ before the measurement.

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

This concludes the Bernstein-Vazirani algorithm.

10 Discrete Fourier Transformation

In Chapter 11, we will discuss Shor's algorithm, which requires 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.

10.1 Definition of the DFT

The DFT is defined as follows:

Definition 10.1 (Discrete Fourier Transformation). The discrete Fourier transform (DFT) is a linear transformation on $\mathbb{C}^N$ represented by the matrix

$$\text{DFT}_N := \frac{1}{\sqrt{N}}(\omega^{kl})_{kl=0,\dots,N-1} \in \mathbb{C}^{N \times N}$$

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

It applies $\omega^N = 1$ and $\omega^i \neq 1$ for all $0 < i < N$.

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

Theorem 10.1 (Properties of the DFT). *Here are some properties of the DFT which can be used without further proof.*

1. The DFT_N is unitary.
2. $\omega^t = \omega^{t \bmod N}$ for all $t \in \mathbb{Z}$.
3. Let $t \mid N$ and s be integers and the quantum state $\psi \in \mathbb{C}^N$ be given by

$$\psi_i := \begin{cases} \sqrt{\frac{t}{N}} & \text{if } i = at + s \text{ for some } a \\ 0 & \text{else.} \end{cases}$$

In other words ψ is t -periodic. Then for $\phi := \text{DFT}_N$ we have

$$|\phi_i| = \begin{cases} \frac{1}{\sqrt{t}} & \text{if } \frac{N}{t} \mid i \\ 0 & \text{else.} \end{cases}$$

The first peak of ϕ (besides ϕ_0) is at $\frac{N}{t}$.

10.2 Constructing the DFT

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

To start the circuit, we recall the definition of the DFT_N from Definition 10.1: $\text{DFT}_N := \frac{1}{\sqrt{2^n}}(\omega^{kl})_{kl}$ with $\omega := e^{2\pi i/2^n}$. To apply the DFT_N to a quantum state $|j\rangle$ we calculate

$$\begin{aligned}
\text{DFT}_N |j\rangle &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} |k\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{k_1 \in \{0,1\}} \dots \sum_{k_n \in \{0,1\}} e^{2\pi i j (\sum_{l=1}^n k_l 2^{-l})} |k_1 \dots k_n\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{k_1 \in \{0,1\}} \dots \sum_{k_n \in \{0,1\}} \bigotimes_{l=1}^n e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
&= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^n \sum_{k_l \in \{0,1\}} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
&= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^n (|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle) \\
&= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^n (|0\rangle + e^{2\pi i 0.j_{n-l+1} \dots j_n} |1\rangle).
\end{aligned}$$

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 $\text{DFT}_N |j\rangle$ as the following tensor product of quantum states

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

From this rewritten tensor product, we can get an idea on how to construct the quantum circuit for the DFT_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:

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

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 $\text{DFT}_{2^3} |j\rangle$ as the following tensor product:

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

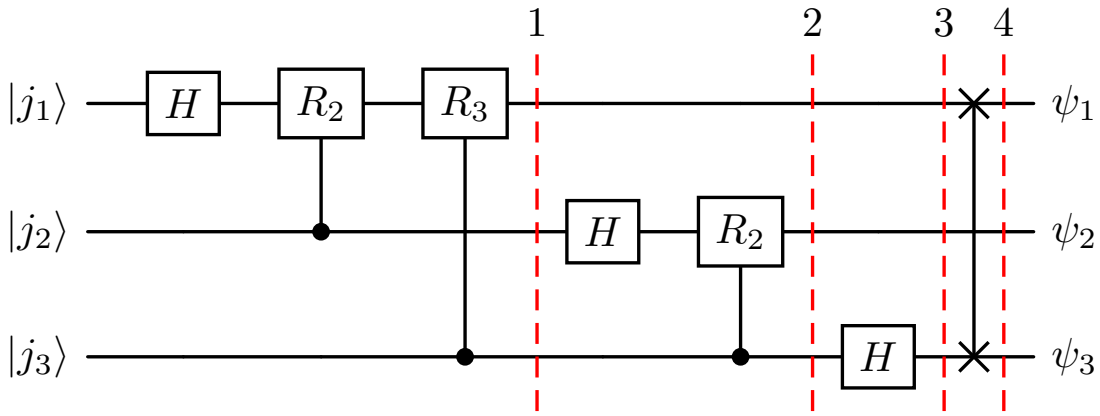


Figure 10.1: The DTF for three qubits

1. 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 Hadamard-gate to bring $|j_1\rangle$ into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{j_1} |1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_1} |1\rangle)$. 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 $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2} |1\rangle)$ and after applying R_3 we have the state $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 j_3} |1\rangle)$ on the top wire. This is the same as ψ_3 , so we are done on the first wire (We are at the first slice in the figure).
2. The next step is to construct the ψ_2 state on the middle wire. We again use a Hadamard-gate to bring $|j_2\rangle$ into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_2} |1\rangle)$. 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 $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_2 j_3} |1\rangle)$, which

is ψ_2 . (We are at the second slice in the figure).

3. On the bottom wire, we can just do a Hadamard-gate to bring $|j_3\rangle$ into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_3} |1\rangle)$. We then have ψ_1 on the bottom wire. (We are at the third slice in the figure).
4. When applying this circuit, we get the state $\psi_3 \otimes \psi_2 \otimes \psi_1$ as a result. This is 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 and delivers the correct output for DFT_{2^3} .

The more general approach to construct the DFT_N as a quantum circuit with n qubits ($N = 2^n$) 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 Hadamard-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_i . 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_N with the SWAP is shown in this figure.

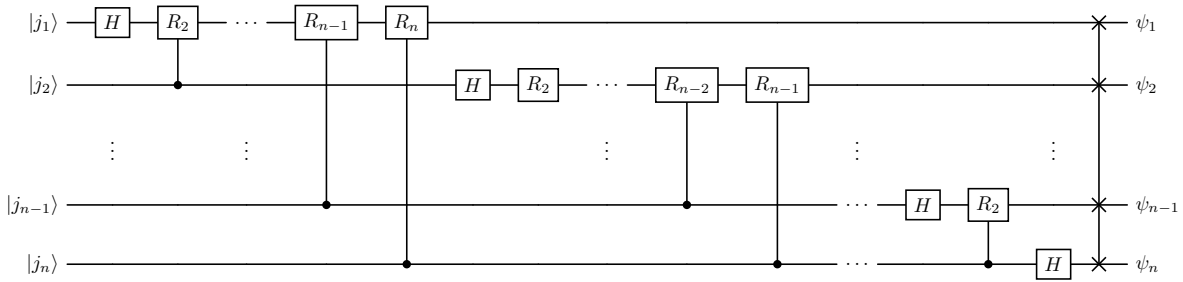


Figure 10.2: The DTF for n qubits

11 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.

11.1 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. Thus we hope that quantum circuits will be particularly good at problems related to periods. We now look into using period finding to factor integers. We first look at the definition of some problems:

Definition 11.1 (Factoring problem). Given integer N with two prime factors $p, q > 2$ 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 two distinct prime factors.

Definition 11.2 (Period finding problem). Given R , and given $f : \mathbb{Z} \rightarrow X$ with $f(x) = f(y)$ iff $x \equiv y \pmod{r}$ for some fixed secret $r \leq R$, find r .
We call r the *period* of f .

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

Definition 11.3 (Order finding problem). For known a and N which are relatively prime, find the period r of $f(i) = a^i \pmod{N}$. (This is similar to finding the smallest $i > 0$ with $f(i) = a^i \pmod{N} = 1$).
We call r the order of a written $r = \text{ord}_N a$.

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 an integer N as an input for the factoring problem.

1. Pick an $a \in \{2, \dots, N-1\}$ relatively prime to N .
2. Compute the order of a , so that $r := \text{ord}_N a$ (Using an algorithm for the order finding problem with $R := N$ because $\text{ord}_N a \leq N$ always).

3. If the order r is odd, restart at 1.
4. Calculate $x := a^{\frac{r}{2}} + 1 \pmod{N}$ and $y := a^{\frac{r}{2}} - 1 \pmod{N}$.
5. If $\gcd(x, N) \in \{1, N\}$, we restart at 1.
6. Return $p = \gcd(x, N)$ and $q = \frac{N}{\gcd(x, N)}$.

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

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

This means that $N \mid xy$ and therefore $p \mid xy$ and $q \mid xy$. From this it can be concluded that $p \mid x$, $q \mid y$, $p \nmid y$ and $q \nmid x$ which leads to $\gcd(x, N) = p$. Or p and q swapped.

Theorem 11.1 (Probability of an abort). *If N has at least two different prime factors and N is odd, then the probability to restart 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). Therefore, though factoring is the goal of this section, we focus on order finding now.

11.2 The quantum algorithm for period finding

We now look into an quantum algorithm that solves the period finding problem within reasonable runtime.

For the quantum circuit we need an $f : \mathbb{Z} \rightarrow X$ which is r -periodic for some r .

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

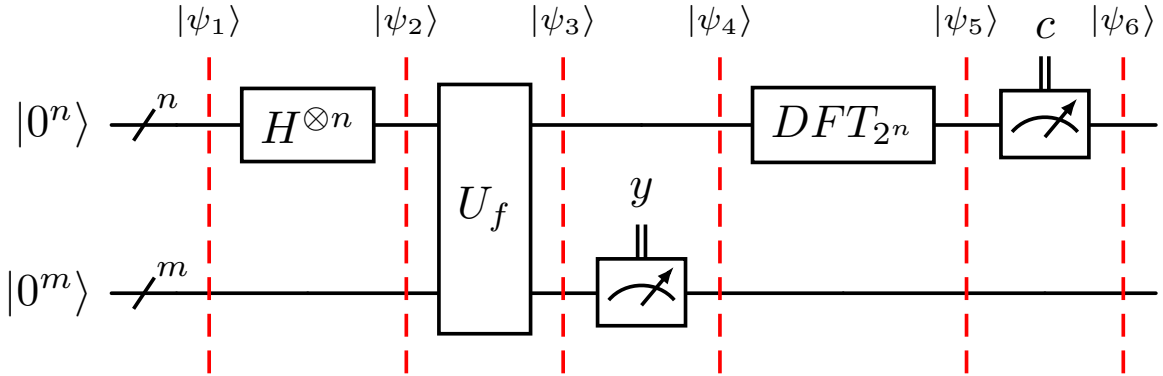


Figure 11.1: Shor's algorithm (quantum part)

Note that we evaluate f only on n bits, i.e. $\{0, \dots, 2^n - 1\} \subseteq \mathbb{Z}$. And we assume $X \subseteq \{0, 1\}^m$ for the output.

The algorithm works as follows:

1. We start with $|\psi_1\rangle = |0^n\rangle \otimes |0^m\rangle$.
2. We bring the top wire into the superposition over all entries. The quantum state is then $|\psi_2\rangle = G \cdot \sum_{x \in \{0, \dots, 2^n - 1\}} |x\rangle \otimes |0^m\rangle$ with the constant $G := 2^{-\frac{n}{2}}$.
3. We apply U_f , which is the unitary of $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$. This calculates the superposition over all possible values $f(x)$ on the bottom wire. The resulting quantum state is $|\psi_3\rangle = G \cdot \sum_{x \in \{0, \dots, 2^n - 1\}} |x, f(x)\rangle$.
4. To understand the algorithm better, we measure the bottom wire at this point. This will give us one random value $y = f(x_0)$ 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 $x \equiv x_0 \pmod r$. This means, that the resulting quantum state on the top wire is periodic. So the complete quantum state is $|\psi_4\rangle = C \cdot \sum_{x \equiv x_0 \pmod r} |x\rangle \otimes |f(x_0)\rangle$ with the constant $C = \frac{\sqrt{r}}{\sqrt{2^n}}$. (Assuming $r | 2^n$)
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 non-zero entries at multiples of $\frac{2^n}{r}$ as seen in Theorem 10.1. (Assuming $r | 2^n$)
6. We measure the top wire and get one random multiple of $\frac{2^n}{r}$, which we can denote as $c = a \cdot \frac{2^n}{r}$ for some a .

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 $r | 2^n$. Since this does usually not hold, we only get approximate multiples of $\frac{2^n}{r}$ (which is not even an integer) and thus we need a more complex post processing which we discuss in the following section.

11.3 Post processing

So far we have seen a quantum algorithm for finding an approximate multiple of the period r of a function. We now need a final step to find r . For this, we need to add one condition to the circuit. We require $n \geq 2 \log R$.

We start with the following theorem:

Theorem 11.2. *If $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ is r -periodic, the following holds with probability $\Omega(\frac{1}{\log \log r})$:*

$$\frac{-r}{2} \leq rc \pmod{2^n} \leq \frac{r}{2} \quad (*)$$

where c is the output of the second measurement of the quantum circuit described in

Section 11.2 and n is the number of qubits on the upper wire of the quantum circuit.

Note that $\frac{1}{\log \log r}$ decreases very slowly. For most practical input sizes, the probability will be $\approx \frac{1}{3}$.

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) and R is the upper bound on r :

Then there exists an integer d such that:

$$|rc - d2^n| \leq \frac{r}{2} \quad // \text{ from } (*)$$

$$\text{Hence: } \left| \frac{c}{2^n} - \frac{d}{r} \right| \leq \frac{1}{2^{n+1}} \quad // \text{ divide by } r \cdot 2^n$$

The fraction $\varphi := \frac{c}{2^n}$ is known, but the fraction $\frac{d}{r}$ is unknown. All we know is that it is a fraction and that its denominator is $\leq R$. Thus the goal is to find such a fraction $\frac{1}{2^{n+1}}$ -close to φ .

For this we use another theorem:

Theorem 11.3. *If $|\varphi - \frac{d}{r}| \leq \frac{1}{2L}$ and $r^2 \leq L$ then $\frac{d}{r}$ 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 positive 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 $a_i \geq 0$ always has to be the biggest possible integer and we stop if ... becomes 0. We call $[a_0; a_1, a_2, a_3, \dots]$ the continued expansion of t . The expansion is finite iff t is rational. For a given continued expansion, a prefix $[a_0; a_1, \dots, a_i]$ is called a convergent. So Theorem 11.3 means that 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 11.3 (with $\varphi := \frac{c}{2^n}$ and $L := 2^n$) 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. If $r \leq 2^n$ and this $\frac{d}{r}$ is $\frac{1}{2^{n+1}}$ -close to $\frac{c}{2^n}$: Return r .
3. Continue from 1 with next convergent.

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 11.2.

This completes the postprocessing of Shor's algorithm.

12 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 $f : \{0, 1\}^n \rightarrow \{0, 1\}$, 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. Otherwise it will output a 0.

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 $\sqrt{2^n}$ 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.

12.1 Preparations

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

The uniform superposition $|*\rangle$ simply denotes the superposition over all classical possibilities $|*\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$.

12.1.1 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:

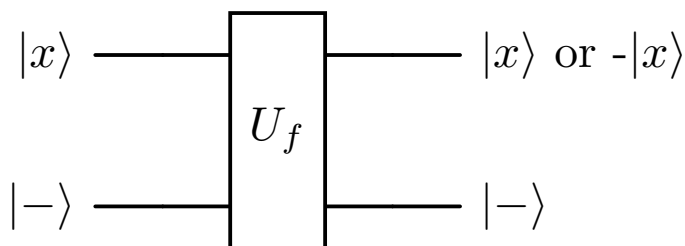


Figure 12.1: The circuit for V_f

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

12.1.2 Constructing FLIP_*

As a second ingredient for Grover's algorithm, we need a unitary FLIP_* (defined below). To realize this, we first need FLIP_0 , which is defined by the unitary

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

FLIP_0 is implemented by the following quantum circuit:

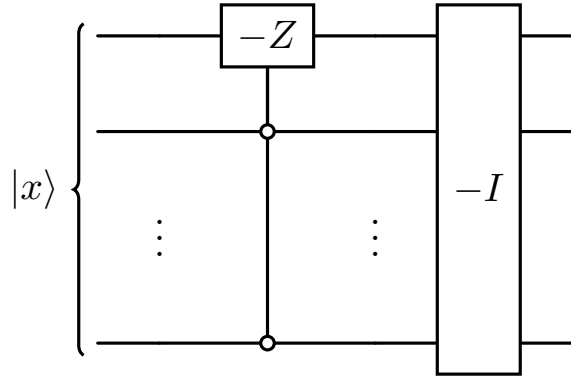


Figure 12.2: The circuit for FLIP_0

Z is the Pauli matrix from definition Definition 7.2. The empty circles indicates a negative control wire. So Z is only applied if the other wires are $|0\rangle$.

Now we can define the unitary called FLIP_* . This unitary does nothing, if it is applied on the uniform superposition $|*\rangle$. For any other quantum state $|\psi\rangle$ orthogonal to $|*\rangle$ it maps to $-|\psi\rangle$. So FLIP_* is described by:

$$\text{FLIP}_* |\psi\rangle := \begin{cases} |*\rangle & \text{if } |\psi\rangle = |*\rangle \\ -|\psi\rangle & \text{if } \langle \psi | * \rangle = 0 \text{ (orthogonal).} \end{cases}$$

We can construct this FLIP_* by the following quantum circuit:

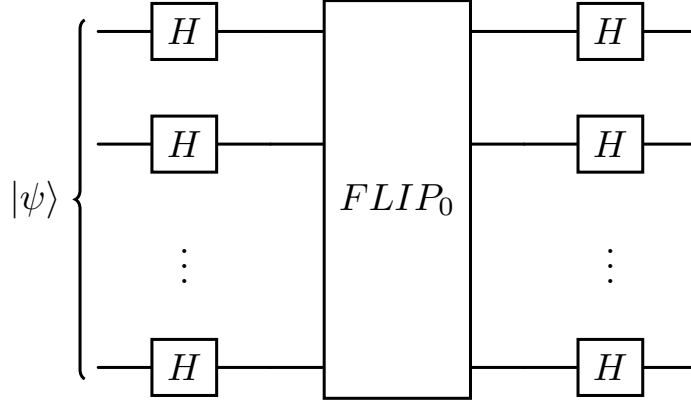


Figure 12.3: The circuit for $FLIP_*$

12.2 The algorithm for searching

The actual algorithm takes a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 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:

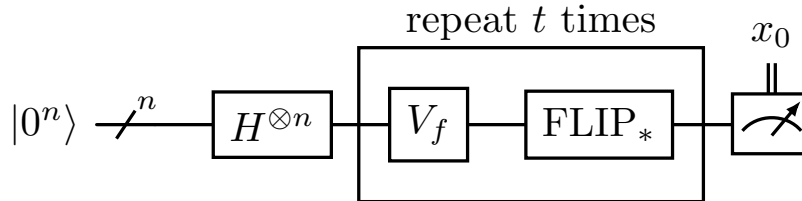


Figure 12.4: The quantum circuit for Grover's algorithm

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 $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$ 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.
6. We do a measurement.

The measurement in step 6 will then give us x_0 with high probability.

12.2.1 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 $|*\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$. We know that $|x_0\rangle$ is part of this superposition, therefore we can rewrite $|*\rangle$ as follows

$$|*\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle = \frac{1}{\sqrt{2^n}} \underbrace{|x_0\rangle}_{\text{good}} + \sqrt{\frac{2^n - 1}{2^n}} \underbrace{\sum_{x \neq x_0} \frac{1}{\sqrt{2^n - 1}} |x\rangle}_{\text{bad}}$$

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:

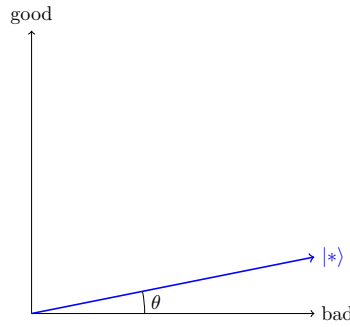


Figure 12.5: Geometric interpretation of $|*\rangle$

The angle 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 $\cos \theta = |\langle * | \text{bad} \rangle| = \frac{\sqrt{2^n - 1}}{\sqrt{2^n}}$. From this we can derive 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 of our desired $|x_0\rangle$ and not change the amplitude to the rest of the state.

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

This looks like this in the geometric interpretation:

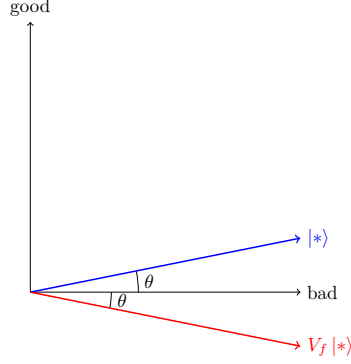


Figure 12.6: Geometric interpretation after V_f

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:

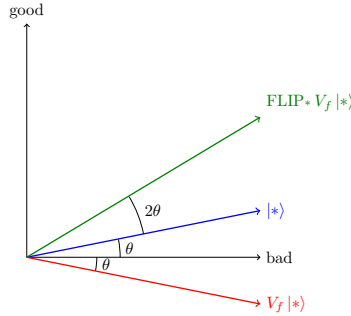


Figure 12.7: Geometric interpretation after FLIP_*

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.

13 Quantum Physics

To further understand how quantum computers work, we take a look at the basics of quantum physics. We will omit or simplify a lot of physics details and strive to give a first impression only.

13.1 Quantum state

As an example, we consider a single atom with one electron.

At any time, the electron can be found in a superposition between (infinitely) many positions, leading to it being in a “fuzzy probability cloud”. Certain superpositions are special¹; we call them orbitals. A photon can then be in superposition between different orbitals (i.e., a linear combination of the different “probability clouds”)².

We can assign each orbital a number, e.g. $0, 1, 2, \dots$. In each orbital the electron has a specific energy level $E_0, E_1, E_2, \dots$. If the electron is in a particular orbital, it is said to be in the corresponding quantum state $|0\rangle, |1\rangle, |2\rangle, \dots$.

Using this notation, we can describe the quantum state of the electron with an infinite one-dimensional vector:

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \end{pmatrix} = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle + \dots$$

¹Roughly speaking, they are eigenvectors of the infinitely-dimensional Hamiltonian of the electron. We explain the concept of Hamiltonians in Section 13.3.

²For mathematical reasons beyond the scope of this exposition, *any* valid “probability cloud” equals a superposition of many orbitals.

The coefficients α_i are complex numbers that represent the probability amplitude of the electron being in the corresponding quantum state $|i\rangle$. With probability $|\alpha_i|^2$ the electron is in the quantum state $|i\rangle$ (if we measure).

13.2 Time evolution of a quantum state

We consider an electron in the quantum state $|n\rangle$ with the energy level E_n . After a time t the electron is in the quantum state

$$e^{-iE_n t/\hbar} |n\rangle,$$

where $\hbar = 1.054571817 \dots \cdot 10^{-34}$ Js is the so-called reduced plank constant. In the following we use $\hbar = 1$ and omit the units for simplicity.

This means that if the electron is in some orbital, and we wait, then the electron is still in the same orbital. But the phase will have changed.

Example: Quantum state evolution

We assume an electron to be in the quantum state

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

Furthermore, we set $E_0 = 0$ and $E_1 = 1$.

If we wait the time $t = \pi$, the quantum state evolves to

$$\begin{aligned} & \frac{1}{\sqrt{2}} e^{-iE_0 t/\hbar} |0\rangle + \frac{1}{\sqrt{2}} e^{-iE_1 t/\hbar} |1\rangle \\ &= \frac{1}{\sqrt{2}} e^{-i0\pi/1} |0\rangle + \frac{1}{\sqrt{2}} e^{-i1\pi/1} |1\rangle \\ &= \frac{1}{\sqrt{2}} e^0 |0\rangle + \frac{1}{\sqrt{2}} e^{-i\pi} |1\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \\ &= |-\rangle. \end{aligned}$$

13.3 Energy / Hamiltonian

From now on, we assume that there are only finitely many orbitals N . (This is not true but allows us to avoid mathematical subtleties.)

So far, we know that the quantum state $|\psi\rangle = \alpha_0|0\rangle + \dots + \alpha_N|N\rangle$ has probability $|\alpha_i|^2$ of being in quantum state $|i\rangle$, which has energy level E_i .

Therefore, the total energy E_{total} of the electron can be calculated as

$$E_{\text{total}} = |\alpha_0|^2 \cdot E_0 + \dots + |\alpha_N|^2 \cdot E_N = \sum_{i=0}^N |\alpha_i|^2 \cdot E_i. \quad (1)$$

There is also another method to calculate the energy of the electron. For this we need the Hamiltonian:

Definition 13.1 (Hamiltonian, diagonal case). For an electron with the basis quantum states $|0\rangle, |1\rangle, \dots, |N\rangle$ and the corresponding energy levels $E_0, E_1, \dots, E_N$ the Hamiltonian is defined by

$$H := \begin{pmatrix} E_0 & 0 & \dots & 0 \\ 0 & E_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E_N \end{pmatrix} = \text{diag}(E_0, E_1, \dots, E_N).$$

With this Hamiltonian H , the total energy E_{total} can be calculated as

$$E_{\text{total}} = \langle \psi | H | \psi \rangle.$$

$\langle \psi | H | \psi \rangle$ is the inner product of $|\psi\rangle$ and $H|\psi\rangle$.

This follows from

$$\begin{aligned} E_{\text{total}} &= \sum_{k=0}^N |\alpha_k|^2 E_k = \sum_{k=0}^N \bar{\alpha}_k E_k \alpha_k \\ &= \sum_{k=0}^N \sum_{l=0}^N \bar{\alpha}_k E_{kl} \alpha_l = \sum_{k=0}^N \bar{\alpha}_k \left(\sum_{l=0}^N E_{kl} \alpha_l \right) \\ &= \sum_{k=0}^N \bar{\alpha}_k \left(H |\psi\rangle \right)_k \\ &= \langle \psi | H | \psi \rangle. \end{aligned}$$

Why is the concept of a Hamiltonian useful? Why don't we just use the formula (1)? The reason is that is general, our basis states $|0\rangle, |1\rangle, \dots$ will not always be pure energy states. Then (1) does not apply. But we can always define some matrix H that describes the total energy:

Definition 13.2 (Hamiltonian). The total energy of an N -dimensional quantum system is described by matrix $H \in \mathbb{C}^{N \times N}$, the Hamiltonian H is always Hermitian (meaning $H^\dagger$

$= H\rangle$. The total energy of any quantum state $|\psi\rangle$ is then $\langle\psi|H|\psi\rangle$,

What Hamiltonian corresponds to a specific quantum state depends on the specific physical system.

13.4 Schrödinger equation

If $H(t)$ is the Hamiltonian of our system at time t , and $\psi(t)$ is the quantum state at time t , then the following holds:

$$\frac{d\psi(t)}{dt} = \frac{i}{\hbar} H(t)\psi(t).$$

This is the time-dependent Schrödinger equation.

From now on we assume that $H(t)$ is constant, i.e. $H(t) = H$ for all t .

For the next step, we need a new mathematical operation: The exponentiation of a matrix.

Definition 13.3 (Matrix exponentiation). For a diagonal matrix D , the exponentiation is defined as:

$$e^D := \begin{pmatrix} e^{D_{11}} & 0 & \dots & 0 \\ 0 & e^{D_{22}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{D_{nn}} \end{pmatrix}.$$

Let A be a diagonalizable matrix, then there exists a unitary matrix U and a diagonal matrix D such that $A = U^\dagger D U$. In this case the exponentiation of A is defined as

$$e^A = e^{U^\dagger D U} := U^\dagger e^D U.$$

The eigenvectors of H are called “energy eigenstates”. They have a very simple time evolution:

If $\psi(0)$ is an energy eigenstate of H with the total energy E as eigenvalue, meaning $H\psi(0) = E\psi(0)$, then the solution of the time-dependent Schrödinger equation is

$$\psi(t) = e^{-iEt/\hbar}\psi(0).$$

We can check the plausibility of this result by substituting it into the time-dependent

Schrödinger equation:

$$\begin{aligned}
\frac{\partial \psi(t)}{\partial t} &= \frac{\partial}{\partial t} \left(e^{-i\frac{E}{\hbar}t} \psi(0) \right) = \frac{\partial}{\partial t} \left(e^{-i\frac{E}{\hbar}t} \right) \psi(0) \\
&= \left(-i\frac{E}{\hbar} \right) e^{-i\frac{E}{\hbar}t} \psi(0) = -\frac{i}{\hbar} e^{-i\frac{E}{\hbar}t} \left(E\psi(0) \right) \\
&= -\frac{i}{\hbar} e^{-i\frac{E}{\hbar}t} \left(H\psi(0) \right) = -\frac{i}{\hbar} H \left(e^{-i\frac{E}{\hbar}t} \psi(0) \right) \\
&= -\frac{i}{\hbar} H\psi(t).
\end{aligned}$$

This shows that ψ is indeed a solution. We omit the proof that it is unique.

14 From Quantum Physics to a Quantum Computer

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

As an example, we consider the construction of a simple hypothetical single-qubit quantum computer. We encode the qubit using the excitation of an electron. For this purpose, we assume that the electron has only the two basis states $|0\rangle$ and $|1\rangle$, where $|0\rangle$ represents the ground state. The states have the energy levels π and 2π .

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 measure the qubit.
3. We need to be able to apply a unitary on the qubit.

We look into how to construct a unitary. The cooling and measuring operations are out of scope of this chapter.

The Hamiltonian H of our single-qubit quantum computer is given by:

$$H = \begin{pmatrix} \pi & 0 \\ 0 & 2\pi \end{pmatrix}.$$

According to Section 13.2 (with our assumption $\hbar = 1$), the time evolution of our quantum computer is:

$$\begin{aligned} |0\rangle &\mapsto e^{-i\pi t} |0\rangle, \\ |1\rangle &\mapsto e^{-i2\pi t} |1\rangle. \end{aligned}$$

In other words, after time t the quantum state is multiplied with the unitary

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

Thus, waiting for a duration t is equivalent to applying the unitary U_t .

As an example, waiting for one second corresponds to:

$$U_1 = \begin{pmatrix} e^{-i\pi} & 0 \\ 0 & e^{-i2\pi} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} = -Z,$$

where Z is the Pauli- Z -gate.

This means that the unitary $-Z$ is applied every second “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 Pauli- Z -gate.

How can we obtain new unitaries beyond just Z ?

One solution is to apply external energy to the system, for example shining a laser on the atom, applying a magnetic field, This modifies the system’s Hamiltonian from H to $H + \delta H$.

As an example, we consider

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

δH is much bigger than H , so $H + \delta H \approx \delta H$.

Then after time t , the time evolution unitary becomes

$$U'_t = e^{-i\frac{H+\delta H}{\hbar}t} \approx e^{-i\frac{\delta H}{\hbar}t} =: U_t^{\text{approx}}.$$

For $t = \frac{\pi}{2000}$, this gives

$$U_{\frac{\pi}{2000}}^{\text{approx}} = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix} = -i \cdot X$$

where X is the Pauli- X -gate. Again the factor $-i$ does not make a physical difference as it is a global phase factor, so the $-iX$ is physically equal to the Pauli- X -gate.

In reality, the exact time evolution unitary would be (up to a position of three digits)

$$U'_{\frac{\pi}{2000}} = \begin{pmatrix} 0.002i & -0.007 - 0.999i \\ -0.007 - 0.999i & -0.002 \end{pmatrix}$$

which approximates the unitary $-iX$ with only an error up to about 0,2%.

15 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.

We look at single atoms with one electron “orbiting” the nucleus. The quantum state is encoded as the excitation (orbital) of the electron.

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

15.1 Setup

The setup for our quantum computer looks as follows:

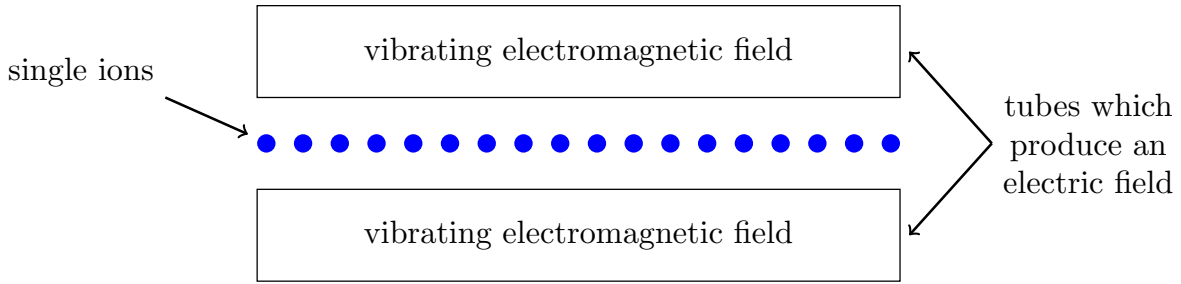


Figure 15.1: Setup for the ion based quantum computer

The ions are trapped between two electromagnetic fields.

The state of each ion can be $|0\rangle$ or $|1\rangle$. (We ignore higher excitation levels.) In addition, the whole chain of ions is vibrating as a whole. There are different amounts of vibration and depending on the amount, we have vibration states $|0\rangle$, $|1\rangle$ and $|2\rangle$. (Again we ignore higher excitation states.) So our total quantum system is described by:

$$\mathbb{C}^3_{\text{vibration}} \otimes \mathbb{C}^2_{\text{ion}_1} \otimes \dots \otimes \mathbb{C}^2_{\text{ion}_n}.$$

E.g. the vibration is $|2\rangle$, the first ion is $|+\rangle$, and the second ion is $|0\rangle$, then the state is

$$\frac{1}{\sqrt{2}} |200\rangle + \frac{1}{\sqrt{2}} |210\rangle.$$

15.2 Operations

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

15.2.1 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 .

15.2.1.1 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. 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.

15.2.1.1.1 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 $E_{\text{big}} - E_1$

and $E_{\text{big}} - E_2$ but not with $E_{\text{big}} - E_0$. This will “shoot” all the low energy electrons from E_1 and E_2 to a higher level where they will either fall down to E_1 or 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 $|0\rangle$.

15.2.2 Measurements

Next 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 $E_{\text{aux}} > E_0, E_1$.

We now shine a laser with the energy $E_{\text{aux}} - E_0$ 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 . If the state is $|1\rangle$ the ion lights up. If the state is $|1\rangle$, no photon get absorbed and light can be detected.

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

15.2.3 Unitaries

Finally 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.

15.2.3.1 Single qubit gates

We have already discussed in Chapter 14 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 $\Delta E = E_1 - E_0$ 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. $U_{\pi, \frac{\pi}{2}}$ gives the Pauli- X -gate, the gate $U_{\frac{\pi}{2}, \frac{\pi}{2}}$ followed by $U_{\pi, \frac{\pi}{2}}$ gives the Pauli- Z -gate and the gates $U_{\frac{\pi}{2}, \frac{\pi}{4}}$ and $U_{\pi, \frac{\pi}{2}}$ gives the Hadamard-gate.

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.

15.2.3.2 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 $|0\rangle$ and $|1\rangle$ and the overall vibration is described by another quantum system $|0\rangle, |1\rangle$ and $|2\rangle$. So looking at the k -th ion we have the basis:

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle, |20\rangle, |21\rangle.$$

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 $\omega + \varphi$, where ω is the frequency corresponding to $\Delta E = E_{20} - E_{11}$, onto the ion then the following Hamiltonian is applied:

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

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

$$U = e^{-iH_0\pi} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

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

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

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

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

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

So we have created the unitaries C-Z, SWAPLIKE and $\text{SWAPLIKE}^\dagger$. 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:

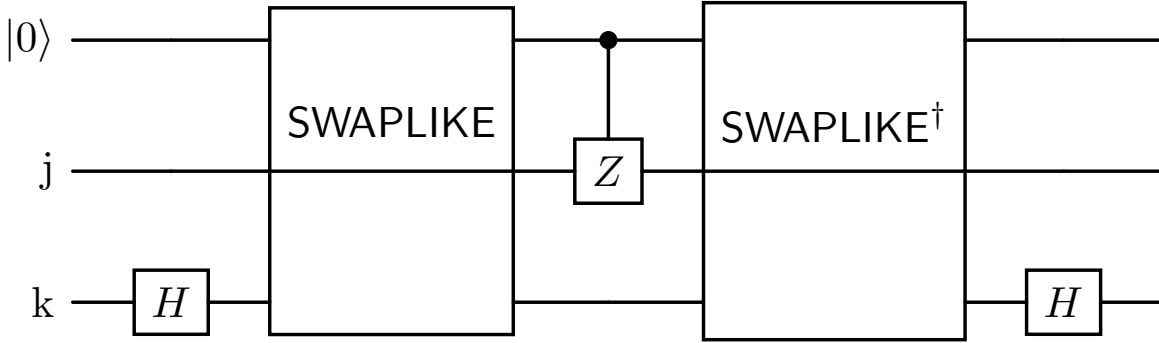


Figure 15.2: Circuit for CNOT

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

16 Universal set of gates

We have seen that various gates are possible with an ion-based quantum computer. For other architectures, other gates may be possible. At this point, the question arises: Which set of gates are necessary to generate all possible gates. Such a set is referred to as universal set of gates.

Formally, this means:

Definition 16.1 (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 n , for all unitaries $U \in \mathbb{C}^{2^n \times 2^n}$ and for all ϵ there exists a circuit C consisting of elements of G without auxiliary qubits such that for all quantum states ψ it holds that $\|U\psi - C\psi\| \leq \epsilon$.

We call G universal set for single-qubit gates when this holds for $n = 1$.

16.1 Pauli gates

We recall the Pauli matrices X , Y and Z (see Definition 7.2). If we only have access to the X and Y gates, can we construct the Z gate from them?

We can combine the Y and X gates, 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 Z gate from the X and Y gate up to a global phase factor. With some simple math, one can also show that it is possible to get a X -gate (up to a global phase factor) from Y and Z , and a Y gate (up to a global phase factor) 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. So they cannot be used to create entanglement. For example, the CNOT gate can create entanglement, and therefore it cannot be constructed using only Pauli gates.

But maybe it is possible to create all single-qubit gates from the Pauli matrices. Unfortunately, this is also not true. The product of a Pauli matrix is always either another Pauli matrix (up to a global phase factor) or the identity matrix.

In conclusion, we cannot get a universal set of gates nor even a universal set for single-qubit gates from the Pauli matrices. We must therefore consider a different set of gates.

16.2 Other known gates

We also know the gates CNOT, Toffoli (i.e. CNOT with two control wires), X and SWAP. These gates are sufficient for constructing U_f .

Are these gates a universal set of gates?

The answer is also no. Each of those gates G performs an operation of the form:

$$G|x\rangle = |\text{something}\rangle,$$

where $|\text{something}\rangle$ is another computational basis state (ket-state).

This means that any combination of these gates can only map one computational basis state to another. Therefore, they cannot create superposition or arbitrary quantum states, which are essential. As such, these gates are not sufficient to form a universal set of gates.

Note that any combination of those gates computes $|x\rangle \rightarrow |f(x)\rangle$ for some classically easy to compute f . Therefore these gates also provide no quantum speedup.

16.3 Rotation gates

We look at a different type of gates: The rotational gates R_X, R_Y and R_Z . These are defined as:

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

These represent infinitely many gates. They are also very natural on physical quantum computers. For example, $R_X(\theta)$ corresponds to applying the X gate for the time θ . Analogous holds for $R_Y(\theta)$ and $R_Z(\theta)$.

It turns out that any single-qubit gate can be decomposed into these rotational gates. However, the construction of an arbitrary multiple qubit gate from this is not possible. This leads to the following result:

Theorem 16.1 (Rotation gates). *The set $\{R_X(\theta), R_Y(\theta), R_Z(\theta) : \theta \in [0, 2\pi]\}$ is a universal set for single-qubit gates.*

Furthermore, the following holds:

Theorem 16.2 (Single qubit gates and CNOT). *The set of $\{all\ single\ qubit\ gates, CNOT\}$ is a universal set of gates.*

Combining both theorems, we obtain the following corollary.

Corollary 16.1 (Rotation gates and CNOT). *The set of $\{R_X(\theta), R_Y(\theta), R_Z(\theta), CNOT : \theta \in [0, 2\pi]\}$ is a universal set of gates. Also only two of the three rotational gates are enough for a universal set of gates.*

16.4 Clifford gates

We look at another set of gates called *Clifford* gates. This set consists of CNOT, the Hadamard-gate H and S , which is defined as:

$$S := \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

While there are many gates, which can be constructed from the set of Clifford gates, it is not a universal set of gates. For example, there exists a gate called T , which cannot be constructed from the Clifford gates. The T gate is defined as:

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

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

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

Theorem 16.3 (Gottesmann-Knill theorem). *A quantum circuit only using Clifford gates can be efficiently simulated on a classical computer.*

However, if we add T , we obtain a universal set of gates:

Theorem 16.4 (Clifford gates and T). *The set of all Clifford gates and T is universal.*

There exists also another example of a universal set of gates:

Theorem 16.5 (Toffoli and H). *The set of $\{Toffoli, H\}$ is universal.*

16.5 Gottesmann-Knill theorem

We take a closer look at why Theorem 16.3 works. For this, we need two definitions:

Definition 16.2 (Stabilizing). Given a set M of unitaries, the state ψ is stabilized by M iff for all $U \in M$ is $U\psi = \psi$.

Example: Stabilizing $|0\rangle$

The quantum state $|0\rangle$ is stabilized by the set $M = \{Z\}$, because $Z|0\rangle = |0\rangle$.

The state $|1\rangle$ is not stabilized by the set M , because $Z|1\rangle = -|1\rangle$.

The only states stabilized by M are $c \cdot |0\rangle$ for $c \in \mathbb{C}$. So $|0\rangle$ is uniquely stabilized by M (up to a global phase factor).

Let $Pauli$ be the set of all n -qubit tensor products of X, Y, Z, I and the factors ± 1 and $\pm i$. For $n = 4$, P contains for example $X \otimes I \otimes Z \otimes Z$ and $-iX \otimes Y \otimes I \otimes I$. With this we can define a stabilizer state:

Definition 16.3 (Stabilizer state). We call ψ a stabilizer state iff it exists a $M \subseteq Pauli$ such that ψ , and only ψ (up to a global phase factor), is stabilized by M .

Example: Stabilizing $|0\rangle^n$

The quantum state $|0\rangle^n = |0\rangle \otimes \dots \otimes |0\rangle$ is uniquely stabilized by $M_0 = \{Z \otimes I \otimes \dots \otimes I, I \otimes Z \otimes \dots \otimes I, I \otimes I \otimes \dots \otimes Z\}$.

Therefore the quantum state $|0\rangle^n$ is a stabilizer state and $|M_0| = n$.

The notation $\psi \sim M$ means that ψ is uniquely stabilized by M .

The big trick is now that if $\psi \sim M$, it follows that $U\psi \sim M^U := \{UVU^\dagger : V \in M\}$. This follows from

$$UVU^\dagger U\psi = UV\psi = U\psi.$$

Now we can use a useful fact: For any Clifford gate G , if $M \subseteq Pauli$, then also $M^G \subseteq Pauli$.

To prove the Gottesmann-Knill theorem (Theorem 16.3) we want to simulate the result of applying a sequence $G_1, G_2, \dots, G_l$ of Clifford gates to $|0\rangle^n$ and measure the resulting state. This simulation algorithm works as follows:

1. Set $M_0 := \{Z \otimes I \otimes \dots \otimes I, I \otimes Z \otimes \dots \otimes I, \dots, I \otimes I \otimes \dots \otimes Z\}$. (This is the stabilizer of the state $|0\rangle^{\otimes n}$.)
2. For each $i \in \{1, 2, \dots, l\}$ set $M_i := M_{i-1}^{G_i}$. (M_i is the stabilizer of the state after applying G_i .)

3. To calculate the measurement outcomes, do linear algebra (not further specified here).

Since M_i will always be a set of n tensor products of n Pauli matrices, this gives a polynomial time algorithm for simulating Clifford circuits. Additionally, we can simulate a measurement of the final state (details omitted). So Clifford gates alone do not give a quantum speedup.

17 Quantum error correction

One of the biggest challenges with quantum computing is noise. This means for example that when we apply a unitary to our quantum system, this unitary might not behave exactly as we expect. To overcome this challenge we introduce quantum error correction.

Classically error correction works as follows: We start with some message w , which we encode as c with some error correcting code. This c is then transmitted over some noisy communication channel, which introduces some error. The recipient therefore receives a slightly changed version c' . We now try to recover c from c' and then hopefully get w without any error. This can be visualized as follows:

$$\text{word } \omega \xrightarrow{\text{encode}} \text{codeword } c \xrightarrow{\text{noise}} c' \xrightarrow{\text{recovery}} \text{codeword } c \xrightarrow{\text{decode}} \text{word } \omega.$$

Important parameters of any error correcting code are the length of the message which we encode $|w|$, the length of our encoded message $|c|$ and the number of errors, which *recovery* can correct.

17.1 Repetition code

One of the simplest classical error correcting codes is the repetition code. The idea is very straightforward: The message ω is one bit, and we simply repeat it three times. So we encode a 0 as 000 and a 1 as 111. With this encoding, we can correct one error on the transmitted c by just picking the bit in the majority. E.g. if the received message is 010 due to an error, we can still correct it to 000.

To summarize $|\omega| = 1$, $|c| = 3$ and one error can be corrected.

We now try to construct a similar error correcting code for quantum computers. To do so, we need to overcome two challenges:

1. In the classical repetition code, we looked at all bits to decide which of them is in the majority. In the quantum world this would mean a measurement of our system, which would destroy our superposition. We therefore need to find a way of measuring the error without measuring the underlying data.
2. The errors introduced in a quantum system are way more complex. In a classical setting, a single bit can have two states, where one is the correct state and the other is the correct state flipped. Contrary a single qubit can have infinitely many different states and therefore also infinitely many possible errors.

17.1.1 Bit flip code

We first consider a direct analogue to the repetition code, the so-called bit flip code. It encodes $|0\rangle$ as $|000\rangle$, and $|1\rangle$ as $|111\rangle$.

We can encode a quantum state $|a\rangle$ using the following circuit:

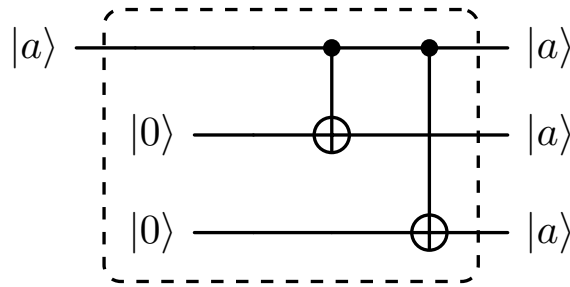


Figure 17.1: Encoding for the bit flip code

Note that of course we can also encode non-basis states. E.g. $|+\rangle$ as $\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle$ (not as $|+\rangle \otimes |+\rangle \otimes |+\rangle$).

Let's assume a single X -error happened. By this, we mean that an X -gate was applied to one of the code qubits but we don't know which. When we measure all three wires we can detect if and on which wire the X -error has happened. If all bits are the same, no error occurred. If one bit is different than the other two bits, the error must have occurred on this wire. The problem is that we then also measure the original message and not only the error position. So we would destroy a superposition, in an encoding of $|+\rangle$ for example. To get rid of this problem we can use the following circuit:

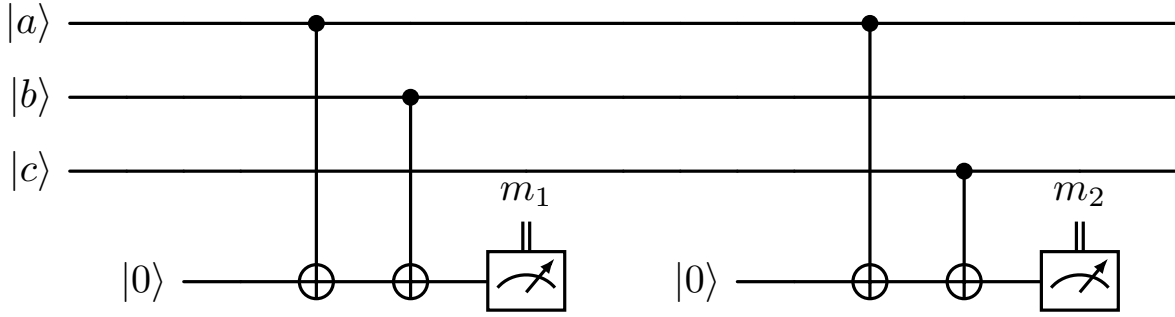


Figure 17.2: Setup for the bit flip code

Note that the wires start with the three different qubits $|a\rangle$, $|b\rangle$ and $|c\rangle$, because they might be different through an error. It is $m_1 = a \oplus b$ and $m_2 = a \oplus c$. Now we know the following:

- If $m_1 = m_2 = 0$: no error,
- If $m_1 = 0$ and $m_2 = 1$: error on wire 3,
- If $m_1 = 1$ and $m_2 = 0$: error on wire 2,
- If $m_1 = 1$ and $m_2 = 1$: error on wire 1.

Since we measure no information on the first three wires, we can detect where the X -error occurred without losing information. (To prove this, one would of course need to explicitly calculate what happens when we start with a superposition.) And then we just apply X to that wire to correct the error.

17.1.2 Phase flip code

Using this bit flip code we can only correct X -errors. But what about e.g. Z -errors? For these errors, we can use a similar concept by encoding $|0\rangle$ as $|+\rangle \otimes |+\rangle \otimes |+\rangle$ and $|1\rangle$ as $|-\rangle \otimes |-\rangle \otimes |-\rangle$. This encoding is possible with:

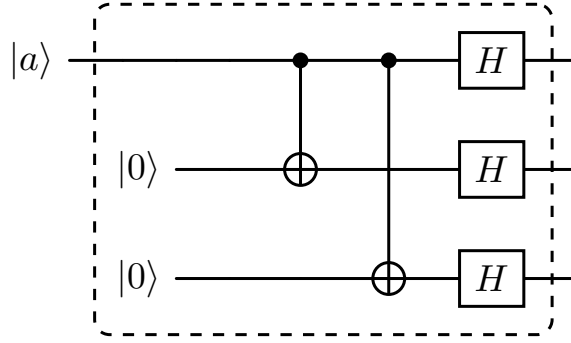


Figure 17.3: Encoding for the phase flip code

The position of the error can be determined like for the bit flip code using this circuit:

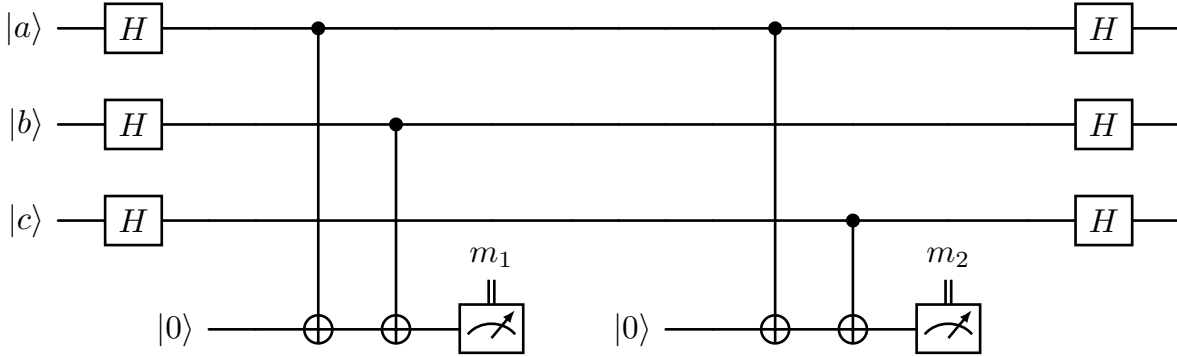


Figure 17.4: Setup for the phase flip code

17.2 Shor's code

We now have an error correcting code for X -errors and Z -errors based on repetition. Using both of these techniques, we can create an error correcting code for XZ -errors. This is called Shor-code.

The idea is to encode one qubit first with the phase flip code and then encode each of the three new qubits using the bit flip code. This encodes the qubits $|0\rangle$ and $|1\rangle$ as follows:

$$\begin{aligned}
 |0\rangle &\mapsto \frac{1}{2\sqrt{2}} \left(|000\rangle + |111\rangle \right) \otimes \left(|000\rangle + |111\rangle \right) \otimes \left(|000\rangle + |111\rangle \right), \\
 |1\rangle &\mapsto \frac{1}{2\sqrt{2}} \left(|000\rangle - |111\rangle \right) \otimes \left(|000\rangle - |111\rangle \right) \otimes \left(|000\rangle - |111\rangle \right).
 \end{aligned}$$

With this encoding we can correct an X error. We do this by handling each block as an individual bit flip code. If an Y error occurred the three blocks are different and we can use a variation of the error handling from the bit phase code.

So we can use the error correcting techniques from above to correct X -errors and Z -errors. Since Y can be constructed from X and Z , we also can correct Y -errors with this technique. All in all the Shor-code handles one X, Y or Z -error.

Without proof we introduce a theorem:

Theorem 17.1 (Correction of arbitrary errors). *If a quantum error correction corrects single X, Y and Z -errors, it will correct arbitrary single qubit errors, if the error correction is of the form measure the error and then correct.*

From this theorem a corollary follows:

Corollary 17.1 (Shor-code). *The Shor-code corrects arbitrary 1-qubit errors.*

17.3 Steane code

Another code is the Steane code. One qubit is encoded to seven qubits and it can correct one error:

$$\begin{aligned}
|0\rangle &\mapsto \frac{1}{\sqrt{8}} \left(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \right. \\
&\quad \left. + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle \right), \\
|1\rangle &\mapsto \frac{1}{\sqrt{8}} \left(|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \right. \\
&\quad \left. + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle \right).
\end{aligned}$$

This code has some nice additional properties useful for fault-tolerant computation (see Section 18.1).

18 Fault-tolerant computation

Error correcting codes are effective for storage and transfer of data. However, they are not directly suitable for continuous computation. In this context two problems arise:

1. How to apply an operation on logical qubits? For example, suppose we want to apply a Hadamard-gate to a logical qubit encoded using Shor's code. A naive approach would be to decode the nine physical qubits back to obtain a single logical qubit, apply the Hadamard-gate, and then encode back to the nine physical qubits:

$$\xrightarrow{\text{decode}} \xrightarrow{\text{operation}} \xrightarrow{\text{encode}} .$$

This is problematic because errors may occur between encoding and decoding. The error correcting code cannot protect against this. Therefore, we must find another to apply operations directly on the encoded logical qubit.

2. Even if we solve the first problem: The new operations (which may be more complex) could themselves introduce errors – maybe too many to correct.

18.1 Operations on logical/physical qubits

We begin by solving the first problem: Performing an operation directly on the encoded qubit without decoding and encoding again.

Let us recall Shor's code. We write the encoded qubit $|0\rangle$ as $|\tilde{0}\rangle$ and the encoded qubit $|1\rangle$ as $|\tilde{1}\rangle$. Suppose we want to apply an X -gate to the logical qubit, i.e. construct an operation $\tilde{X}$ that acts on the nine physical qubits such that:

$$\begin{aligned}\tilde{X} : |\tilde{0}\rangle &\mapsto |\tilde{1}\rangle, \\ \tilde{X} : |\tilde{1}\rangle &\mapsto |\tilde{0}\rangle.\end{aligned}$$

This is possible by applying Z to the first qubit on each block (i.e. physical qubits 1, 4 and 7) or applying Z to all nine physical qubits. Similarly the operation $\tilde{Z}$ works also by applying $X^{\otimes 9}$. We call such easily to implement gates **transversal**:

Definition 18.1 (Transversal gate). A gate is called transversal if it can be implemented by applying identical gates to each physical qubit separately.

(Note that there are slightly different definitions of “transversal”. Some only call a gate G transversal if it can be implemented by applying G to each qubit.)

However, this method does not work for all unitaries. For example, the Hadamard-gate H cannot be implemented on Shor's code as $\tilde{H}$ in such a simple way.

A better alternative is the Steane code, which encodes a logical qubit into seven physical qubits and can correct one error (see Section 17.3).

For each gate $G \in \{Y, X, Z\}$ the logical operation G is implemented as the physical operation $\tilde{G} = G^{\otimes 7}$. And S is implemented by $S^{\otimes 7}$ followed by $Z^{\otimes 7}$. Similarly, the logical $\widetilde{CNOT}$ -gate can be implemented using seven $CNOT$ -gates. Thus these gates are all **transversal**.

As shown in Section 16.4 in the previous chapter, these gates alone do not form a universal set of gates. However, if the physical $\tilde{T}$ -gate can also be implemented, we achieve a universal set of gates (see Theorem 16.4).

Unfortunately, the T -gate is not transversal. And we know from Section 16.4 and Theorem 16.4 that we need T to get a universal set of gates. Nevertheless, it is known that a more complex method exists to implement $\tilde{T}$ on the Steane code. (We omit the details here.)

Conclusion: With the Steane code, it is possible to implement all quantum circuits on encoded qubits without decoding and encoding again.

18.2 Fault-tolerant gates

While the Steane code solves the first problem, we now look at the second problem: We need to compute on encoded qubits using physical gates with an error probability p such that error probability per logical gate is $\ll p$; ideally arbitrary small.

In the following, we will focus on applying unitaries and will ignore initialization and measurement.

To solve the problem, two ingredients are necessary:

1. A quantum error correcting code for one logical qubit, capable of correcting one error (e.g., the Steane code).
2. A “fault-tolerant implementation” of logical gates. The definition follows.

Let us examine the following circuit:

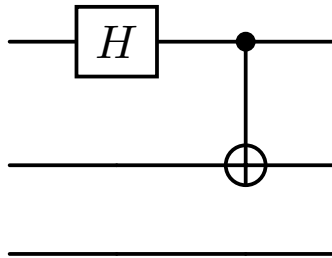


Figure 18.1: Example situation for fault-tolerance to apply

To make this circuit fault-tolerant, we replace it with:

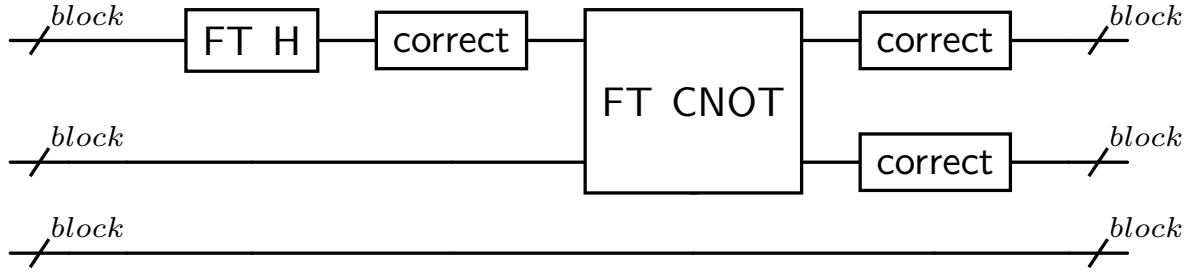


Figure 18.2: Example fault-tolerant implementation

In this transformation:

- Each physical qubit is encoded as a block of qubits using a quantum error correcting code (e.g., the Steane code).
- Each gate is replaced by a fault-tolerant implementation (e.g., H by fault-tolerant (FT) H).
- An error correction step is added after each gate.

For this construction to work, the following three requirements must be satisfied:

- (A) For each fault-tolerant gate implementation: If there is at most one physical error in the inputs, then there is at most one physical error per output block. (Transversal gates satisfy this automatically.)
- (B) For each fault-tolerant gate implementation: If the input contains no error and at most one physical gate error occurs, then there is at most one physical error per output block. (Transversal gates satisfy this automatically.)
- (C) For each error correction procedure: If there is no error in the inputs and at most one error occurred, then there is at most one error in the output.

18.2.1 Analysis of a circuit

Let us analyze the following circuit for $CNOT$ (an analogous analysis works for other gates and gives the same result):

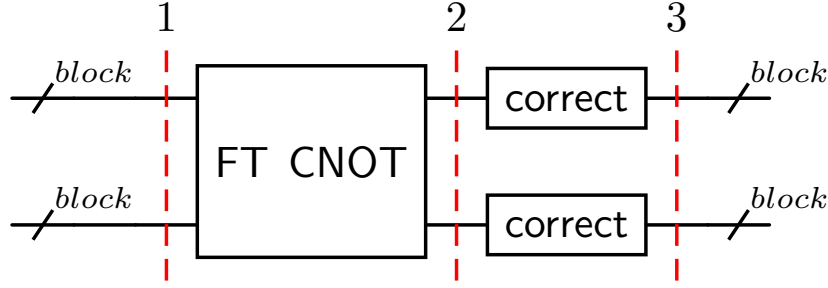


Figure 18.3: Example to analyze fault-tolerance

For the analysis we make three assumptions:

1. Each physical gate has an error probability of p (i.e., with probability p , the gate does some thing arbitrary; with probability $1 - p$ it does what it is supposed to).
2. All errors occur independently.
3. The circuit satisfies the three requirements for fault-tolerance listed above.

Let G be the number of physical gates used in this circuit.

We will now analyze what errors got introduced by this gate with what probabilities:

Position 1 (before the FT CNOT):

- P_1 : Probability that exactly one error is present (in all blocks together).
- P_2 : Probability that at least two errors are present.

Position 2 (after the FT CNOT):

P'_1 : Probability of at most one error per block (but at least on error in total). This can occur in three ways:

- No error before the FT CNOT, and an error occurred during the FT CNOT: Probability of this is $\leq G \cdot p$.
- One error already existed before the FT CNOT (probability P_1), and no additional error occurred during FT CNOT: Probability of this is $\leq P_1$.
- Also, it could happen that there were at least two errors, and we “accidentally” correct some. We ignore this case because those cases will be counted in the case (P'_2) anyway and a 100% correct formulation would be very complicated.

So:

$$P'_1 \leq Gp + P_1.$$

P'_2 : Probability of at least two errors in at least one block. This can occur in five ways:

- No error before the gate, and a single error during the gate: Cannot lead to two errors by (B).
- No error before the gate, but at least two errors during the gate: Probability of this is $\leq (Gp)^2$.
- One error before the gate, and no error during the gate: Cannot lead to two errors by (A).
- One error on the input of the gate, and one during the gate: Probability of this is $\leq P_1 \cdot Gp$.
- At least two errors before the gate: Probability of this is $\leq P_2$.

So:

$$P'_2 \leq (Gp)^2 + P_1 \cdot Gp + P_2.$$

Position 3 (after the error correction):

P''_1 : Probability of exactly one error. This can occur in one way:

- No error at position 2, but one error happens during error correction: Probability of this is $\leq Gp$.

So:

$$P''_1 \leq Gp.$$

P''_2 : Probability of at least two errors. This can occur in five ways:

- No error at position 2, and one error occurs during the correction: Cannot lead to two errors by (C).
- No error at position 2, but at least two errors occur during the correction: Probability of this is $\leq (Gp)^2$.
- One error at position 2, and no error occurs during the correction: Does not happen because the code corrects one error.
- At most one error per block at position 2, and at least one error occurs during the correction: Probability of this is $\leq P'_1 \cdot Gp \leq (P_1 + Gp) \cdot Gp = P_1 \cdot Gp + (Gp)^2$.
- Two or more errors at position 2: Probability of this is $\leq P'_2 \leq (Gp)^2 + P_1 \cdot Gp + P_2$.

So:

$$P''_2 \leq (Gp)^2 + P_1 \cdot Gp + (Gp)^2 + (Gp)^2 + P_1 \cdot Gp + P_2.$$

If we additionally assume $P_1 \leq Gp$, we have:

$$P''_2 \leq 5 \cdot (Gp)^2 + P_2.$$

At the beginning of a large circuit consisting of l fault-tolerant implemented gates, we have $P_1 = 0 \leq Gp$, and thus by induction, we have $P''_2 \leq 5 \cdot (Gp)^2 + P_2$ after each fault-tolerant implemented gate. So the final P''_2 will be $\leq l \cdot 5 \cdot (Gp)^2$. So per fault-tolerant gate, we have a probability of $\leq 5 \cdot (Gp)^2 + P_2$ of introducing an uncorrectable error.

18.2.2 Development of the error probability

So P_2 increases by $(Gp)^2$ per logical gate. So for a p -error physical gate, we get a $5(Gp)^2$ -error logical gate assuming one correctable error in the output is acceptable.

For some nontrivial, but not arbitrary small p , it is $5(Gp)^2 < p$.

Let try an example with $G = 50$ and $p = \frac{1}{10000}$:

$$5(Gp)^2 = \frac{5 \cdot 50^2}{10000^2} = \frac{1}{8000}.$$

In this case the logical error is higher than the physical one, so the fault-tolerance fails. Now let change the example to $p = \frac{1}{15000}$:

$$5(Gp)^2 = \frac{5 \cdot 50^2}{15000^2} = \frac{1}{18000}.$$

Now the logical error is lower, so the fault-tolerance works.

One easily sees that whenever $p < \frac{1}{5G^2}$, the error will be reduced.

Reducing the error arbitrarily:

Using an fault-tolerant implementation as described above, we can construct a quantum computer with logical error probability $p_{i+1} < p_i$ from a quantum computer with error probability p_i , as long as $p_i < \frac{1}{5G^2}$. But that means that if we start with a quantum computer with logical error probability $p_1 < \frac{1}{5G^2}$, we can construct one with logical error probability p_2 , and based on that with logical error probability p_3 (adding one more layer of fault-tolerant computation), and from that with logical error probability p_4 and so on. And $p_1 > p_2 > p_3 > p_4 > \dots$ (and a slightly more careful analysis shows that the error drops fast).

This leads us to the following theorem:

Theorem 18.1 (Threshold theorem). *If the physical gate error rate is below a certain threshold, then the logical error rate can be made arbitrary small using fault-tolerant quantum computation.*

The threshold (here $\frac{1}{5G^2}$) is believed to lie between 0.1% and 1% in reality.

19 Quantum complexity

So far, we know that quantum computers can potentially be exponentially faster than classical computers. However, the exact extent of this advantage remains unknown.

For instance, can quantum computers solve NP-hard problems in polynomial time?

We start with the common complexity class NP (non-deterministic polynomial time):

Definition 19.1 (The complexity class NP). A language L is in NP if and only if there exists a deterministic classical algorithm A running in polynomial time such that:
If x is a YES-Instance, then a ω exists such that $A(x, \omega) = 1$,
if x is a NO-Instance, then no ω exists such that $A(x, \omega) = 1$ (in other words: for all ω is $A(x, \omega) \neq 1$).

Note that this definition is a bit informal because we do not explicitly specify in which argument A is polynomial, and how the length of ω is bounded etc. While important for a precise analysis, we omit such details in this chapter.

This means than an NP-algorithm can efficiently verify YES-instances and never incorrectly accepts a NO-instance.

Example: SAT (Satisfiability)

Given a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ (e.g. represented as a circuit), the SAT problem is defined as:

YES-Instance: A z exists such that $f(z) = 1$,

NO-Instance: No z exists such that $f(z) = 1$.

The SAT problem is in NP.

Furthermore, SAT is NP-complete. This means that if we can solve SAT in polynomial time, we can solve any problem in NP in polynomial time.

The key question is whether quantum computers can solve SAT (or any other NP-complete problem) in polynomial time. This is closely related to the famous “P = NP” problem.

Although the question remains unsolved even for classical computers, there are weak indications for quantum computers suggesting a negative answer.

19.1 Oracle lower bounds

It is known that quantum computers cannot solve SAT in polynomial time relative to a black-box function (oracle). That is, if the only thing the algorithm can do with f is to evaluate it. (No “decompilation” or similar.)

For classical computers, it is easy to see that $\Theta(2^n)$ queries are necessary and sufficient to distinguish YES- and NO-instances in the black-box case. For quantum computers, this bound is reduced to $\Theta(2^{\frac{n}{2}})$.

This is achievable with Grover’s algorithm: For a function f , Grover’s algorithm finds a z such that $f(z) = 1$, if such a z exists. This can be used to distinguish YES- or NO-instances with arbitrary small error.

We now ask: Could there be even faster quantum algorithms?

Assume we have a quantum algorithm A^f that uses arbitrary quantum operations (unitaries, auxiliary qubits, measurements, ...) and can query the oracle $V_f : |x\rangle \rightarrow (-1)^{f(x)} |x\rangle$.¹

Theorem 19.1 (Quantum SAT lower bound). *If a quantum algorithm can distinguish YES- and NO-instances of SAT with constant error, it must perform at least $\Omega(2^{\frac{n}{2}})$ queries to V_f .*

More precisely:

If

$$\Pr[A^{f_x}() \rightarrow 1 : x \xleftarrow{\$} \{0,1\}^n] - \Pr[A^{f_0}() \rightarrow 1] \geq \text{const} > 0$$

and the algorithm makes q oracle queries, then

$$q \in \Omega(2^{\frac{n}{2}}).$$

Here f_x is the function $f_x(x) = 1$ and $f_x(z) = 0$ for all $z \neq x$. And $f_0(z) = 0$ for all z .

19.1.1 Proof sketch

To give a proof sketch for Theorem 19.1, we look at the circuits for A^{f_x} and A^{f_0} :

¹One may wonder why V_f and not U_f (Section 12.1.1). In Section 12.1.1 we saw that from U_f , we can construct V_f . Vice versa, given V_f and the classical knowledge of a single $f(x)$, we can construct U_f . So it does not matter much, which we choose.

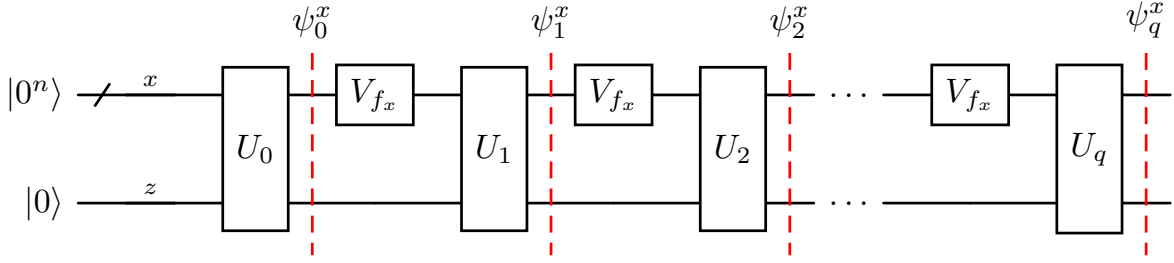


Figure 19.1: An execution of A_x^f

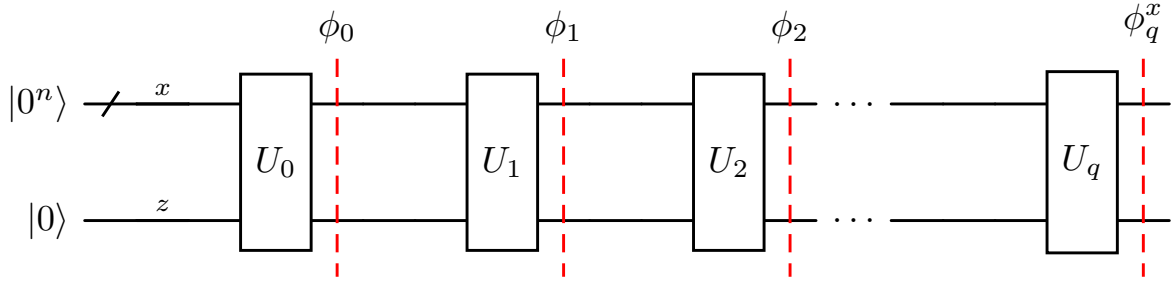


Figure 19.2: An execution of A_0^f

We analyze the final states ψ_q^x and ϕ_q of the algorithms/circuits. We define the average squared distance between the final states:

$$D := \sum_{x \in \{0,1\}^n} 2^{-n} \|\psi_q^x - \phi_q\|^2.$$

We aim to show: $D \leq \frac{4q^2}{2^{-n}}$. For this we need a lemma:

Lemma 19.1 (Quantum inequality). *Given two quantum states ψ and ϕ , then for any quantum measurement it is*

$$\left| \Pr[\text{outcome is 1 given } \psi] - \Pr[\text{outcome is 1 given } \phi] \right| \leq \|\psi - \phi\|.$$

With this lemma we get:

$$\begin{aligned}
\Delta &:= \Pr[A^{f_x}() \rightarrow 1 : x \overset{\$}{\leftarrow} \{0, 1\}^n] - \Pr[A^{f_0}() \rightarrow 1] \\
&= \sum_{x \in \{0, 1\}^n} 2^{-n} \left(\Pr[A^{f_x}() \rightarrow 1] - \Pr[A^{f_0}() \rightarrow 1] \right) \\
&\leq \sum_{x \in \{0, 1\}^n} 2^{-n} \sqrt{\|\psi_q^x - \phi_q\|^2} \quad // \text{ Quantum inequality above} \\
&\leq \sqrt{\sum_{x \in \{0, 1\}^n} 2^{-n} \|\psi_q^x - \phi_q\|^2} \quad // \text{ Jensen's inequality} \\
&= \sqrt{D}.
\end{aligned}$$

So it is $\Delta \leq \sqrt{D}$ and since we assumed $\Delta \geq \text{const}$ it is:

$$D \geq \Delta^2 \geq \text{const}.$$

Hence, if we show $D \leq \frac{4q^2}{2^{-n}}$ then it follows $\frac{q^2}{2^{-n}} \geq \text{const}$ and therefore $q \in \Omega(2^{\frac{n}{2}})$.

We prove this via induction on the number of queries k , defining:

$$D_k := \sum_{x \in \{0, 1\}^n} 2^{-n} \|\psi_k^x - \phi_k\|^2.$$

Note that $D = D_q$. To prove $D \leq \frac{4q^2}{2^{-n}}$, we will show $D_k \leq \frac{4k^2}{2^{-n}}$ by induction.

For $k = 0$ it is $\psi_k^x = \phi_k$, as you can see in the circuits. And therefore it is:

$$D_0 = 0 \leq \frac{4 \cdot 0^2}{2^{-n}} = 0.$$

Now we look at the induction step $k \rightarrow k + 1$:

We have $D_k \leq \frac{4k^2}{2^{-n}}$ and want to show $D_{k+1} \leq \frac{4(k+1)^2}{2^{-n}}$:

$$\begin{aligned}
2^n D_{k+1} &= \sum_{x \in \{0,1\}^n} \left\| \psi_{k+1}^x - \phi_{k+1} \right\|^2 \\
&= \sum_{x \in \{0,1\}^n} \left\| U_{k+1} (V_f \otimes I_2) \psi_k^x - U_{k+1} \phi_k \right\|^2 \quad // \text{ by looking at the circuits} \\
&= \sum_{x \in \{0,1\}^n} \left\| U_{k+1} \left((V_f \otimes I_2) \psi_k^x - \phi_k \right) \right\|^2 \\
&= \sum_{x \in \{0,1\}^n} \left\| (V_f \otimes I_2) \psi_k^x - \phi_k \right\|^2 \quad // \text{ because unitaries preserve norms} \\
&= \sum_{x \in \{0,1\}^n} \left\| (V_f \otimes I_2) (\psi_k^x - \phi_k) + \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\|^2 \\
&\leq \sum_{x \in \{0,1\}^n} \left\| (V_f \otimes I_2) (\psi_k^x - \phi_k) \right\|^2 + 2 \left\| (V_f \otimes I_2) (\psi_k^x - \phi_k) \right\| \cdot \left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\| \\
&\quad + \left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\|^2 \\
&= \sum_{x \in \{0,1\}^n} \left\| \psi_k^x - \phi_k \right\|^2 + 2 \left\| \psi_k^x - \phi_k \right\| \cdot \left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\| \\
&\quad + \left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\|^2 \quad // \text{ because unitaries preserve norms} \\
&= 2^n D_k + 2 \sum_{x \in \{0,1\}^n} \left\| \psi_k^x - \phi_k \right\| \cdot \left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\| + \sum_{x \in \{0,1\}^n} \left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\|^2 \\
&\stackrel{(*)}{=} 2^n D_k + 4 \sum_{x \in \{0,1\}^n} \left\| \psi_k^x - \phi_k \right\| \sqrt{\Pr[M \rightarrow x]} + 4 \underbrace{\sum_{x \in \{0,1\}^n} \Pr[M \rightarrow x]}_{=1 \left(\sum \text{ over all measurement outcomes} \right)} \\
&\leq 2^n D_k + 4 \sqrt{\sum_{x \in \{0,1\}^n} \left\| \psi_k^x - \phi_k \right\|^2 \cdot \sum_{x \in \{0,1\}^n} \Pr[M \rightarrow x]} + 4 \quad // \text{ Cauchy-Schwarz-inequality} \\
&= 2^n D_k + 4 \sqrt{2^n D_k \cdot 1} + 4 \\
&\leq 4k^2 + 4\sqrt{4k^2} + 4 \quad // \text{ induction hypothesis} \\
&= 4k^2 + 8k + 4 \\
&= 4(k+1)^2.
\end{aligned}$$

So it is $D_{k+1} \leq \frac{4(k+1)^2}{2^{-n}}$. Note that $(*)$ holds since when we consider $\left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\|^2$

it is:

$$\begin{aligned} (V_f - I_{2^n})|z\rangle &= V_f|z\rangle - |z\rangle \\ &= \begin{cases} -|z\rangle - |z\rangle = -2|z\rangle & (z = x) \\ |z\rangle - |z\rangle = 0 & (z \neq x) \end{cases} \end{aligned}$$

from which follows that $(V_f - I_{2^n}) = 2P_x$ where P_x is the projection outcome onto $|x\rangle$. Hence:

$$\begin{aligned} &\left\| \left((V_f \otimes I_2) - I_{2^{n+1}} \right) \phi_k \right\|^2 \\ &= \left\| \left((V_f - I_{2^n}) \otimes I_2 \right) \phi_k \right\|^2 \\ &= 4 \left\| \underbrace{(P_x \otimes I_{2^n}) \phi_k}_{\substack{\text{non-normalized} \\ \text{post-measurement-state} \\ \text{when measuring } |x\rangle}} \right\|^2 \\ &= 4 \Pr[M \rightarrow x] \end{aligned}$$

where M is the measurement of the X -register.

In conclusion, no quantum algorithm can solve SAT in less than $\Omega(2^{\frac{n}{2}})$ queries with an oracle. So SAT is quantum-hard relative to some oracle.

19.2 Quantum Merlin Arthur

The question arises, if there is a “quantum NP”?

For this we define the class QMA (Quantum Merlin Arthur) analogous to Definition 19.1:

Definition 19.2 (The complexity class QMA). A language L is in QMA if and only if there exists a quantum algorithm A running in polynomial time such that:
 If x is a YES-Instance, then a quantum state $|\psi\rangle$ exists such that $\Pr[A(x, |\psi\rangle) = 1] \geq \frac{2}{3}$,
 if x is a NO-Instance, then for all quantum states $|\psi\rangle$ is $\Pr[A(x, |\psi\rangle) = 1] \leq \frac{1}{3}$.

Thus, QMA plays a similar role in quantum complexity theory as NP in classical complexity theory. (And as MA in the complexity of randomized algorithms.)

There are known QMA-complete problems. One example is the “Local Hamiltonian problem”. For the definition and explanation of a Hamiltonian see Section 13.3.

Definition 19.3 (2-local Hamiltonian). A Hamiltonian H is 2-local if it can be written

as:

$$H = \sum_{i=1}^k H_i$$

where each H_i is a Hamiltonian that operates only on two qubits.

Example: 2-Local Hamiltonian problem

Given a 2-local Hamiltonian H (specified by the individual H_i), determine whether:

YES-Instance: The smallest eigenvector of H is $\leq \frac{1}{3}$,

NO-Instance: The smallest eigenvector of H is $\geq \frac{2}{3}$.

This problem is indeed QMA-complete (without proof):

Theorem 19.2 (2LA is QMA-complete). *The 2-Local Hamiltonian problem is QMA-complete.*