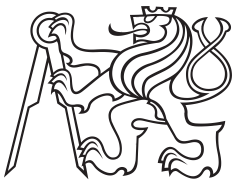
Bachelor Project



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Mathematics

Hidden subgroups and quantum algorithms

Matouš Pikous

Supervisor: Doc. RNDr. Jiří Velebil, Ph.D.
Field of study: Cybernetics and robotics
May 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Pikous Matouš** Personal ID number: **483593**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Hidden subgroups and quantum algorithms

Bachelor's thesis title in Czech:

Skryté podgrupy a kvantové algoritmy

Guidelines:

The theme of the bachelor project is covering the necessary basics for the bachelor thesis topic.
The expected output of the bachelor project is a written summary and its public defence at the department.
The topics of the project in more detail are:
(1) Mathematics of simple quantum systems, measurements.
(2) Q-bits, quantum gates and basic quantum circuits.
(3) Deutsch's algorithm and its HSP formulation.
(4) Towards more complex quantum algorithms: the general
Quantum Fourier Transform.

Bibliography / sources:

[1] M. Hirvensalo, Quantum computing, 2nd ed, Springer, 2004.
[2] R. Josza, Quantum factoring, discrete logarithms, and the hidden subgroup problem, IEEE Computing in Science & Engineering 3.2 (2001), 34--43.
[3] MA. Nielsen and I.L.Chuang, Quantum computation and quantum information, Cambridge University Press, 2011.
[4] P. Shor, Algorithms for quantum computation: discrete logarithms and factoring, Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE Comput. Soc. Press (1994), 124--134.

Name and workplace of bachelor's thesis supervisor:

doc. RNDr. Jiří Velebil, Ph.D., Department of Mathematics, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **14.09.2020** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until:
by the end of summer semester 2021/2022

doc. RNDr. Jiří Velebil, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

I would like to thank my supervisor Doc. RNDr. Jiří Velebil, Ph.D. for his supportive guidance not only with the thesis. It is him and my friend Martin Rektoris to whom I am grateful for not quitting my studies prematurely.

Declaration

I hereby declare that the thesis is my own work. I also declare that I cited all the sources in the bibliography in accordance with the Guidelines of ethical standards for preparation final theses No 1/2009.

Abstract

In this thesis we introduce the reader to the basic concepts in quantum computation from the mathematical perspective. We lay down the foundations of group theory and we define the hidden subgroup problem. The knowledge is then applied to Deutsch's algorithm and Shor's algorithm. This thesis is intended for graduate or undergraduate students with interest in mathematics but with no prior experience with quantum computation.

Keywords: quantum algorithm, hidden subgroup, Deutsch's algorithm, quantum computer

Supervisor: Doc. RNDr. Jiří Velebil, Ph.D.
Department of Mathematics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 4
Prague 6

Abstrakt

V této práci je čtenář uveden do základní problematiky kvantových výpočtů z pohledu matematiky. Jsou zde položeny základy teorie grup a je zde formulován problém skryté podgrupy. Tyto poznatky jsou pak aplikovány v Deutschově algoritmu a v Shorově algoritmu. Tato práce je určena pro studenty magisterských nebo bakalářských oborů, kteří mají zájem o matematiku, avšak nemají žádnou předchozí zkušenost s kvantovými výpočty.

Klíčová slova: kvantový algoritmus, skrytá podgrupa, Deutschův algoritmus, kvantový počítač

Překlad názvu: Skryté podgrupy a kvantové algoritmy

Contents

Introduction

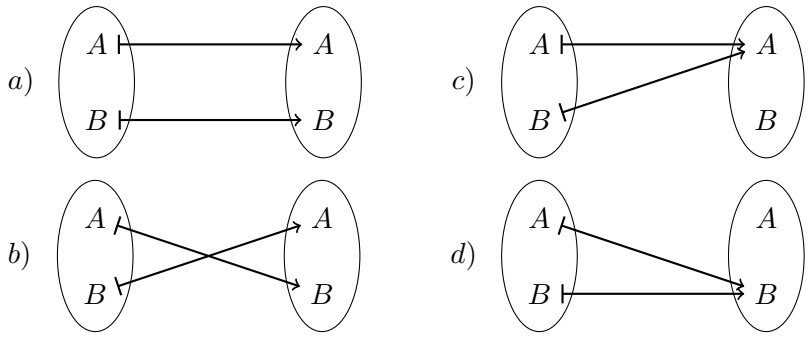
This thesis begins with two lovers Alice and her paramour Bob who are sending their love letters to each other. As in all stories there is the third vertex of the love triangle, the eavesdropper, Bob's spurn lover Eve. Eve is jealous and desperate to read Bob's and Alice's letters. Unfortunately for her, Alice and Bob encrypt their messages with sophisticated algorithms such as RSA where it is crucial to guess a decomposition of a number into two prime numbers. [1] Eve has no chance to break this code with standard computers efficiently, so she learns quantum physics and theory of quantum computation to construct a quantum computer, a delicate machine which will help her to read the two lovers' secrets in polynomial time.
In this text we will present the mathematical foundation for quantum computation. Nothing what follows is new, we only give overview of known facts. For more details, we refer the reader to the following publications. If you want to know more about quantum computation we recommend [18]. You can read about mathematics (especially linear algebra) behind quantum computation here [21]. If you are interested in quantum physics and you have no prior experience then the books [16], [5] and [19] should be helpful.

Motivation

To introduce you to the topic of quantum computation we will begin with an example where quantum computers outperform ordinary computers. Let us have a set with two elements A and B inside. We can think of four different

[1] You can read more about RSA in the books [20] or [15].

functions on the set namely "the do nothing" function, "the swap" function and two "merge" functions. We will name them properly in few lines. Firstly we notice difference between the functions. The first two give different outputs for different inputs while the merging functions do not. We will call these functions identity, swap and the rest two are constant functions. You can see them in the following picture.

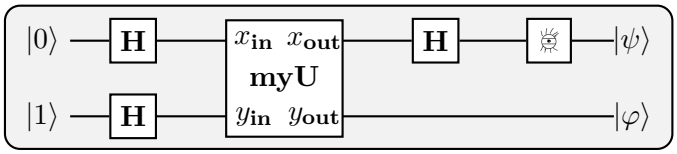


Now an adversary gives us an unknown function from $\{A, B\}$ to $\{A, B\}$, let us call it f, and our job is to determine whether it is a constant function or not. Ordinarily, we just try both inputs A and B and see the results. Standard computation of the problem could be interpreted as answering questions: "What is the result for input A? -It is B. -What is the result for input B? -It is B." It is often said, that quantum computers somehow miraculously can ask all queries at once, something like "What is the result of all inputs?". This simplification is misleading and does not give us any intuition. A better metaphor is posing a carefully chosen question on a fundamental property of the function, so the process looks like: "-How do the outputs of the function look like? -They are all constant." This attitude helps us to get the results faster than with the traditional approach as we can see in this example where we are able to determine what kind of function f is on a single inquiry. How do we ask such a question? Let us see!

Very informally quantum computation is an application of linear algebra with a slightly changed notation. Due to the reasons of physics, all operations on quantum computers are represented by unitary matrices and the result is then "measured". We will represent the inputs A and B by vectors of the canonical basis and just denote them in a somewhat silly form as $|0\rangle$ and $|1\rangle$.

We would love to know how the function f works for all combinations of inputs and still do only one computation. Since we are in the realm of linear algebra we will try to use a linear combination of inputs. For this, we

will need a unitary matrix, for now let us call it **H** (the Hadamard gate), which takes vector $|0\rangle$ and returns a linear combination of basis vectors and it should also be able to take vector $|1\rangle$ and return a linear combination of $|0\rangle$ and $|1\rangle$. There are plenty of matrices to choose from, but we would like to have a matrix which is its inverse. It may seem we are going to miraculously ask for both inputs after all, but here comes the carefully posed question. We will construct a special gate **myU**. We can imagine it takes two inputs $x_{\mathbf{in}}$ and $y_{\mathbf{in}}$ where $x_{\mathbf{in}} = \mathbf{H}|0\rangle$, $y_{\mathbf{in}} = \mathbf{H}|1\rangle$. At this point, we should draw a picture of what is happening.



Very informally, the **myU** gate applies f on x and then performs some still unknown process and returns two outputs. These outputs are expected to be also some combination of $|0\rangle$ and $|1\rangle$, but when Hadamard gate is applied to one of these a miracle happens and we can see $|0\rangle$ if the function f is constant and $|1\rangle$ if the function is not constant. And there we have it, the Deutsch's algorithm (see the original paper [11] or the books [18] [22]), the first quantum computer algorithm which outperforms standard computers. Of course I am just waving my hands in the air and I am sure none of my arguments have persuaded the reader of the correctness of this algorithm. For that we have to lay down the mathematical foundations first and after that, we will revisit this example in Chapter 2 and Chapter 4.

Synopsis

As the motivational example tries to show, quantum computers can resolve some problems much quicker compared to the ordinary computer with binary operations only. The theory for quantum computers is much richer and unfortunately for the reader it is more complicated. In this thesis we lay down the foundations for quantum computation. There are no new information presented in this thesis. There are plenty of almost perfect sources and great books to read. However, if the reader is a newcomer and does not have any previous experience with quantum computation, the common publications may be overwhelming. This thesis covers the essentials to understand quantum computers from the mathematical perspective and presents and rearranges the know-how in a compact way.

1. Firstly in **Chapter 1** we will remind ourselves about linear algebra in

complex spaces, we will postulate rules for quantum computation and we will introduce the gate notation.

2. Secondly in **Chapter 2** we present the Deutsch's algorithm and we apply knowledge from the previous chapter.

3. Then in **Chapter 3** we will focus on group theory, the Fourier transform on Abelian groups and the hidden subgroup problem.

4. Next in **Chapter 4** we will show path for quantum computers for a general hidden subgroup problem. The Shor's algorithm will be also presented in this chapter. Finally we will formulate the hidden subgroup problem for Deutsch's problem.

Chapter 1

Foundation of quantum computing

One of my teachers once said: "Quantum computation is like linear algebra but with strange notation." On the first sight the notation may feel over-complicated and uncomfortable to use but it will simplify our job and make equations more intuitive and readable. After all, the strange notation is the good notation. It is also necessary to introduce new notation for some other basic concepts so that it is coherent throughout the following chapters.

Disclaimer. We will assume that quantum computers will be finite in size and we will be able to run only finite codes, therefore we can simplify our mathematical apparatus and we will work with vectors in $\mathbb{C}^n$ and linear forms on $\mathbb{C}^n$. The side benefit is that the mathematics behind will be more reachable for the readers and the proofs will not drown into technicalities. We will sadly not deal with quantum mechanics and we will just borrow the name quantum state as a synonym for a vector (of unit length). You can read more about quantum physics in the book [16] or [5].

1.1 Basic definitions

We assume the reader has basic knowledge of linear algebra. Before we start with quantum computation we, however, need to unify the notation that we will use later on.

1.1.1 Notation (Complex conjugate) Suppose $z \in \mathbb{C}$, where $z = a + bi$

and $a, b \in \mathbb{R}$; $i = \sqrt{-1}$. We define $z^* = a - bi$ as the *complex conjugate* to z.

1.1.2 Definition (Inner product) Suppose L is a linear space (over $\mathbb{C}$). A function $L \times L \to \mathbb{C}$ denoted by $\langle -|- \rangle$ is called *inner product*, if it satisfies:

$$\langle x|y\rangle = \langle y|x\rangle^* \tag{1.1}$$
$$\langle x|\alpha y + \beta z\rangle = \alpha\langle x|y\rangle + \beta\langle x|z\rangle \tag{1.2}$$
$$\langle x|x\rangle \geq 0 \text{ and } \langle x|x\rangle = 0 \text{ if and only if } x = \vec{0} \tag{1.3}$$

for all $x, y, z \in L$ and $\alpha, \beta \in \mathbb{C}$. Note that the linearity is in the second argument.

1.1.3 Example (Standard inner product in $\mathbb{C}^n$) Let us have a function $\mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}$ defined by putting:

$$\langle x|y\rangle = \langle \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{pmatrix} | \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ y_n \end{pmatrix} \rangle = \sum_{i=1}^{n} {x_i}^* y_i \tag{1.4}$$

Without any effort we can show that this function is an inner product, which we will use in next chapters if not specified otherwise.

1.1.4 Remark Some linear spaces with inner product are Hilbert spaces. We will not deal with infinite dimensional spaces in this text and all we need is only that all finite dimensional spaces over $\mathbb{C}$ are Hilbert spaces, see [21].

1.1.5 Definition (Hermitean conjugate matrix) Consider vector spaces $\mathbb{C}^s$ equipped with inner product $\langle -|- \rangle_s$ and $\mathbb{C}^r$ with inner product $\langle -|- \rangle_r$. Having a linear mapping $\tau : \mathbb{C}^s \to \mathbb{C}^r$ represented by matrix $\mathbf{T}$, we say that $\mathbf{T}^\dagger$ is *Hermitean conjugate* matrix to $\mathbf{T}$ when the following:

$$\langle \mathbf{T}^\dagger x|y\rangle_s = \langle x|\mathbf{T}y\rangle_r \tag{1.5}$$

is satisfied for all $x \in \mathbb{C}^r$ and all $y \in \mathbb{C}^s$.

1.1.6 Lemma (Elements of conjugate matrix) *Suppose $\tau : \mathbb{C}^s \to \mathbb{C}^r$ is given by matrix $\mathbf{T}$ with elements (t_{ij}) then matrix $\mathbf{T}^\dagger$ has elements (t^*_{ji}).*

Proof. The equation (1.5) holds for any $x \in \mathbb{C}^r$ and $y \in \mathbb{C}^s$, in particular for basis vectors $e_n \in \mathbb{C}^r$ and $f_m \in \mathbb{C}^s$ the equation

$$\langle \mathbf{T}^\dagger e_n|f_m\rangle_s = \langle e_n|\mathbf{T}f_m\rangle_r \tag{1.6}$$

holds as well. By definition of the inner product (1.1) we can rewrite (1.6) as

$$\langle \mathbf{T}^\dagger e_n|f_m\rangle_s = \langle f_m|\mathbf{T}^\dagger e_n\rangle_s^*, \tag{1.7}$$

which says how elements (θ_{nm}) of $\mathbf{T}^\dagger$ look like. The above allows us to write all expressions in the following equation:

$$\begin{aligned}(\theta^*_{nm}) &= \langle f_m|\mathbf{T}^\dagger e_n\rangle^*_s && (1.8)\\ &= \langle \mathbf{T}^\dagger e_n|f_m\rangle_s && (1.9)\\ &= \langle e_n|\mathbf{T}f_m\rangle_r && (1.10)\\ &= (t_{mn}). && (1.11)\end{aligned}$$

Therefore for any matrix $\mathbf{T}^\dagger$ Hermitean conjugate to a matrix $\mathbf{T}$ the equation

$$(\theta_{nm}) = (t^*_{mn}) \tag{1.12}$$

holds.

■

1.1.7 Lemma (Rules for conjugate matrices) *For any complex matrices* $\mathbf{T}$, $\mathbf{S}$ *and* $\alpha \in \mathbb{C}$ *the equations*

$$\begin{aligned}(\mathbf{T}+\mathbf{S})^\dagger &= \mathbf{T}^\dagger + \mathbf{S}^\dagger && (1.13)\\ (\alpha\mathbf{T})^\dagger &= \alpha^*\mathbf{T}^\dagger && (1.14)\\ (\mathbf{TS})^\dagger &= \mathbf{S}^\dagger\mathbf{T}^\dagger && (1.15)\\ (\mathbf{T}^\dagger)^\dagger &= \mathbf{T} && (1.16)\end{aligned}$$

hold.

Proof. The proof is a simple consequence of Lemma 1.1.6 and definition of Hermitean conjugate matrix 1.1.5 and it is left as an exercise. ■

Among Hermitean conjugate matrices there are some with special properties that deserve their own name.

1.1.8 Definition (Self-adjoint matrix) We call a matrix $\mathbf{T}$ *self-adjoint* if

$$\mathbf{T}^\dagger = \mathbf{T}. \tag{1.17}$$

holds.

1.1.9 Definition (Positive matrix) Having a Hermitean adjoint matrix $\mathbf{T}$ we say it is *positive*, if the equation

$$\langle \mathbf{T}x|x\rangle \geq 0 \tag{1.18}$$

holds for all $x \in \mathbb{C}^n$.

1.1.10 Definition (Unitary matrix) We call a Hermitean adjoint matrix $\mathbf{T}$ *unitary* if the equation

$$\mathbf{T}^\dagger\mathbf{T} = \mathbf{T}\mathbf{T}^\dagger = \mathbf{E} \tag{1.19}$$

holds, where the matrix $\mathbf{E}$ is the identity matrix.

There is more to read about adjoint matrices in [2].

We have dealt mainly with the foundations of linear algebra for quantum computation, but the world is more complicated. When we firstly saw the inner product during lectures of linear algebra, those were times when we worked in the spaces of $\mathbb{R}^n$, we could say that it was linear in both arguments. The postulate (1.2) gives linearity in the second argument and thanks to the field of reals the postulate (1.1) ensures linearity in the first argument. We can easily prove that the inner product is not linear, meanwhile we can wrongly say that it is somehow "twice-linear". That is a shame because we do not know how to easily join two nonlinear functions and especially with multiple arguments which it is exactly what we need in computation. Maybe there is a way how to correct these nonlinear mappings to linear functions as we will see in this section. Firstly we just define the special "twice-linear" mapping and then we will continue with the definition of tensor product.

1.1.11 Definition (Multilinear mapping) Let us have linear spaces L_1, L_2, ... L_n, where $n \in \mathbb{N}$ and a linear space V all over the same field $\mathbb{F}$. We call a function

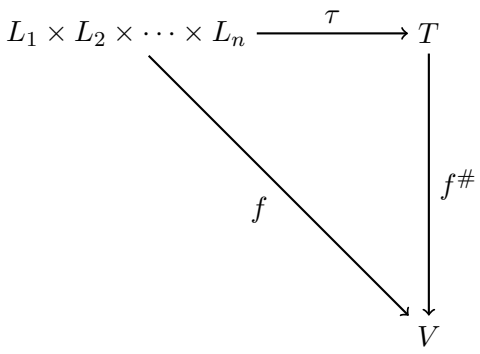
$$\begin{aligned} f : L_1 \times L_2 \times \cdots \times L_n &\to V \\ (l_1, l_2, \dots l_n) &\mapsto f(l_1, l_2, \dots l_n) \end{aligned}$$

multilinear if the following condition holds. If we choose any $l_i = \alpha x + y$ where $i = 1, 2, ...n$ and we fix all l_j, where $j \neq i$ it holds that

$$\begin{aligned} f(l_1, l_2, \dots l_i, \dots l_n) &= f(l_1, l_2, \dots \alpha x + y, \dots l_n) \\ &= \alpha f(l_1, l_2, \dots, x, \dots l_n) + f(l_1, l_2, \dots y, \dots l_n) \end{aligned} \tag{1.20}$$

In particular if the number of input arguments is n we will call the mapping n-*linear*. Special cases are 1-linear mapping called just *linear* and 2-linear mapping called *bilinear*.

The slogan is *"A multilinear mapping is linear in every argument separately"*. Now, when we defined bilinear mappings properly we can focus on the tensor product. We shall begin with motivational picture:



In the picture f denotes a multilinear function. The letter τ represents a multilinear mapping which we do not know yet. The letter T denotes an unknown linear space and $f^\#$ is a linear function such that $f^\#\tau = f$. It would be lovely to have such a mapping τ and space T and even lovelier if the function $f^\#$ was unique. If it was so, we could "pretend" that multilinear function "behave" like linear functions. That is exactly the way how we will define the tensor product.

1.1.12 Definition (Tensor product) Let us have linear spaces L_1, L_2, ... L_n (where $n \in \mathbb{N}$) over the same field $\mathbb{F}$. We say that a linear space T together with multilinear mapping

$$\tau : L_1 \times L_2 \times \cdots \times L_n \to T \tag{1.21}$$

form a *tensor product* of spaces L_1, L_2, $\ldots L_n$ if they have the following property: For all linear spaces V and any multilinear mapping

$$f : L_1 \times L_2 \times \cdots \times L_n \to V \tag{1.22}$$

there exists exactly one linear mapping

$$f^\# : T \to V \tag{1.23}$$

such that

$$f^\# \cdot \tau = f. \tag{1.24}$$

We will write $L_1 \otimes L_2 \otimes \cdots \otimes L_n$ instead of T. In some publications the function τ is represented by symbol $\otimes$ used in the infix notation to mimic the symbol for multiplication. See e.g., [12] or [21] (with different notation).

The above definition defines a tensor product but it is not clear that a tensor product of two (or more) linear spaces actually exists. We show in the following example that it is indeed the case for spaces that interest us. Moreover, the tensor product has quite an easy description.

1.1.13 Example In quantum computation we work in finite spaces over complex numbers hence we can use some simplifications. This example is inspired by Kronecker's product [17] which is in some cases interchangeable for the tensor product as defined for example in [16].
Let us have a basis $(x_1, x_2, \ldots, x_n)$ of space $\mathbb{C}^n$ and let $(y_1, y_2, \ldots, y_m)$ be a basis of space $\mathbb{C}^m$. The only possible functions in quantum computation are of a kind

$$f : \mathbb{C}^n \times \mathbb{C}^m \to \mathbb{C}^k, \text{ where } k \in \mathbb{N}. \tag{1.25}$$

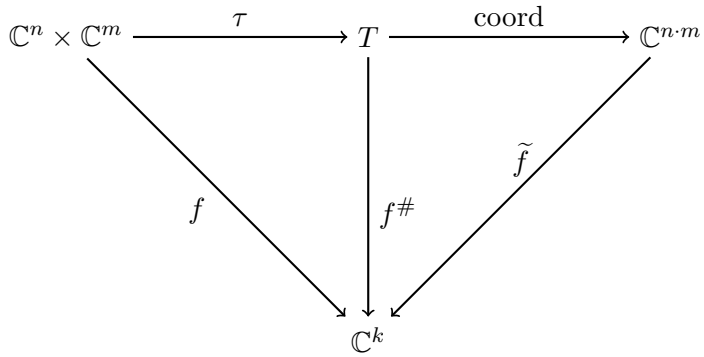
We define the space T as linear space of matrices of with n rows and m columns and the function τ as following:

$$\tau(x_i, y_j) = x_i \cdot y_j^T \text{ where } i = 1, 2, ..n \text{ and } j = 1, 2, \ldots m. \tag{1.26}$$

For every vector $v = \sum_i^n \alpha_i x_i$ and $w = \sum_j^m \beta_j y_j$ we define

$$\tau(v, w) = \sum_i^n \sum_j^m \alpha_i \beta_j x_i y_j^T. \tag{1.27}$$

It can be easily proven that τ is indeed a multilinear mapping and also that the space T is a linear space. It remains to be shown that there exists only one $f^\#$ for every function f. To do so, we will use coordinate system of the space T. Let us draw a picture to clarify what we aim to do.



The coordinate system of the space T is $\mathbb{C}^{n \cdot m}$, because the matrices have $n \cdot m$ independent parameters. A given function f tells us its value on the basis vectors. It is then simple to decide what the function

$$\widetilde{f} : \mathbb{C}^{n \cdot m} \to \mathbb{C}^k \tag{1.28}$$

looks like. We define it in such a way that, if we apply $\widetilde{f}$ to a vector $e_{i,j} = \text{coord}(\tau(x_i, y_j))$ of the canonical basis, the result $\widetilde{f}(e_j)$ is the same as $f(x_i, y_j)$. There is just one definition of the function $\widetilde{f}$, which is linear and finding coordinates with respect to chosen basis is also linear and unique, therefore when these mappings are composed, the result is also unique and linear. The result is nothing more than our function $f^\#$.
When we use tensor product in quantum computation, we can imagine that

this process happens in the background. This example also shows us that the matrices we use to describe quantum computation can have too many parameters and it is not sound to write them down. We will improve our notation in 1.4.

1.2 Notation for quantum computation

1.2.1 Notation (Ket) A vector $x \in \mathbb{C}^n$ will be denoted by $|x\rangle$ (pronounced: *ket* x). We will identify $|x\rangle$ also with a linear map $\mathbb{C} \to \mathbb{C}^n$ which maps $1 \mapsto |x\rangle$. If $||x|| = 1$ will call x a *quantum state*. We will often use Greek letters ψ or φ to denote qubits. [1]

1.2.2 Notation (Bra) The covector corresponding to the vector $y \in \mathbb{C}^n$ will be denoted by $\langle y|$ (pronounced: *bra* y). For example $\langle y|$ is vector y which is transposed and all elements are complex conjugated. We also note that $\langle y|$ corresponds to linear map $\mathbb{C}^n \to \mathbb{C}$ sending $|e_i\rangle$ to $\langle y|e_i\rangle$.

1.2.3 Notation (Braket) Whenever vectors $|y\rangle$ and $|x\rangle$ are from the same space we simplify $\langle y|\ |x\rangle$ to $\langle y|x\rangle$ (pronounced: *braket* y x). We can see that $\langle y|$ evaluated at $|x\rangle$ yields the standard inner product $\langle y|x\rangle$ on $\mathbb{C}^n$. [2]

1.2.4 Notation (Computational basis and qubits) The canonical basis of $\mathbb{C}^2$ will be denoted by:

$$\begin{pmatrix}1\\0\end{pmatrix} = |0\rangle \quad \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle \tag{1.29}$$

We will call any quantum state of $\mathbb{C}^2$ *quantum bit* or *qubit*.

The canonical basis for $\mathbb{C}^4$ will be denoted by:

$$\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = |00\rangle \quad \begin{pmatrix}0\\1\\0\\0\end{pmatrix} = |01\rangle \quad \begin{pmatrix}0\\0\\1\\0\end{pmatrix} = |10\rangle \quad \begin{pmatrix}0\\0\\0\\1\end{pmatrix} = |11\rangle \tag{1.30}$$

The notation for canonical basis of spaces of the form C^{2^n} is defined inductively. These basis will be called *computational basis*. This representation should feel similar to binary representation of numbers and it is so on purpose.

[1] More about qubits in 1.2.4.
[2] See Example 1.5.2.

1.2.5 Example (Two useful matrices) Unitary operators are crucial in quantum computing so we shall see some of them before we dive deeper. The following matrix $\mathbf{X}$ is often called "**not**" in quantum computation. The values of $\mathbf{X}$ at $|0\rangle$ and $|1\rangle$ are $\mathbf{X}|0\rangle = |1\rangle$ and $\mathbf{X}|1\rangle = |0\rangle$. This is similar with how not operation works in standard computing.

$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.31}$$

The following matrix $\mathbf{H}$ is called *Hadamard matrix*.

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.32}$$

The matrix $\mathbf{H}$ is rather important; more about it later, see e.g. 1.4.2 below.

1.3 Postulates of quantum computing

The quantum computation can be viewed from the perspective of a physicist and could be based on observations of the micro-world and careful measurements. We leave the dirty job to physicists and we will play a purely intellectual game in what follows. Thus, we only postulate the rules of quantum computation, see e.g., in [13], [19] and [18].

1.3.1 Postulate (The space and the state) The quantum computation is performed in a finitely-dimensional Hilbert space over $\mathbb{C}$ (in physics it could be interpreted as an isolated system). Any state of computation is represented by quantum state $|\psi\rangle$ and it is true that $\langle\psi|\psi\rangle = 1$.

1.3.2 Postulate (Joining of spaces) Joining two systems is represented by a *tensor product* 1.1.12 of the corresponding Hilbert spaces.

1.3.3 Postulate (Change of states) Any change of quantum state, where $|\psi_1\rangle \mapsto |\psi_2\rangle$, can be performed by *unitary matrix* $\mathbf{U}$ so that $\mathbf{U}|\psi_1\rangle = |\psi_2\rangle$. The matrix $\mathbf{U}$ will be called a *quantum gate*.

1.3.4 Postulate (The measurement) A *measurement* is a system $\mathbf{M}_m$, where $\mathbf{M}$ is a linear function and m is result we get (so called *measurement outcome*). The following condition (called *completeness*)

$$\sum_m \mathbf{M}_m^\dagger \mathbf{M}_m = \mathbf{E} \tag{1.33}$$

must hold, where the summation is performed over all possible outcomes and the matrix $\mathbf{E}$ is identity matrix. The probability that we get m when we apply $\mathbf{M}_m$ on state $|\psi\rangle$ is given by:

$$p(m) = \langle\psi|\mathbf{M}_m^\dagger \mathbf{M}_m|\psi\rangle = \langle\mathbf{M}_m\psi|\mathbf{M}_m\psi\rangle \tag{1.34}$$

1.3.5 Remark We can see that $p(m) \geq 0$ but we have to show that the sum of probabilities for all possible measurements is equal to one. With one hundred percent certainty we have to measure at least something.

$$\sum_m \langle\psi|\mathbf{M}_m^\dagger \mathbf{M}_m|\psi\rangle = \langle\psi|\sum_m \mathbf{M}_m^\dagger \mathbf{M}_m|\psi\rangle = \langle\psi|\mathbf{E}|\psi\rangle = \langle\psi|\psi\rangle = 1 \tag{1.35}$$

where the last equality holds since $|\psi\rangle$ is a unit vector.

1.3.6 Remark The new state $|\psi'\rangle$, which we obtain after measurement, is given by the following equation.

$$|\psi'\rangle = \frac{1}{\sqrt{p(m)}}\mathbf{M}_m|\psi\rangle \tag{1.36}$$

The fraction is just a normalization factor because $|\psi'\rangle$ must be also a unit vector.

1.3.7 Remark Now we will focus on special measurement called *projective measurement*. Let $\mathbf{M} \in \mathbb{C}^{n\times n}$ be self-adjoint matrix (it is sometimes called an *observable matrix*) and $|\varphi_1\rangle, |\varphi_2\rangle, \ldots |\varphi_n\rangle$ normalized eigenvectors of $\mathbf{M}$ with corresponding eigenvalues $m_1, m_2, \ldots m_n$ then we can rewrite $\mathbf{M}$ as follows

$$\mathbf{M} = \sum_{i=1}^{n} m_i |\varphi_i\rangle\langle\varphi_i| \tag{1.37}$$

That should not surprise us because $|\varphi_i\rangle\langle\varphi_i|$ is in fact the projection[3] on the span of the eigenvector $|\varphi_i\rangle$. We should also add that $|\varphi_i\rangle\langle\varphi_i| = \mathbf{M}_{m_i}$ and that the following two equations

$$\mathbf{M}_{m_i}^\dagger = \mathbf{M}_{m_i} \tag{1.38}$$

$$\sum_{i=1}^{n} \mathbf{M}_{m_i}^\dagger \mathbf{M}_{m_i} = \mathbf{E} \tag{1.39}$$

hold.

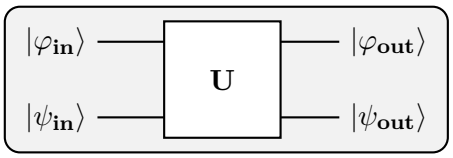
1.4 Quantum gates

We use unitary matrices to manipulate the states, but we realize that when we want to describe higher-dimensional problems the matrices grow in size exponentially and they do clutter several pages. That would be at least cumbersome to read and difficult to work with. We will take inspiration from ordinary computers where instead of long formulas in Boolean algebra we

[3] See Example 1.5.2.

use logical gates which are much more readable. We will represent unitary matrices with quantum gates. The quantum gates are explained in greater detail in [4] or [22], where the gates are also shown with their corresponding matrices.

1.4.1 Notation (Generic quantum gate) In the picture below there is shown what a quantum gate looks like for a generic unitary matrix **U**. The wires on the left-hand side are the input wires, where it takes (in this case two) input vectors. On the right-hand side there are the output wires. Since **U** is a unitary matrix, the number of output wires must be the same us the number of input wires.



Now you may be posing a question to yourself: how can a matrix, a linear function, have two or more inputs? To explain this peculiarity we should remind ourselves of the second postulate of quantum computing 1.3.2. Let us name the input vectors $|\psi_{\mathbf{in}}\rangle, |\varphi_{\mathbf{in}}\rangle \in \mathbb{C}^n$. Firstly we will form the tensor product

$$|\psi_{\mathbf{in}}\rangle \otimes |\varphi_{\mathbf{in}}\rangle = |\rho\rangle \tag{1.40}$$

which is isomorphic to $|\rho'\rangle$ in $\mathbb{C}^{n^2}$. Finally we apply the unitary matrix **U** hidden in the gate to the vector $|\rho'\rangle$.

1.4.2 Notation (Not gate and Hadamard gate) Let us remind ourselves of the "**not**" matrix and the Hadamard matrix from example 1.2.5 before the corresponding gates will be presented.

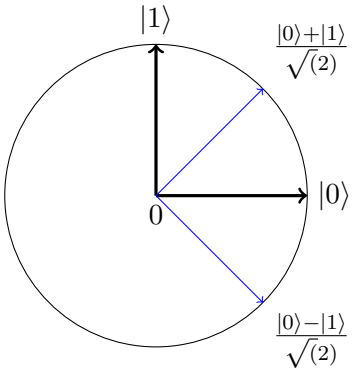
$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.41}$$

We have spoken about the "**not**" matrix and how it behaves in example 1.2.5. Hadamard matrix is still unknown to us and we only know that it will be important. If we want to get information about a linear function we should try to apply it on the basis vectors. So let us have the computational basis $(|0\rangle, |1\rangle)$ and apply the matrix **H**.

$$\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{1.42}$$

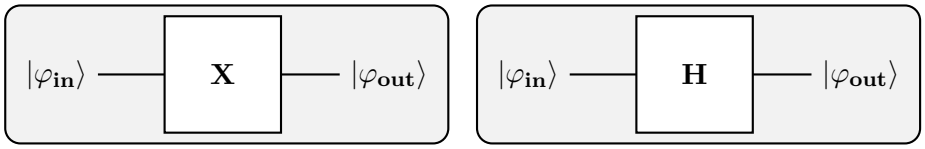
$$\mathbf{H}|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \tag{1.43}$$

If we cannot imagine how the gate behaves we can look at the picture below.



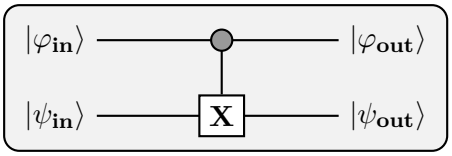
When applied on the computational basis, the result is a superposition of both states. That is something unheard of in classical computers and one cannot find a classic gate for such a operation. But there is more to the **H** gate and it has some specific properties. Notice that $\mathbf{H}^\dagger\mathbf{H} = \mathbf{E} = \mathbf{H}\mathbf{H}^\dagger$ and also that $\mathbf{H} = \mathbf{H}^\dagger$ therefore $\mathbf{H}^{-1} = \mathbf{H}$.

If we look back on the picture we can also see that if measure the result of $\mathbf{H}|0\rangle$ we will measure state $|0\rangle$ with fifty percent probability and we could measure state $|1\rangle$ with the same probability. Here are the gates for "Not" and Hadamard matrix.



Another approach to write the gates is to use Einstein's summation convention. This convention is used often in physics but we will stick with our gate representation.

1.4.3 Notation (Controlled not gate) The last gate I want to introduce is called *controlled not* gate which has an abbreviation "*C-not*". As the name suggests the action on the qubit on the second wire is determined by the qubit on the first wire. Because this kind of behavior is special we will leave the black box gate notation and denote it with its own symbol which you can see on the picture below.

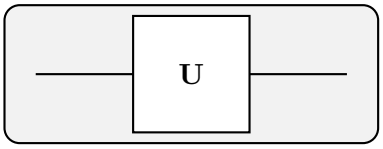


In this case the best way to understand the gate is by using something like a truth table for the input basis vectors. Let us name the inputs $|\varphi\rangle$ and $|\psi\rangle$. The output is given by the table:

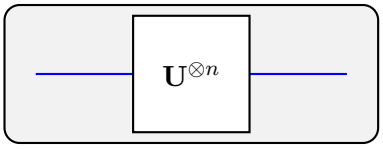
input		output	
$\|\varphi_{\mathbf{in}}\rangle$	$\|\psi_{\mathbf{in}}\rangle$	$\|\varphi_{\mathbf{out}}\rangle$	$\|\psi_{\mathbf{out}}\rangle$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

From the truth table we can see another important fact: The **C-not** gate behaves like "**xor**" on the second output, i. e. $|\psi_{out}\rangle = |\varphi_{in}\rangle \oplus |\psi_{in}\rangle$.

1.4.4 Notation (Parallel gates) If we want to apply a one wire gate

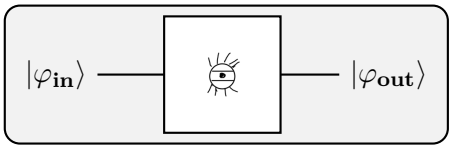


on n different wires, we denote it by



The exponent reminds us of the postulate of joining spaces 1.3.2 we apply here. We will distinguish by colour the wires representing spaces with different dimension.

1.4.5 Notation (The measurement gate) We also have to update our notation for the measurement. In some sources we can see a gate with a clock face on it but we will denote the measurement gate with an eye on the gate. The gate takes an input on the left and gives the result on the right.



Now we have prepared all necessary tools for quantum computation and we are ready to see the Deutsch's algorithm from introduction again.

Before we do it in Chapter 2 below, we conclude with some worked out examples.

1.5 Examples

1.5.1 Example (How to work with a vector) Let us consider a state $\psi \in \mathbb{C}^4$ and linear transformations given by matrices $\mathbf{A} \in \mathbb{C}^{4\times 4}$, $\mathbf{B} \in \mathbb{C}^{4\times 2}$. Firstly we can see that it is possible to "smuggle" the matrix $\mathbf{A}$ into ket

$$\mathbf{A}\psi = \mathbf{A}|\psi\rangle = |\mathbf{A}\psi\rangle \tag{1.44}$$

which seems obvious. The word "smuggle" is used intentionally because we can use this principle to store matrices and move them throughout the computation and release them if necessary. We should be always careful about the dimensionality of the ket. If we do the same with matrix $\mathbf{B}$ and denote the dimensionality of the ket by subscript we get

$$\mathbf{B}\psi = \mathbf{B}|\psi\rangle_4 = |\mathbf{B}\psi\rangle_2 \tag{1.45}$$

We are able to put matrices into bras as well but with a tiny change:

$$\langle\psi|\,\mathbf{A} = \left\langle \mathbf{A}^\dagger\psi\right| \tag{1.46}$$

1.5.2 Example (Beauty of bra-kets) Enjoying our notation we shall write a silly thing and see what it means, if it means anything at all.

$$|0\rangle\langle 0| = \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}^T = \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} \tag{1.47}$$

It is a projection matrix. Let us see how it works with a vector $\psi \in \mathbb{C}^2$

$$\psi = |\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.48}$$

$$|0\rangle\langle 0|\psi\rangle = \alpha|0\rangle\underbrace{\langle 0|0\rangle}_{=1} + \beta|0\rangle\underbrace{\langle 0|1\rangle}_{=0} = \alpha|0\rangle \tag{1.49}$$

$$|1\rangle\langle 1|\psi\rangle = \alpha|0\rangle\underbrace{\langle 1|1\rangle}_{=0} + \beta|1\rangle\underbrace{\langle 1|0\rangle}_{=1} = \beta|1\rangle \tag{1.50}$$

$$\psi = (|0\rangle\langle 0| + |1\rangle\langle 1|)\psi = \mathbf{E}\psi \tag{1.51}$$

We have just seen something unexpected yet quite ordinary at the same time. In the expression $|0\rangle\langle 0|\psi\rangle$ we evaluated $|0\rangle\langle 0|$ first and applied it on the $|\psi\rangle$. How about writing bra-ket together and evaluating it first?

$$\langle 0||0\rangle = \langle 0|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}^T\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\end{pmatrix} \tag{1.52}$$

It gives us the value of the inner product, i. e., a scalar.

1.5.3 Example (Rotation and reflection) Although these are simple examples, I wanted to show them because they are core of all other unitary matrices. Every time when you are presented an unitary operator you can imagine it as some kind of a rotation combined with reflecting around some chosen axis. These are also the simplest unitary matrices we can think of.

$$\mathbf{R}_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \qquad \mathbf{M}_x = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.53}$$

1.5.4 Example (Measurement) Imagine we have a a vector space $\mathbb{C}^2$ with computational basis $\{|0\rangle, |1\rangle\}$ and a vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Just for the sake of simplicity let us set $\alpha = \frac{\sqrt{3}}{2}$ and $\beta = \frac{1}{2}$. One can easily check that $|\psi\rangle$ is a unitary vector. It is in a superposition of two states of computational basis so let us see what happens if we measure this vector in the basis. Firstly let us write the projection matrices and than apply them to the vector $|\psi\rangle$.

$$\mathbf{M}_{|0\rangle} = |0\rangle\langle 0| \qquad \mathbf{M}_{|1\rangle} = |1\rangle\langle 1| \tag{1.54}$$

$$\mathbf{M}_{|0\rangle}|\psi\rangle = |0\rangle\langle 0|\psi\rangle = |0\rangle\langle 0|\left(\alpha|0\rangle + \beta|1\rangle\right) = \alpha|0\rangle \tag{1.55}$$

$$\mathbf{M}_{|1\rangle}|\psi\rangle = |1\rangle\langle 1|\psi\rangle = |1\rangle\langle 1|\left(\alpha|0\rangle + \beta|1\rangle\right) = \beta|1\rangle \tag{1.56}$$

Now, when we know how the vector $|\psi\rangle$ behaves when the projection matrices are applied we can calculate the probability of measuring $|0\rangle$ or $|1\rangle$.

$$\begin{aligned} p(|0\rangle) &= \langle\psi|\mathbf{M}^\dagger_{|0\rangle}\mathbf{M}_{|0\rangle}|\psi\rangle = \langle\psi|\mathbf{M}_{|0\rangle}\mathbf{M}_{|0\rangle}|\psi\rangle = \\ &\langle\psi|\mathbf{M}_{|0\rangle}|\psi\rangle = \alpha\langle\psi|0\rangle = \alpha\alpha^*\langle 0|0\rangle = \alpha\alpha^* \end{aligned} \tag{1.57}$$

$$\begin{aligned} p(|1\rangle) &= \langle\psi|\mathbf{M}^\dagger_{|1\rangle}\mathbf{M}_{|1\rangle}|\psi\rangle = \langle\psi|\mathbf{M}_{|1\rangle}\mathbf{M}_{|1\rangle}|\psi\rangle = \\ &\langle\psi|\mathbf{M}_{|1\rangle}|\psi\rangle = \beta\langle\psi|1\rangle = \beta\beta^*\langle 1|1\rangle = \beta\beta^* \end{aligned} \tag{1.58}$$

Let us substitute for α and β so

$$p(|0\rangle) = \alpha\alpha^* = \frac{\sqrt{3}}{2}\frac{\sqrt{3}}{2} = \frac{3}{4} \tag{1.59}$$

$$p(|1\rangle) = \beta\beta^* = \frac{1}{2}\frac{1}{2} = \frac{1}{4} \tag{1.60}$$

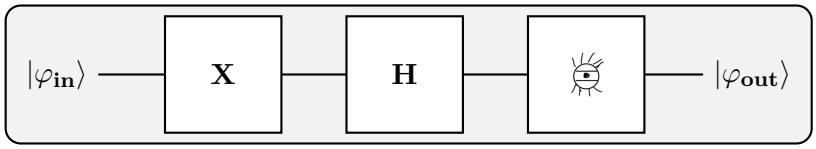
It is also worth to note that $\frac{3}{4} + \frac{1}{4} = 1$ so everything ended up as we expected. You may already see how the vector $|\psi_1'\rangle$ or $|\psi_2'\rangle$ will look like but we should do it properly.

$$|\psi_1'\rangle = \frac{1}{\sqrt{p(|0\rangle)}}\mathbf{M}_{|0\rangle}|\psi\rangle = \frac{1}{\sqrt{\frac{3}{4}}}\frac{\sqrt{3}}{2}|0\rangle = |0\rangle \tag{1.61}$$

$$|\psi_2'\rangle = \frac{1}{\sqrt{p(|1\rangle)}}\mathbf{M}_{|1\rangle}|\psi\rangle = \frac{1}{\sqrt{\frac{1}{4}}}\frac{1}{2}|1\rangle = |1\rangle \tag{1.62}$$

The conclusion is that when we measure vector $|\psi\rangle$ with probability $\frac{3}{4}$ we will measure $|0\rangle$ and with probability $\frac{1}{4}$ we will measure vector $|1\rangle$. It was obvious how it had to end up but it was fun, wasn't it?

1.5.5 Example (Quantum program) We know the fundamental blocks, the quantum gates including the measurement gate. Any quantum program will be constructed as a combination of these. Let us present our first quantum program:



We can obviously rewrite it back to the matrices, but the beauty is we do not have to. Let us say the input is $|1\rangle$ the "data flow" is as follows:

1. Firstly, $|1\rangle$ is changed to $|0\rangle$ by **X** gate.
2. Then $|0\rangle$ is transformed to a superposition a $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ by **H** gate.
3. Lastly after measurement we obtain state $|0\rangle$ with probability $\frac{1}{2}$ or state $|1\rangle$ with probability $\frac{1}{2}$.

1.5.6 Remark (Detective story) In the previous example 1.5.5 we saw the superposition of states $|0\rangle$ and $|1\rangle$ which will be typical for quantum algorithms. This phenomenon seems quite unique to quantum world. However, as this remark shows, we can easily explain what it means to be "somewhere between $|0\rangle$ and $|1\rangle$".
Once it was given me as an advice to explain things so my grandmother could understand them. Of course it was a hyperbole but now I take it quite literally and I will rewrite the above example. My grandmother likes detective stories and she has read tons of them and there is no series on TV where she does not know the murderer just from the first scene of the film. That is not impressive when we watch the detective Columbo movies but watching the Hercule Poirot series, it still amazes me. In one of the movies I saw several years ago there was a typical locked-room theme with an ingenious resolution. I will not tell you in what film this scene happened not only because I saw it years ago and I was not able to find it but also because I do not want to spoil the movie to you.
There was a house or mansion where a murder took place. The corpse was found lying on the floor in a puddle of blood. All windows of the room were closed and the door was locked from the inside. It was clear the victim did not commit suicide. How did the murderer escape the room? Let us call the murderer Eve. After she killed her victim she opened the door. Then Eve delicately turned the key in the keyhole so that the spring inside the

key mechanism did not push nor pull the latch and therefore the lock was neither locked nor unlocked. When Eve left the room she smashed the door forcefully. The hit was so strong it caused the mechanism of the lock move a bit and the latch was released. The door remained locked.
If we say that locked door is state $|1\rangle$ and $|0\rangle$ means the door is unlocked the whole story is the last part of the program from above. Eve applied the Hadamard gate on the key and put it in superposition of two the states locked and closed. When she smacked the door, she just measured the state which collapsed to the position locked.
There is a lesson to be learned from this example. On a standard computer Eve could not escape the room and left the door locked from inside because standard computers have only bits “on” and “off”. In our case standard door locks can be locked or unlocked and nothing else. That is why the murder seems to be an impossible crime: we just do not expect locks to be locked and unlocked at the same time. Moreover the so called "collapse of the state" seem to be something unearthly and incomprehensible. This behavior of quantum computers should not startle us in the following chapters and when something will look like a miracle it is as miraculous as turning a key into the middle position and smashing the door. We should just remind ourselves that quantum computers are simply not like ordinary computers.

Chapter 2

The Deutsch's algorithm revisited

Let us exercise our freshly acquired knowledge from the last chapter. We will use quantum gates to work out a real meaningful example. From the introduction we already intuitively know how the Deutsch's algorithm works. But until now we lacked the tools to explain it properly. Our exposition of the algorithm is basically that of [18].

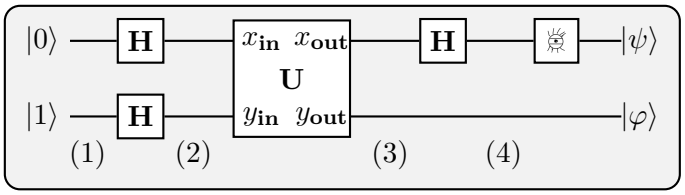
2.1 The Deutsch's algorithm

Let us recollect that a constant function f has the same Think sovalue $f(x)$ for all inputs x.

2.1.1 Theorem (The decision of a constant function) *Suppose B is a computational basis of $\mathbb{C}^2$. Let $f : B \to B$ be any function.* [1] *A quantum computer can decide whether the function f is constant or not by a single inquiry.*

Proof. We will prove this theorem by creating a correct algorithm which will solve the task by one inquiry. We create an algorithm according to the picture below

[1] Let us stress that f is not linear.



where **H** is Hadamard gate and **U** is a linear mapping $\mathbb{C}^2 \otimes \mathbb{C}^2 \to \mathbb{C}^2 \otimes \mathbb{C}^2$ with input $(x_{\mathbf{in}} \otimes y_{\mathbf{in}})$ and output $(x_{\mathbf{out}} \otimes y_{\mathbf{out}})$.The matrix **U** is unitary and we can describe how it behaves on the basis.

$$x_{\mathbf{out}} = x_{\mathbf{in}} \tag{2.1}$$
$$y_{\mathbf{out}} = f(x_{\mathbf{in}}) \oplus y_{\mathbf{in}} \tag{2.2}$$

Note that this is only how the matrix **U** looks like for the basis vectors, it would not make any sense if $x_{\mathbf{out}} = x_{\mathbf{in}}$ for all inputs.

There are four possible options how function f can look like, namely

$$a)\ f_1(x) = x \tag{2.3}$$
$$b)\ f_2(x) = 1 - x \tag{2.4}$$
$$c)\ f_3(x) = 0 \tag{2.5}$$
$$d)\ f_4(x) = 1 \tag{2.6}$$

and so there are four different **U** gates. We have written down all possible binary inputs and outputs of the **U** gate in the following table.

		$f_1(x) = x$		$f_2(x) = 1 - x$		$f_3(x) = 0$		$f_4(x) = 1$	
input		output$_1$		output$_2$		output$_3$		output$_4$	
$x_{\mathbf{in}}$	$y_{\mathbf{in}}$	$x_{\mathbf{out}}$	$y_{\mathbf{out}}$	$x_{\mathbf{out}}$	$y_{\mathbf{out}}$	$x_{\mathbf{out}}$	$y_{\mathbf{out}}$	$x_{\mathbf{out}}$	$y_{\mathbf{out}}$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

This is a convenient way how to represent the matrix **U** because it takes up less space but still it contains all the necessary information. For a model example we will use a matrix for $f_1(x) = x$ which is

$$\mathbf{U}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.7}$$

Now let us see how the program works in greater detail.

1. In step (1) we have $|0\rangle \otimes |1\rangle$ on the input. We apply the Hadamard gate on both wires to get a linear combination of the computational basis.

2. In step (2) we get the following states:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{2.8}$$

We do not have to count with the square roots in the denominator, because the **U** gate is a linear function and we can put it directly on the output. Before we apply the matrix **U** we have to do tensor product on the input sates. The result can be represented by a vector $|\mu\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$.

$$|\mu\rangle = (|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) \tag{2.9}$$

Then we apply the matrix **U**. Because this problem is relatively small, we will show everything on our model example with matrix $\mathbf{U}_1$ to demonstrate how it works with standard matrix notation.

$$\mathbf{U}\,|\mu\rangle = \mathbf{U}_1 \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$
$$= (|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle)$$

3. When we look at step (3) there are four possible results depending on the function f. All possible options are written down in the normalized form as follows:

$$\begin{cases} \pm\dfrac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{if } f(0) = f(1) \\ \pm\dfrac{|0\rangle - |1\rangle}{\sqrt{2}} \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{if } f(0) \neq f(1) \end{cases}$$

4. When we apply the Hadamard gate on the output $x_{\mathbf{out}}$ we finally get the following results in state (4):

$$\begin{cases} \pm|0\rangle \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{if } f(0) = f(1) \\ \pm|1\rangle \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{if } f(0) \neq f(1) \end{cases}$$

We are at the end of the proof. The first state $|\psi\rangle$ indicates whether function f is constant or not and we need only single query. ■

Chapter 3

Hidden subgroup problem

It seems hard to be satisfied with the form of Deutsch's algorithm. Why should we pick the Hadamard gate? It works, but why? There must be something hidden underneath. We promise to the reader that they will find out what is hidden here, but we have to begin slowly from the foundations. Firstly we will have to say what are groups, cosets and subgroups. We will have to speak about characters of groups and the Fourier transform. Finally we explain what a Hidden Subgroup Problem (aka HSP) is. We show in Chapter 4 that HSP is a common core of a class of quantum algorithms. For more about group theory, see, e. g. [1].

3.1 Groups

Firstly for the newcomers we define what groups are and we write down the basic rules for groups. We will begin slowly because groups are not covered in some classes and we do not want to get lost when the nice part has not begun yet.

3.1.1 Definition (Group) We call a set G with a binary operation $*$ a *group* if

$$(x * y) * z = x * (y * z) \text{ for all } x, y, z \in G, \tag{3.1}$$

there exist $e \in G$, such that

$$e * x = x * e = x \text{ for all } x \in G \tag{3.2}$$

and for every $x \in G$ there exist $x^{-1} \in G$ such that

$$x^{-1} * x = x * x^{-1} = e. \tag{3.3}$$

An alternative definition is that of a group is a set equipped with a binary operation $*$, unary operation $(-)^{-1}$ and a constant e. We will denote a group by $(G, *, (-)^{-1}, e)$. Because it can be easily shown that both definitions are equivalent, we will switch between them depending on the circumstances. We will abuse the notation and denote a group by its carrier set.

3.1.2 Notation Having a group $(G, *, (-)^{-1}, e)$ and any element $x \in G$ we denote

$$\begin{aligned}
&\text{the neutral element } e \text{ as } x^0\\
&\text{an element } x * x \text{ as } x^2\\
&\text{an element } x * x * x \text{ as } x^3\\
&\text{an element } \underbrace{x * x * \cdots * x}_{n \text{ times}} \text{ as } x^n.
\end{aligned}$$

We extend the notation for expression

$$x^{-n} = (x^n)^{-1} \tag{3.4}$$

which allows us to write expression as for example $x^{27} * x^{-31} = x^{-4}$.

3.1.3 Definition (Order of element) An *order* of element $x \in G$ is the smallest positive natural number r such that $x^r = e$. If $x^r = e$ does not hold for any positive r we say that the order is infinity.

There are some special groups that behaves "nicely". These deserve their own name.

3.1.4 Definition (Commutative group) A group G is called *Commutative* if

$$(x * y) = (y * x) \text{ for all } x, y \in G \tag{3.5}$$

holds. These groups are frequently called *Abelian groups*.

3.1.5 Definition (Cyclic group) We call a finite group G *cyclic* if there exist $g \in G$ such that any $x \in G$ can be written as g^n some natural number n.

3.1.6 Notation (Equivalence classes modulo n) We say that numbers $a, b \in \mathbb{Z}$ are *congruent modulo* n (where $n \in \mathbb{N}$ is greater than zero) if

$$a = b + k \cdot n \text{ for some } k \in \mathbb{Z} \tag{3.6}$$

holds. We denote this relation by symbol "$\overset{n}{\equiv}$". Relation "$\overset{n}{\equiv}$" is indeed an equivalence therefore we have equivalence classes. The relation is also a congruence with respect to addition "$+$" and multiplication "$\cdot$". We will abuse our notation and we will write $a + b = c$ instead of clumsy $[a]_{\overset{n}{\equiv}} + [b]_{\overset{n}{\equiv}} = [c]_{\overset{n}{\equiv}}$. Set of these equivalence classes will be denoted by $\mathbb{Z}_n$.

3.1.7 Example (Real numbers) We already know some groups and they are not something mysterious.

1. Let us have a set of real numbers $\mathbb{R}$, we can create a group $(\mathbb{R}, +, -, 0)$ with addition as the binary operation and 0 as the neutral element. We can check that all three equations (3.1), (3.2) and (3.3) hold indeed.

2. If we try to create a group $(\mathbb{R}, \cdot, (-)^{-1}, 1)$ where the operation is multiplication and the neutral element is 1, we fail. The equation (3.2) does not hold for number 0. That inspires us to set $\mathbb{R}^\times = \mathbb{R} \setminus \{0\}$ which is the set of all invertible real numbers. It is easy to prove that $(\mathbb{R}^\times, \cdot, (-)^{-1}, 1)$ is actually a group.

3.1.8 Example (Numbers of $\mathbb{Z}_n$) The following examples of groups are essential for Abelian groups because in some sense any Abelian group "behaves" like $\mathbb{Z}_n$.

1. Consider the group $G = (\mathbb{Z}_n, +, -, 0)$. We can easily show that equations (3.1), (3.2) and (3.3) hold. Moreover the equation (3.5) holds as well. That means G is a commutative group. It does not surprise us that any $x \in G$ can be rewritten as

$$x = \underbrace{1 + 1 + \cdots + 1}_{x \text{ times}}.$$

   The group G is therefore a cyclic group.

2. In a similar fashion as in example 3.1.7 we can also construct the group $G = (\mathbb{Z}_n^\times, \cdot, (-)^{-1}, 1)$, where $\mathbb{Z}_n^\times$ is set of all invertible elements. One would ask how many elements $\mathbb{Z}_n^\times$ has. It can be shown that $\mathbb{Z}_n^\times$ has $n - 1$ elements if n is prime. For non-prime n the answer is not so simple. With that in mind we define the Euler's function.

3.1.9 Definition (Euler's totient function) If we are given a group, let us denote it $G = (\mathbb{Z}_n^\times, \cdot, (-)^{-1}, 1)$, where $\mathbb{Z}_n^\times$ is set of invertible elements with respect to multiplication "$\cdot$" for some $n \in \mathbb{N}$ greater than 0. We define the function by putting

$$\varphi(n) = \operatorname{card}(\mathbb{Z}_n^\times) \tag{3.7}$$

We call the function φ *Euler's totient function*. (The function card returns the cardinality of a given finite set. Sometimes for typographic reasons it can be denoted by symbol "|-|".)

Now when we have familiarized ourselves with groups we are going to introduce subgroups and we will proceed to the hidden subgroup problem.

3.1.10 Definition (Subgroup) We say H is a *subgroup* of $(G, *, (-)^{-1}, e)$ if the conditions

$$e \in H, \tag{3.8}$$
$$\text{if } x, y \in H \text{ then also } x * y \in H, \tag{3.9}$$
$$\text{and for every } x \in H, \text{ there exist } x^{-1} \in H. \tag{3.10}$$

hold.

The slogan is "A subgroup of a group is analogous to a subspace of a linear space".

It is just a slogan, yet there is some truth to it. Examples 3.1.7 and 3.1.8 already shown that at least some linear spaces could be understood as groups with some more properties. That is indeed true for all linear spaces. A question comes to our mind, if a subgroup is a generalization of a subspace one could ask is there something as an "affine space" of a group? That is a good question and the answer is positive.

3.1.11 Notation (Smallest subgroup) Having a group with carrier G and subset X of G we denote by $Sg_G(X)$ the smallest group which contains all elements from X. It means

$$Sg_G(X) = \bigcap_{i \in I} \{H \mid H, \text{ where } H \text{ is subgroup of } G \text{ and } X \subseteq H\} \tag{3.11}$$

Disclaimer. From now on we will continue only with Abelian groups to make our lives easier.

3.1.12 Lemma *Let G be an Abelian group and let H be a subgroup of G. We define relation $R(-,-)$ on G by putting*

$$R(x, y) \text{ if and only if } (x * y^{-1}) \in H \tag{3.12}$$

for any $x, y \in G$. The relation $R(-,-)$ is an equivalence relation and the equivalence classes have the form $[x]_R = \{y \in G | y = x * h \text{ for some } h \in H\}$.

PROOF. The proof is quite simple so it is left as an exercise. ■

3.1.13 Definition (Coset) For every $x \in G$ the equivalence classes $[x]_R = x * H$ of Lemma 3.1.12 will be called *coset* w.r.t. H.

3.1.14 Lemma (Cardinality of cosets) *Given a group G and a subgroup H, all cosets in G with respect to H have the same cardinality.*

PROOF. We will denote a coset by $x * H$. We need to show there is a bijection between H and $x * H$. We define a function

$$\begin{aligned} f : H &\to x * H \\ h &\mapsto x * h \end{aligned}$$

The function f is surjective, because for any $k \in x * H$ we can find $h \in H$ such that $f(h) = k$. The element k can be rewritten as $k = x * h_2$ where $h_2 \in H$. We can then choose $h = h_2$. The function f injective because for two distinct $h_1, h_2 \in H$ we have results $f(h_1) \neq f(h_2)$. We will show this by a contradiction. Assume then $h_1 \neq h_2$ but $f(h_1) = f(h_2)$. by simplifying the equality

$$\begin{aligned} f(h_1) &= f(h_2) \\ x * h_1 &= x * h_2 \\ h_1 &= h_2 \end{aligned}$$

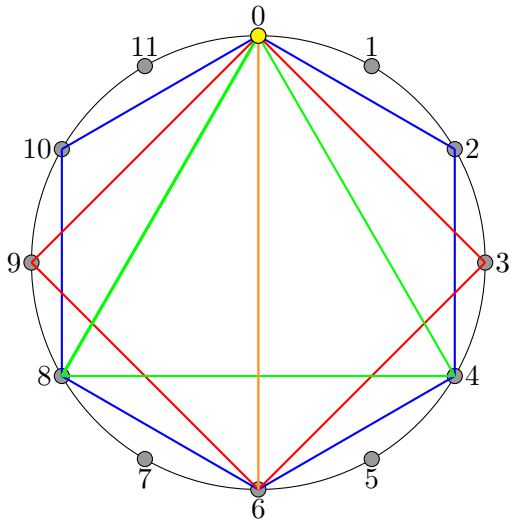
we obtain a contradiction with our assumption. Therefore $f(h_1) \neq f(h_2)$ and the function f must be injective. ■

3.1.15 Lemma (Cosets form a partition) *All cosets in a group G defined by relation (3.12) form a partition of the group G.*

PROOF. It is evident any element $x \in G$ is at least in the coset $x * H$. It remains to be shown that if K,L are cosets such that $K \cap L \neq \emptyset$, then $K = L$. To that end, choose $x \in K \cap L$. According to Lemma 3.1.12 is x equivalent to all elements of L and also to all elements od K and therefore L and K are the same equivalence class (coset). ■

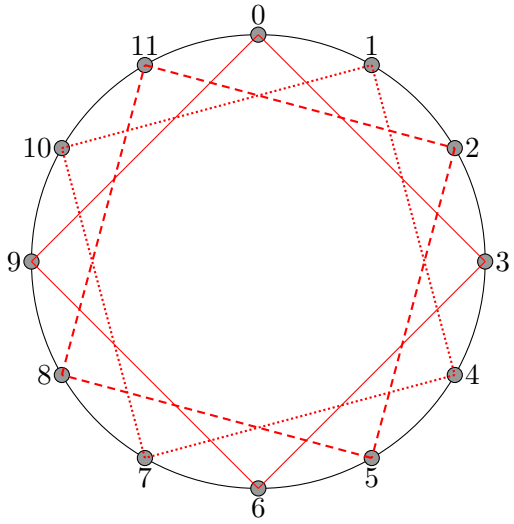
3.1.16 Example (A clock face) Let us have a group $G = (\mathbb{Z}_{12}, +, -, 0)$

which we will represent by a clock face in the picture below.



All possible subgroups of G are depicted in the picture. The subgroup $H_1 = \{0\}$ is depicted in yellow. The subgroup $H_2 = \{0, 6\}$ is depicted in orange. The subgroup $H_3 = \{0, 4, 8\}$ is in green. The subgroup $H_4 = \{0, 3, 6, 9\}$ is depicted in red. The subgroup $H_5 = \{0, 2, 4, 6, 8, 10\}$ is depicted in blue.

Let us focus only on the subgroup H_4. We can see there are three cosets of H_4 which are depicted in the picture below.



The first coset $0 + H_4$ is depicted by the filled line which is in fact the subgroup itself. The second coset $1 + H_4$ is depicted by the dotted line and

the last coset $2 + H_4$ is depicted by dashed line. There are in total three cosets with respect to four-element subgroup in $\mathbb{Z}_{12}$. If we look more carefully we can see that a rule emerges. The number of cosets with respect to any subgroup H is $\frac{\text{card}(G)}{\text{card}(H)}$.

3.1.17 Theorem (Lagrange's theorem [7], [1]) *For any finite commutative group G and H subgroup of G*

$$\text{card}(G) = \text{card}(H) \cdot \#\ \textit{of cosets of } H \tag{3.13}$$

holds.

Proof. By Lemma 3.1.15 cosets form a partition of group G. Since G is finite, so the partition $\{C_1, \ldots, C_n\}$ into cosets. It follows that

$$\begin{aligned} \text{card}(G) &= \sum_{i=1}^{n} \text{card}(C_i) && (3.14)\\ &= \sum_{i=1}^{n} \text{card}(C_1) \text{ by Lemma 3.1.14} && (3.15)\\ &= n \cdot \text{card}(C_1), && (3.16) \end{aligned}$$

where n is number of cosets. ■

3.1.18 Theorem (Fundamental theorem of Abelian groups [1]) *Every finite Abelian group G is isomorphic to $\prod_{i=1}^{k} \mathbb{Z}_{n_i}$ for some k and $n_1, n_2 \ldots n_k$.*

3.1.19 Theorem (Euler's theorem) *Let G be a finite Abelian group and n is number of its elements. For every $x \in G$ the equation* [1]

$$x^n = e \tag{3.17}$$

holds.

Proof. We can inspire ourselves by the proof of the Fermat's little theorem [15]. We denote elements of G by $x_1, x_2, \ldots, x_n$. For every $a \in G$ the function

$$f : x \mapsto x * a \tag{3.18}$$

is a bijection,we have shown that in lemma 3.1.14. Thus

$$\begin{aligned} x_1 * x_2 * \cdots * x_n &= f(x_1) * f(x_2) * \cdots * f(x_n)\\ &= x_1 * a * x_2 * a * \cdots * x_n * a \text{ group } G \text{ is commutative}\\ &= \underbrace{a * a * \cdots * a}_{n \text{ times}} * x_1 * x_2 * \cdots * x_n && (3.19)\\ &= a^n * x_1 * x_2 * \cdots * x_n && (3.20) \end{aligned}$$

[1] According to notation 3.1.2 where we said that $x^n = \underbrace{x * x * \cdots * x}_{n \text{ times}}$.

Since

$$x_1 * x_2 \cdots * x_n = a^n * x_1 * x_2 \cdots * x_n$$

holds, then also the equation

$$e = a^n \tag{3.21}$$

holds. ■

3.1.20 Example (Almost Fermat's little theorem) The proof of 3.1.19 is so nice it is tempting to apply it in the following example.

1. Let $G = (\mathbb{Z}_6, +, -, 0)$ then $a^6 = 0$ should hold for all $a \in \mathbb{Z}_6$. Remember that we denote $\underbrace{a + a + \cdots + a}_{n \text{ times}} = a^n$. We will rewrite the proof and so we get

$$\sum_{x=0}^{5} x = \sum_{x=0}^{5} (x + a) \tag{3.22}$$

$$= \underbrace{a + a + \cdots + a}_{6 \text{ times}} + \sum_{x=0}^{5} x_i$$

$$= a^6 + \sum_{x=0}^{5} x. \tag{3.23}$$

and therefore

$$a^6 = 0 \text{ in } \mathbb{Z}_6. \tag{3.24}$$

2. Now we will show a familiar problem and which makes perfect sense. For the group $G = (\mathbb{Z}_n^\times, \cdot, (-)^{-1}, 1)$ (see Example 3.1.8) we get

$$a^{\varphi(n)} = 1. \tag{3.25}$$

The proof is the same as above. This expression is known to us from the lessons of discrete mathematics where we call it Euler's theorem. When the number n is prime, we call the theorem *Fermat's little theorem*, since $\varphi(n) = n - 1$ holds. You can find the theorem in the book [20].

3.2 Characters of groups

It is tempting to start with the hidden subgroup problem but we are not prepared yet. We will firstly need tools to solve the problem before we will meet it face to face. We will touch characters of groups very slightly. The presented theory is based on [8].

3.2.1 Definition (Group homomorphism) Suppose $(A, *, (-)^{-1}, e_A)$ and $(B, *, (-)^{-1}, e_B)$ are groups and let $h : A \to B$ be a function. We call the function h a group homomorphism if for all $x, y \in A$ the equations

$$h(e_A) = e_B \tag{3.26}$$
$$h(x * y) = h(x) * h(y) \tag{3.27}$$
$$h(x^{-1}) = (h(x))^{-1} \tag{3.28}$$

hold.

3.2.2 Definition (Character) Suppose $(G, *, (-)^{-1}, e)$ is a finite Abelian (commutative) group. We call any group homomorphism

$$\chi : G \to \mathbb{C}^{\times}. \tag{3.29}$$

a *character (of group G).* [2]

3.2.3 Lemma (Group of characters) *Characters of any commutative group G form a group $\widehat{G}$ by defining for*

$$\begin{aligned} \chi_1 : G &\to \mathbb{C}^{\times} & \chi_2 : G &\to \mathbb{C}^{\times} \\ x &\mapsto \chi_1(x) & x &\mapsto \chi_2(x) \end{aligned}$$

the operation "$*$" as

$$\chi_1 * \chi_2 = \chi_1(x) \cdot \chi_2(x). \tag{3.30}$$

We define the inverse to χ_1 as

$$\chi^{-1}(x) = (\chi(x))^{-1}. \tag{3.31}$$

The neutral element is such a character χ that

$$\chi(x) = 1 \tag{3.32}$$

holds for all $x \in G$.

Proof. It is trivial to show that the axioms for group from definition 3.1.1 do hold. ■

If we want to know about characters, we can ask what the results of the mappings χ_i are or how many of these characters there are.

[2] Recall from Exercise 3.1.7 the notation for all invertible elements. Analogously $\mathbb{C}^{\times} = (\mathbb{C} \setminus \{0\}, \cdot, (-)^{-1}, 1)$ is a multiplicative group of all invertible complex numbers. It is obvious that $\mathbb{C}^{\times}$ is commutative.

3.2.4 Lemma (Values of characters are roots of unity) *Let us have a finite Abelian group $(G, *, (-)^{-1}, e)$ and let n denote the number of elements of G. Then any character*

$$\chi : G \to \mathbb{C}^{\times} \tag{3.33}$$
$$g \mapsto \chi(g) \tag{3.34}$$

the equality

$$\chi(g)^n = 1 \tag{3.35}$$

holds for any $g \in G$.

Proof. Given a finite Abelian group $G = (G, *, (-)^{-1}, e)$, by Euler's theorem 3.1.19 we can write

$$g^n = e \tag{3.36}$$

for any $g \in G$. A character χ is a group homomorphism therefore we can continue by

$$\begin{aligned} \chi(g)^n &= \chi(g^n) \\ &= \chi(e) \\ &= 1 \end{aligned} \tag{3.37}$$

■

The lemma 3.2.4 says that values of characters are in fact roots of unity.

3.2.5 Lemma (Number of characters of a cyclic group) *There are n distinct characters for any cyclic commutative group $G = (G, *, (-)^{-1}, e)$, where n is number of its elements.*

Proof. Let g be the generator of the group $(G, *, (-)^{-1}, e)$. Every $a \in G$ can be written according to definition 3.1.5 as

$$a = \underbrace{g * g * \cdots * g}_{m \text{ times}} \text{ for some } m. \tag{3.38}$$

For an arbitrary character

$$\chi : G \to \mathbb{C}^{\times} \tag{3.39}$$
$$g \mapsto \chi(g) \tag{3.40}$$

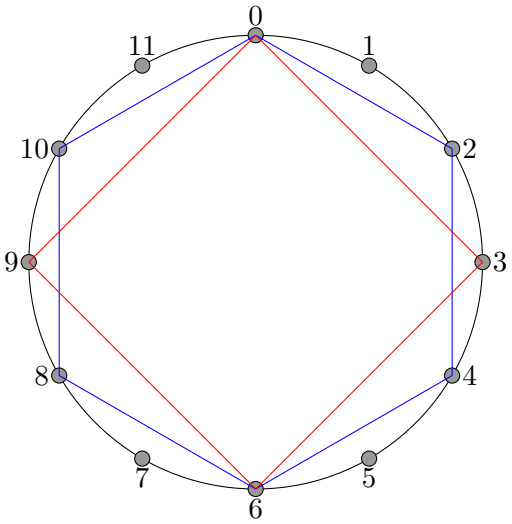
we know as a result of lemma 3.2.4 that the value $\chi(g)$ is n^{th} root of unity. We can choose one of the possible n values of $\chi(g)$ and it determines the value of $\chi(a)$ because

$$\begin{aligned}\chi(a) &= \chi(\underbrace{g * g * \ldots * g}_{m \text{ times}}) \\ &= \underbrace{\chi(g) \cdot \chi(g) \cdot \ldots \cdot \chi(g))}_{m \text{ times}}\end{aligned} \tag{3.41}$$

must hold. The characters are distinct since they differ by their value at g. ■

Unfortunately for us not all Abelian groups are cyclic. We have to prove something much stronger and we have to play better cards from our sleeves. Namely, we have to show how to extend characters from a subgroup to a group.

3.2.6 Example To demonstrate the problem we will use our well-known clock face example 3.1.16. We could be given, let us say, the number 2 and the subgroup $H_3 = \{0, 3, 6, 9\}$ with its character χ. This is the image we drew.



The task is to find a character $\widetilde{\chi}$ such that it behaves like χ for all elements of the subgroup H_3 and is reasonably defined at number 2. If the value of $\chi(6)$ was let us say $\chi(6) = -1$ we would like to have

$$\begin{aligned}-1 &= \chi(6) \\ &= \widetilde{\chi}(2+2+2) \\ &= \widetilde{\chi}(2) \cdot \widetilde{\chi}(2) \cdot \widetilde{\chi}(2) \\ &= \widetilde{\chi}(2)^3.\end{aligned}$$

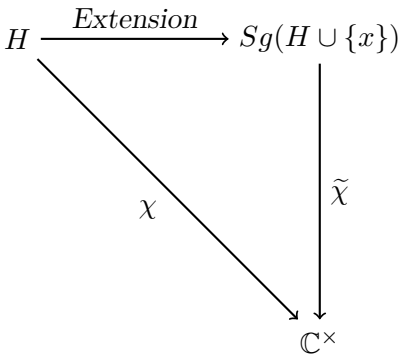
In our case we need to hope that

$$\begin{aligned} 0 &= \chi(0) \\ &= \widetilde{\chi}(2+2+2+2+2+2) \\ &= \widetilde{\chi}(2)^6 \end{aligned}$$

does hold as well.

This simple idea will inspire us how to act in the more general case which is developed in [8].

3.2.7 Lemma (Extension of characters) *Consider commutative group $(G, *, (-)^{-1}, e)$. Let H be a subgroup of G and $c \in G$. For any character χ there exists a character $\widetilde{\chi}$ such that the diagram:*



commutes.

Proof. Thanks to Euler's theorem 3.1.19 there exists a smallest $k \in \mathbb{N}$ greater than 0 such that $x^k = h \in H$ (k can be at worst the number of elements in G). For a given character

$$\begin{aligned} \chi : H &\to \mathbb{C}^\times \\ h &\mapsto \chi(h) \end{aligned} \tag{3.42}$$

we define $\widetilde{\chi}(x)$ as a chosen solution of the equation

$$\widetilde{\chi}(x)^k = \chi(h). \tag{3.43}$$

For any element $g \in H$ and $i \in \mathbb{N}$ we define

$$\widetilde{\chi}(g * x^i) = \chi(g) \cdot \widetilde{\chi}(x)^i. \tag{3.44}$$

We should check whether our definition is correct and whether the mapping $\widetilde{\chi}$ is consistent because there could be an element in $h \in H$ which could be written for some (distinct) $g_1, g_2 \in H$ as

$$h = g_1 * x^i = g_2 * x^j, \tag{3.45}$$

yet $\widetilde{\chi}(g_1 * x^i)$ must be equal to $\widetilde{\chi}(g_2 * x^j)$. We can write

$$g_2^{-1} * g_1 = x^j * (x^i)^{-1} \text{ that is also element of } H.$$

We get $x^j = x^i * g_2^{-1} * g_1$. We show that the following equation holds.

$$\begin{aligned}\chi(g_2) \cdot \widetilde{\chi}(x)^j &= \chi(g_2) \cdot \widetilde{\chi}(x^j) \\ &= \chi(g_2) \cdot \widetilde{\chi}(x^i * g_2^{-1} * g_1) \\ &= \underbrace{(\widetilde{\chi}(g_2))^{-1}}_{=(\chi(g_2))^{-1}} \cdot \underbrace{\widetilde{\chi}(g_1)}_{=\chi(g_1)} \\ &= \chi(g_1) \cdot \widetilde{\chi}(x^i) \end{aligned} \tag{3.46}$$

■

3.2.8 Remark Equation (3.43) gives k different extensions, one extension per one choice of the k^{th} root of 1.

3.2.9 Remark For a finite group G with generators $\{x_1, x_2, \dots x_n\}$ we can create extensions of any character of its subgroup by repeating the process given by the extension lemma 3.2.7.

3.2.10 Theorem (Number of characters of an Abelian group [8]) *For any finite Abelian group G there are $\text{card}(G)$ distinct characters.*

Proof. We will denote by H_i^j a subgroup of $\prod_{i=1}^m \mathbb{Z}_{n_i}$ which has all possible non-zero elements at positions i to j and zeros everywhere else. The fundamental theorem 3.1.18 allows us to rewrite the group G as $\prod_{i=1}^m \mathbb{Z}_{n_i}$ for some m and $n_1, n_2, \dots, n_m$. It is clear that the subgroups H_i^i for different i all overlap trivially at element $e = (0, 0, \dots, 0)$ and they all have one generator g_i which has 1 at position i and 0 everywhere else. These subgroups are cyclic. We define

$$\chi_0(e) \mapsto 1 \tag{3.47}$$

and inductively all characters χ_{i+1} of the group $G_{i+1} = H_1^{i+1}$ from the $G_i = H_1^i$ by Lemma 3.2.7. Remark 3.2.8 by equation (3.43) says that the number of extensions from characters of group G_i to a group G_{i+1} equals to the lowest power p of generator $g_{i+1} \in H_{i+1}^{i+1}$ such that $g_{i+1}^p \in G_k$. Since these groups overlap only at the neutral element e we get $g_i^{n_i} = e$ by Euler's theorem 3.1.19. From that follows there are n_i extensions of characters from G_i to G_{i+1}. The number n_i is equal to cardinality of the group H_{i+1}^{i+1}. With every k^{th} inductive step the number of characters grows by n_k. We conclude that the group $\prod_{i=1}^m \mathbb{Z}_{n_i}$ has

$$\sum_1^m n_i = \sum_1^m \text{card}(\mathbb{Z}_{n_i}) = \text{card}(\prod_{i=1}^m \mathbb{Z}_{n_i}) = \text{card}(G) \tag{3.48}$$

distinct characters. ■

3.2.11 Lemma *Given a character χ, the inverse character χ^{-1} has complex conjugate values to χ.*

Proof. The proof is simple, for any $a \in \mathbb{C}$ the equation

$$a \cdot a^* = |a|^2 \tag{3.49}$$

holds. We simply substitute $a = \chi(x)$ for arbitrary x and we get

$$\begin{aligned} \chi(x) \cdot \chi(x)^* &= |\chi(x)|^2 \\ &= 1 \text{ by Lemma 3.2.4.} \end{aligned} \tag{3.50}$$

We get $\chi(x)^{-1} = \chi(x)^*$. ■

It may seem silly at first but there is a good reason to perceive characters as vectors.

3.2.12 Remark A character χ of a group with carrier $G = \{x_1, x_2, \dots x_n\}$ can be represented by its values which we can write in a column vector

$$\chi = \begin{pmatrix} \chi(x_1) \\ \chi(x_2) \\ \dots \\ \chi(x_n) \end{pmatrix}.$$

We can now use tools of linear algebra in further exploration of the topic.

3.2.13 Theorem (Characters form an orthogonal set) *Let G be a finite Abelian group with elements $\{x_1, x_2, \dots, x_n\}$ and let $\chi_1, \dots, \chi_n$ be characters on G. The characters form an orthogonal set in the sense that*

$$\langle \chi_i | \chi_j \rangle = \langle \begin{pmatrix} \chi_i(x_1) \\ \chi_i(x_2) \\ \dots \\ \chi_i(x_n) \end{pmatrix} | \begin{pmatrix} \chi_j(x_1) \\ \chi_j(x_2) \\ \dots \\ \chi_j(x_n) \end{pmatrix} \rangle = \begin{cases} 0, & \text{if } i \neq j \\ \\ n, & \text{if } i = j \end{cases} \tag{3.51}$$

holds.

PROOF. The proof is essentially presented in [9]. We will rewrite the $\langle \chi_i | \chi_j \rangle$ as in the equation (1.4) and we get

$$\begin{aligned}\langle \chi_i | \chi_j \rangle &= \sum_{k=1}^{n} \chi_i(x_k)^* \chi_j(x_k) \\ &= \sum_{k=1}^{n} \chi_i(x_k)^{-1} \chi_j(x_k) \quad \text{by Lemma 3.2.11} \\ &= \sum_{k=1}^{n} (\chi_i^{-1} \chi_j)(x_k) \end{aligned} \tag{3.52}$$

If $i = j$ we get

$$(\chi_i^{-1} \chi_j)(x) = (\chi_i^{-1} \chi_i)(x) = 1 \tag{3.53}$$

therefore

$$\sum_{k=1}^{n} (\chi_i^{-1} \chi_j)(x_k) = \sum_{k=1}^{n} 1 = n. \tag{3.54}$$

For $i \neq j$ we will use a trick that the function $f : f(x) = x + g$ for a fixed $g \in G$ is a permutation. We will choose g such that $\chi(g) \neq 1$. We just simplify the term $(\chi_i^{-1} \chi_j)(x_k)$ as $\chi(x_k)$ and we can continue where we stopped.

$$\begin{aligned}\sum_{k=1}^{n} \chi(x_k) &= \sum_{k=1}^{n} \chi(f(x_k)) \\ &= \sum_{k=1}^{n} \chi(x_k + g) \\ &= \sum_{k=1}^{n} \chi(x_k) \cdot \chi(g) \\ &= \chi(g) \cdot \sum_{k=1}^{n} \chi(x_k) \end{aligned} \tag{3.55}$$

The expression $\chi(g) \neq 1$ and therefore $\langle \chi_i | \chi_j \rangle = \sum_{k=1}^{n} \chi(x_k) = 0$. [9] ■

3.3 Fourier transform on finite Abelian groups

The Fourier transform may seem complicated for beginners, but actually we all perform the Fourier transform everyday. It is simply finding of coordinates for a nice basis. Although the naming is new, we are about to explore an already known part of linear algebra. You can study this topic in greater detail in the book [3].

3.3.1 Example (It is elementary . . .) With disdain some people look down at mathematics saying "You only need simple arithmetics." Undeniably they are right, everything we learned at elementary school is all we need. We just change the names and denote things differently. Imagine we want to buy a new laptop and we are comparing the displays. One of them has 1280×800 pixels, where 1280 pixels are in horizontal way and 800 in vertical way. We just give a special name to counting pixels in perpendicular directions. We will call it the Fourier transform. Nobody thinks about how challenging this Fourier transform was nor how troublesome is to write Fourier series $(1280, 800)$. That is all, we know everything since we were six years old. Be brave!

3.3.2 Theorem (Fourier's theorem) *Having a linear space L with orthonormal basis $B = (b_1, b_2, \dots b_n)$, every $v \in L$ can be rewritten as*

$$v = \sum_{i=1}^{n} \langle b_i|v\rangle b_i. \tag{3.56}$$

The scalars $\langle b_i|v\rangle$ are called Fourier's coefficients.

Proof. Let us put $w = \sum_{i=1}^{n} \langle b_i|v\rangle b_i$. We show then $w = v$. For any b_k we can write

$$\begin{aligned}
\langle b_k|w - v\rangle &= \langle b_k| \sum_{i=1}^{n} \langle b_i|v\rangle b_i - v\rangle \\
&= \langle b_k| \sum_{i=1}^{n} \langle b_i|v\rangle b_i\rangle - \langle b_k|v\rangle \\
&= \sum_{i=1}^{n} \langle b_k|\langle b_i|v\rangle b_i\rangle - \langle b_k|v\rangle.
\end{aligned}$$

If $b_i \neq b_k$ then $\langle b_k|\langle b_i|v\rangle b_i\rangle = 0$. We can simplify the expression above and continue with

$$\langle b_k|\langle b_k|v\rangle b_k\rangle - \langle b_k|v\rangle = \langle b_k|v\rangle - \langle b_k|v\rangle = 0. \tag{3.57}$$

To satisfy $\langle b_k|w - v\rangle = 0$ for any b_k the equation $w - v = 0$ must hold. That means

$$v = w = \sum_{i=1}^{n} \langle b_i|v\rangle b_i. \tag{3.58}$$

■

Finding coordinates is nothing new. There is a special application when the basis is orthonormal which deserves its own notation.

3.3.3 Notation (General Fourier transform) If B is orthonormal basis, we call the function coord_B the general *Fourier transform.*

3.3.4 Notation (Inverse Fourier transform) The function coord_B^{-1} is called the *inverse Fourier transform*, if B is an orthonormal basis.

3.3.5 Remark For space $\mathbb{C}^n$ with an orthogonal basis $B = (b_1, b_2, \dots b_n)$, the Fourier transform can be written as a matrix multiplication by matrix $\mathbf{X}^\dagger = (b_1, b_2, \dots b_n)^\dagger$ and the inverse Fourier transform is represented by matrix $\mathbf{X}$. This observation follows straightforwardly from equation (3.56). It follows that the Fourier transform must be unitary.

3.3.6 Notation (Fourier image) Given a function $g : G \to \mathbb{C}^n$, where G is a group, we define

$$\mathcal{F}_B : g \mapsto f_B g. \tag{3.59}$$

The function $f_B g$ is denoted by $\widehat{g}$ and called the *Fourier image* of function g. In a similar fashion we define

$$\mathcal{F}_B^{-1} : g \mapsto f_B^{-1} g. \tag{3.60}$$

We will call $f_B^{-1} g$ the *inverse Fourier image* of function g and denote it by $\check{g}$. It is worth to remark that $\widehat{\check{g}} = g = \check{\widehat{g}}$.

The Fourier transform is the most powerful tool we have seen yet. The despicable hero of our story Eve (the eavesdropper) is close to using this mighty tool in the quantum problems. If only characters formed an orthonormal basis.

3.3.7 Lemma (Characters form orthogonal basis) *The characters of a finite commutative group G with n elements form a basis of $\mathbb{C}^n$. We can normalize the basis by dividing all characters by constant $\sqrt{n}$.*

Proof. We know from Theorem 3.2.13 that characters of a finite commutative group G form an orthogonal set. Theorem 3.2.10 says there are n distinct characters. The only problem could be if there was a character χ such that $\chi(x) = 0$ for all $x \in G$ which Lemma 3.2.4 prohibits and also tells us $|\chi(x)| = 1$ for all $x \in G$. By Theorem 3.2.13 $\langle \chi | \chi \rangle = ||\chi||^2 = n$ for every χ.Therefore every χ can be normalised when divided by $\sqrt{n}$. ■

3.3.8 Example (Fourier transform with characters I) Suppose we have a commutative group G with elements $\{x_1, x_2, \dots, x_n\}$. There are n characters of this group, namely $X = \{\chi_1, \chi_2, \dots, \chi_n\}$ which we can write as

columns of a matrix called $\mathbf{X}$. For a function $g : G \to \mathbb{C}$ the question is: how does the Fourier image of g look like? The set X is orthogonal and we can normalize it by multiplying every character by factor $\frac{1}{\sqrt{n}}$. The Fourier image is then

$$\widehat{g} = \frac{1}{\sqrt{n}} \begin{pmatrix} \chi_1(x_1) & \chi_2(x_1) & \dots & \chi_n(x_1) \\ \chi_1(x_2) & & & \chi_n(x_2) \\ \dots & & & \dots \\ \chi_1(x_n) & \chi_2(x_n) & \dots & \chi_n(x_n) \end{pmatrix}^{\dagger} \cdot \begin{pmatrix} g(x_1) \\ g(x_2) \\ \dots \\ g(x_n) \end{pmatrix} \tag{3.61}$$

We have just rewritten the definition of the Fourier transform into matrix notation.

3.3.9 Example (Fourier transform with characters II) Some people prefer examples with numbers. We will continue with the previous example 3.3.8. Let G have only two elements. The Fourier transform looks like as follows:

$$\widehat{g} = \frac{1}{\sqrt{n}} \begin{pmatrix} \chi_1(x_1) & \chi_2(x_1) \\ \chi_1(x_2) & \chi_2(x_2) \end{pmatrix}^{\dagger} \cdot \begin{pmatrix} g(x_1) \\ g(x_2) \end{pmatrix} \tag{3.62}$$

We have seen it before, have we not? Let us give some hints. The characters are

$$\chi_1 = \begin{pmatrix} \chi_1(x_1) \\ \chi_1(x_2) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{3.63}$$

$$\chi_2 = \begin{pmatrix} \chi_2(x_1) \\ \chi_2(x_2) \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{3.64}$$

and the number of characters n is 2.

$$\widehat{g} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\dagger} \cdot \begin{pmatrix} g(x_1) \\ g(x_2) \end{pmatrix} \tag{3.65}$$

We have seen this in Notation 1.4.2 and called it the Hadamard gate.

3.3.10 Notation (Quantum Fourier transform) Examples 3.3.8 and in particular 3.3.9 are pushing us to a special use of the Fourier transform in quantum circuits. The computational basis from definition 1.2.4 pushes us to use only spaces of dimension 2^n for some $n \in \mathbb{N}$. We will call the Fourier transform *quantum*, if it finds coordinates with respect to the basis consisting of 2^n characters.

3.4 Hidden subgroup problem

We already know how powerful a tool the Fourier transform is. We have met characters and we know what a subgroup is. But what is a hidden subgroup and what is the problem anyway?

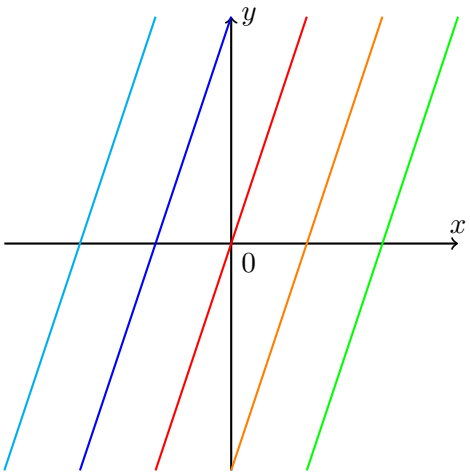
3.4.1 (Simon's promise [10]) Let there be a group $(G, *, (-)^{-1}, e)$ and set C which we will call colours.

A function $\rho : G \to C$ is said to fulfill *Simon's promise*, if the following condition holds:
Simon's promise: *"There exist a subgroup H of G such that ρ is constant on every coset with respect to H and distinct on every distinct coset."* We say that ρ *hides* subgroup H. The Hidden Subgroup Problem (HSP, for short) is to find the hidden subgroup for a function fulfilling Simon's promise.

3.4.2 Example (Example in $\mathbb{R}^2$) In the space $\mathbb{R}^2$ we are given a function

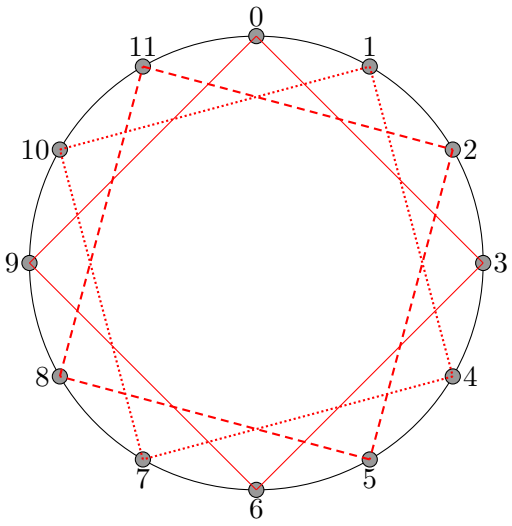
$$\rho : x \mapsto \begin{pmatrix} 3 & -1 \end{pmatrix} x. \tag{3.66}$$

The set of colours is in this case the set of real numbers $\mathbb{R}$, but for better imagination we will represent the cosets with actual colours. The kernel of ρ is the hidden subgroup because $\rho(x) = \textit{constant}$ for all affine spaces collinear with kernel of ρ and for distinct cosets we get distinct values.

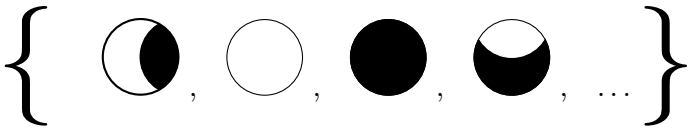


3.4.3 Example (Thirteen months in a year) It would be a shame not to use our favourite example for groups . . . the clock face from Example 3.1.16. We can, for example, define a function ρ such that for x the output would be the remainder of x after division by 3. It is easy to show that the Simon's promise holds for such a function ρ, especially if we draw an illustrative image

of the cosets.



Now we will apply this example on a more concrete example. Imagine that a year has only 364 days and they are represented by the group $\mathbb{Z}_{364}$. The set of colours is the set of all possible and impossible Moon phases.



An oracle, which knows that one Moon cycle is 28 days long tells us that a function ρ which assigns the Moon phase to a day meets Simon's promise. We can now solve how long a month is and we will be amazed there are 13 months in one year. We can use the same example in different applications and find periodicity of many things which happen over a repeating time interval if the Simon's promise is fulfilled.

The next example will be more practical

3.4.4 Example (Order of an element in $\mathbb{Z}_n^\times$) Let n be a natural number greater than 1 (in real cases this number would be large) and an invertible number $a \in \mathbb{Z}_n^\times$ (i.e., such that $\gcd(a, n) = 1$). The question is, what is the order of a? In this example we will show how to reformulate this task as the hidden subgroup problem. We denote the order of a by symbol r. Our group will be $G = (\mathbb{Z}, \cdot, (-)^{-1}, 1)$, the set of colours will be $C = (\mathbb{Z}_n^\times, \cdot, (-)^{-1}, 1)$. We define our function as follows.

$$\begin{aligned} \rho : \mathbb{Z} &\to \mathbb{Z}_n^\times \\ x &\mapsto a^x \end{aligned}$$

Unsurprisingly the hidden subgroup H has the underlying set

$$H = \{g \in G \mid x \text{ is divisible by } r\}.$$

We should check whether Simon's promise holds. We choose elements x, y of G. Just for clarification we can rewrite the values of function ρ as follows.

$$\rho(x) = a^x = \underbrace{a \cdot a \cdots a}_{x \text{ times}} \stackrel{n}{\equiv} R_1 \tag{3.67}$$

$$\rho(y) = a^y = \underbrace{a \cdot a \cdots a}_{y \text{ times}} \stackrel{n}{\equiv} R_2, \tag{3.68}$$

where R_1, R_2 are the remainders after division by n. The values $\rho(x), \rho(y)$ are equal if $R_1 = R_2$. It can only be the case if

$$a^x \stackrel{n}{\equiv} a^y \tag{3.69}$$

$$a^{x-y} \stackrel{n}{\equiv} 0 \tag{3.70}$$

It is trivial to show the expression $x - y$ is multiple of r.

$$x - y = (K_1 + K_2) \cdot r \text{ for some } K_1, K_2 \in \mathbb{Z}. \tag{3.71}$$

$$x - K_1 \cdot r = y + K_2 \cdot r$$

$$x + K_1' \cdot r = y + K_2 \cdot r \tag{3.72}$$

We get exactly the cosets we predicted. The cosets $(x + K_1' \cdot r)$, $(y + K_2 \cdot r)$ are the same if and only if the equality $\rho(x) = \rho(y)$ holds.

3.4.5 Example (Order of an element in $\mathbb{Z}_n$) We will show a variation of the previous example which we get almost for free. Suppose number n is again a natural number greater than 1 and suppose $a \in \mathbb{Z}_n$. The question is as in the previous example: what is the order r of a? Our group will be $G = (\mathbb{Z}, +, -, 0)$, the set of colours will be $C = (\mathbb{Z}_n, +, -, 0)$. We define our function as follows.

$$\begin{aligned} \rho : \mathbb{Z} &\to \mathbb{Z}_n \\ x &\mapsto a^x \end{aligned}$$

We have elements x, y of G. We simply rewrite the values of ρ

$$\rho(x) = a^x = \underbrace{a + a + \cdots + a}_{x \text{ times}} \stackrel{n}{\equiv} R_1 \tag{3.73}$$

$$\rho(y) = a^y = \underbrace{a + a + \cdots + a}_{y \text{ times}} \stackrel{n}{\equiv} R_2, \tag{3.74}$$

where R_1, R_2 are remainders after division by n. The values $\rho(x), \rho(y)$ are equal if and only if $R_1 = R_2$. It follows

$$a^x \stackrel{n}{\equiv} a^y \tag{3.75}$$

$$a^{x-y} \stackrel{n}{\equiv} 0 \tag{3.76}$$

Again it is easy to show that $x - y$ is multiple of r. The rest is what we have already seen.

$$x - y = (K_1 + K_2) \cdot r \text{ for some } K_1, K_2 \in \mathbb{Z}. \tag{3.77}$$

$$x - K_1 \cdot r = y + K_2 \cdot r$$

$$x + K_1' \cdot r = y + K_2 \cdot r \tag{3.78}$$

The values $\rho(x) = \rho(y)$ are the same if and only if the cosets $(x + K_1' \cdot r)$, $(y + K_2 \cdot r)$ are the same.

3.4.6 Example (Discrete logarithm problem [10]) The set C of colours we will use in this example is the group $(\mathbb{Z}_n^\times, \cdot, (-)^{-1}, 1)$. We will need another constraint that the group is cyclic. Therefore by Definition 3.1.5 every element $a \in \mathbb{Z}_n^\times$ can be rewritten as g^k for a generator g some k. The value k is called *discrete logarithm of* a. The problem is to find the value of k for given a.[3] Our group G in this problem will be group $\mathbb{Z}_{\varphi(n)} \times \mathbb{Z}_{\varphi(n)}$ where the we define operation "+" pointwise. [4] It remains to define the function ρ by putting

$$\rho : \mathbb{Z}_{\varphi(n)} \times \mathbb{Z}_{\varphi(n)} \to \mathbb{Z}_n^\times$$
$$(x, y) \mapsto g^x \cdot a^{-y}.$$

We shall check whether the Simon's promise is satisfied and even more importantly we show how the hidden subgroup looks like. We will take some inspiration from the previous example. The results of function ρ for inputs (x_1, y_1) and (x_2, y_2) are the same, if and only if

$$g^{x_1} \cdot a^{-y_1} \overset{n}{\equiv} g^{x_2} \cdot a^{-y_2}. \tag{3.79}$$

Now we use the fact that $a = g^k$

$$g^{x_1} \cdot g^{-y_1^k} \overset{n}{\equiv} g^{x_2} \cdot g^{-y_2^k} \tag{3.80}$$

$$g^{x_1 - y_1^k} \overset{n}{\equiv} g^{x_2 - y_2^k}. \tag{3.81}$$

The exponents on both sides could be the same or dare we say some "period" away. It can be captured by equations

$$x_1 - y_1^k \overset{\varphi(n)}{\equiv} x_2 - y_2^k$$

$$x_1 - x_2 \overset{\varphi(n)}{\equiv} y_1^k - y_2^k$$

$$x_1 - x_2 \overset{\varphi(n)}{\equiv} (y_1 - y_2)^k. \tag{3.82}$$

The equation (3.82) could be intuitively written as $x_1 - x_2 \overset{\varphi(n)}{\equiv} k \cdot (y_1 - y_2)$, unfortunately this is not allowed by our notation. These equations above hold only if (x_i, y_i) are from a subgroup with carrier set

$$H = \{(x, y) \in \mathbb{Z}_{\varphi(n)} \times \mathbb{Z}_{\varphi(n)} \mid x = y^k\}. \tag{3.83}$$

[3] The discrete logarithm problem can be applied in cryptography. The book [15] describes the discrete logarithm problem in grater detail.

[4] Remember notation 3.1.2. In this example the group operation "*" is "+" and so $a^k = \underbrace{a + a + \cdots + a}_{k \text{ times}}$ which could be understood as $k \cdot a$.

Chapter 4

Towards more complex problems

Eve is about to discover the algorithm to read Alice's and Bob's messages. The path through groups, quantum gates, the Fourier transforms is about to end, finally. In this chapter we will show how to solve the hidden subgroup problems in general. We will see how to solve the factorization problem and we will finish with Shor's algorithm.

4.1 Solving hidden subgroup problem

In this section we will show how to create quantum algorithm for general hidden subgroup problem on finite Abelian groups.

We will begin slowly by denoting groups and sets we will use. After that the interesting part will come. There is a group G and we have a set of colours C. The function ρ hides the subgroup H. At this point we will change our notation and we will use $|S|$ instead of $\mathrm{card}(S)$ for a set S. We denote by $\widehat{G}$ the group of all characters from G to $\mathbb{C}^\times$. We define $H^\perp$ as

$$H^\perp = \{\chi \in \widehat{G} \mid \chi(h) = 1 \text{ for all } h \in H\}. \tag{4.1}$$

We can check that $H^\perp$ is a subgroup of $\widehat{G}$. It may seem unreasonable to define $H^\perp$ in this way and even worse to use the letter "H" since it is not a subgroup of G. After some observation we realize it is in fact something like an orthogonal complement on groups. Firstly the number 1 is in fact the

neutral element in $\mathbb{C}^\times$. The element h is something like a ket and $\chi \in H^\perp$ is something like a bra. If you are still uncomfortable there is an isomorphism between $H^\perp$ and some subgroup of G. The last thing we will need for now is a handy way how to denote all the cosets with respect to H.

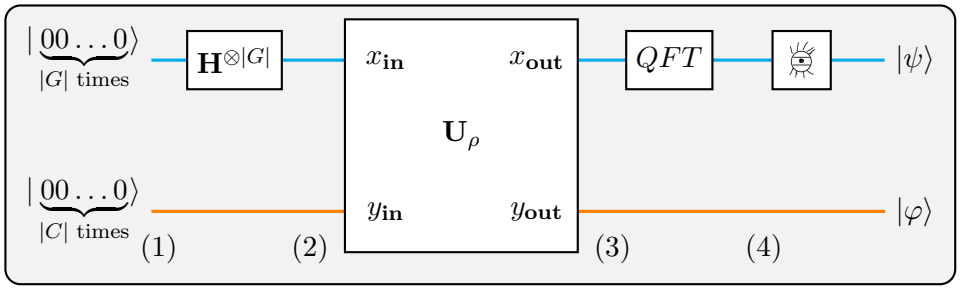
4.1.1 Definition (Transversal) We define T to be a set containing precisely one element from each coset, therefore an element $t_i \in T$ represents the i^{th} coset. It is clear that the equation

$$|T| = \frac{|G|}{|H|} \tag{4.2}$$

holds. The set T is called *transversal*.

Disclaimer. At this moment we will assume the group G is $(\mathbb{Z}_2)^n$ for some n which will simplify our computations because the computational basis 1.2.4 we use has binary states. The algorithm for different groups is similar. It will also help us with the notation. Any element $g \in G$ could be understood as a binary number.

The algorithm for solving the problem works according to the following picture.



The gate $\mathbf{U}_\rho$ is defined on the basis in a similar way to the equations (2.1),(2.2) we have seen in the proof for Deutsch's algorithm. We define $\mathbf{U}_\rho$ by equations

$$x_{\mathbf{out}} = x_{\mathbf{in}} \tag{4.3}$$
$$y_{\mathbf{out}} = \rho(x_{\mathbf{in}}) \oplus y_{\mathbf{in}}. \tag{4.4}$$

Let us see what happens when the program is executed.

1. In step (1) we prepare the input in the state $|\underbrace{00\ldots0}_{|G| \text{ times}}\rangle \otimes |\underbrace{00\ldots0}_{|C| \text{ times}}\rangle$.

2. We apply Hadamard gates on the first wire according to notation of parallel gates 1.4.4 and we get to the step (2). It can be easily shown that the result in state (2) is

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |00\ldots0\rangle. \tag{4.5}$$

The states $|g\rangle$ are corresponding to value of g (for example, $|0010\rangle$ corresponds to the element $(0,0,1,0)$).

3. The third step seems intimidating at first, but because on the second wire there are only zeros, the **xor** is trivial. We get the state:

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |\rho(g)\rangle. \tag{4.6}$$

We can rewrite the equation thanks to our definition of set T in Definition 4.1.1 and so we get

$$\frac{1}{\sqrt{|G|}} \sum_{t \in T} \sum_{h \in H} |t+h\rangle \otimes |\rho(t+h)\rangle = \frac{1}{\sqrt{|G|}} \sum_{t \in T} \sum_{h \in H} |t+h\rangle \otimes |\rho(t)\rangle, \tag{4.7}$$

because the function ρ is constant on every coset.

4. In contrast to the previous step, applying the quantum Fourier transform seems trouble-free, but here the magic happens. Remember Example 3.3.9 which shows in equation (3.61) how the matrix of the Fourier transform works. We will just rewrite it so we are sure what is going on.

4.1.2 Remark The characters of group $\mathbb{Z}_2$ have only values 1 and -1. In combination with the definition of extension of characters 3.2.7, in particular the equation (3.44), the matrix of the quantum Fourier transform has only real entries 1 and -1.

The matrix for our Fourier transform is

$$\mathbf{F} = \frac{1}{\sqrt{n}} \begin{pmatrix} \chi_1(x_1) & \chi_1(x_2) & \ldots & \chi_1(x_n) \\ \chi_2(x_1) & & & \chi_2(x_n) \\ \ldots & & & \ldots \\ \chi_n(x_1) & \chi_n(x_2) & \ldots & \chi_n(x_n) \end{pmatrix}. \tag{4.8}$$

We will just change the notation for entries of the matrix. By x_i we mean a particular $g \in G$ and we will not continue to number χ by an index i but we will use index $t+h$ corresponding to particular element of the group G. We should slow down a bit for a moment and think about how the Fourier transform work.

For a chosen $|\varphi\rangle$ the Fourier transform puts

$$|\varphi\rangle \overset{\mathbf{F}}{\mapsto} \frac{1}{\sqrt{|G|}} \sum_{g \in G} \chi_\varphi(g)|g\rangle. \tag{4.9}$$

Now we can safely apply the Fourier transform on the first wire of expression (4.7) and we get

$$\frac{1}{\sqrt{|G|}} \sum_{t \in T} \sum_{h \in H} \frac{1}{\sqrt{|G|}} \sum_{g \in G} \chi_{t+h}(g)|g\rangle \otimes |\rho(t)\rangle \tag{4.10}$$

We apply a trick that $\chi_a(b) = \chi_b(a)$ and we get

$$\frac{1}{\sqrt{|G|}} \frac{1}{\sqrt{|G|}} \sum_{t \in T} \sum_{g \in G} \chi_t(g) \sum_{h \in H} \chi_h(g)|g\rangle \otimes |\rho(t)\rangle. \tag{4.11}$$

We will continue after a break because we need the following lemma.

4.1.3 Lemma *Suppose we are given a character χ_g on a group G with subgroup H. Then either the first case or the second does hold.*

$$\sum_{h \in H} \chi_g(h) = \begin{cases} |H|, & \text{if } \chi_g \in H^\perp \\ \\ 0 & \text{else.} \end{cases}$$

Proof. The first half is a simple consequence of our definition of $H^\perp$ in (4.1). The second half is a bit more complicated, but not so much. If $\chi_g \notin H^\perp$ then there must be $h' \in H$ such that $\chi_g(h') \neq 1$. Therefore we can rewrite $\sum_{h \in H}$ by simply permuting elements as

$$\begin{aligned} \sum_{h \in H} \chi_g(h) &= \sum_{h \in H} \chi_g(h + h') \\ &= \sum_{h \in H} (\chi_g(h) \cdot \chi_g(h')) \\ &= \chi_g(h') \cdot \sum_{h \in H} \chi_g(h). \end{aligned}$$

We can rearrange both sides and we get

$$\begin{aligned} \sum_{h \in H} \chi_g(h) - \chi_g(h') \cdot \sum_{h \in H} \chi_g(h) &= 0 \\ (1 - \chi_g(h')) \cdot \sum_{h \in H} \chi_g(h) &= 0 \end{aligned}$$

We know $(1 - \chi_g(h')) \neq 0$ and therefore $\sum_{h \in H} \chi_g(h) = 0$. ■

The lemma was refreshing but the Shor's algorithm is waiting for us. We continue with rewriting expression (4.11) using 4.1.3 as

$$\frac{|H|}{|G|} \sum_{t \in T} \sum_{\{g | \chi_g \in H^\perp\}} \chi_t(g)|g\rangle \otimes |\rho(t)\rangle. \tag{4.12}$$

4.1.4 Remark Unsurprisingly we can say that values of $\chi_t(g)$ are $|\chi_t(g)| = |e^{i\alpha}|$ for some alpha. This will help when we proceed with the measurement because $|e^{i\alpha}| = 1$ and it will not affect the measurement at all.

When we apply the measurement, we get g such that $\chi_g \in H^\perp$ with probability $\frac{|H|^2}{|G|^2}|\chi_t(g)|^2 = \frac{|H|^2}{|G|^2}$. It can be shown that $|H| \cdot |H^\perp| = |G|$. We can rewrite the probability as $\frac{1}{|H^\perp|^2}$. It remains to find the $g' \in H$ which would be one of the generators. We can find $\chi_g \in H^\perp$ easily and with tools from linear algebra we can find a g' "perpendicular" to χ_g as the group $(\mathbb{Z}_2)^n$ is a linear space.

4.2 Factorization and the Shor's algorithm

In cryptography, the safety of the public key methods (RSA and the like) depends on our ability to factorize a non-prime number into product of primes p and q. [20] [15]. If only we were able to find the decomposition of the number n, the encryption is broken and Eve can read all Bob's and Alice's messages. There are ways how to breach the encryption physically [6], but we will focus on the mathematical way. Even with the advanced techniques the factorization problem is believed to be unsolvable on a standard computer in reasonable time, especially when prime numbers p and q are chosen carefully and very large.

Encryption and factorization

Disclaimer. We will not develop the theory of cryptography and we will stick to the basics. If you are interested in the topic, the book [20] explains cryptography in greater detail. Although we will just scratch the surface we will try not to omit the necessities.

Very briefly we will present the RSA algorithm. The RSA algorithm is an asymmetric algorithm which means it uses a public encryption key and private decryption key. When Alice wants to create her public and private keys, she proceeds according to the following algorithm.

1. Choose two distinct prime numbers p and q.

2. Compute $n = p \cdot q$. It can be proven that $\varphi(n) = (p-1) \cdot (q-1)$.
3. Choose an element $e \in \mathbb{Z}^{\times}_{\varphi(n)}$.
4. Compute using extended Euclid's algorithm $d = e^{-1}$ where $d \in \mathbb{Z}^{\times}_{\varphi(n)}$. Knowing e it is difficult to find d, provided you do not know $\varphi(n)$.

The public key is (N, e) and the private key is (N, d). The encryption of a message $m \in \mathbb{Z}_n$ is done by the Alice's public key by putting

$$c = m^e \in \mathbb{Z}_n$$

where c is called a cypher-text. The decryption is done by Alice's private key as

$$c^d = m^{d \cdot e} \overset{n}{\equiv} m.$$

We will not prove this statement, but you can find the proof in [20].

The security of messages is in jeopardy if someone finds the value of $\varphi(n)$ and could recreate Alice's private key. It is easy to find the value of $\varphi(n)$ if we know the prime factorization of number n.

Shor's algorithm

Firstly we will introduce the algorithm without any comments so that the steps are clear and the reader is not distracted. Then we will explain how and why the algorithm works and we will give comments step by step.

There is a given $n = p \cdot q$ where p and q are different prime numbers.

1. Randomly choose $a > 1$, where $a \in \mathbb{Z}_n$. Denote $d = \gcd(a, n)$.
2. If $d > 1$ then a is factor of n and therefore the algorithm ends.
3. If $d = 1$ then find the order of a in $\mathbb{Z}^{\times}_n$ and denote it by r.
4. If r is odd, go back to step 1.
5. If the equation $a^{r/2} + 1 \overset{n}{\equiv} 0$ does not hold the expression $\gcd(a^{r/2} + 1, n)$ or $\gcd(a^{r/2} - 1, n)$ is a nontrivial factor of n and the algorithm ends. Otherwise go to step 1.

Explanation

The algorithm shown above deserves an explanation of why all the steps make sense.

1. We choose an element $a \in \mathbb{Z}_n$ at random with uniform distribution. If we are in the first step more than once, we will exclude already chosen number. As the number n is finite this procedure ensures that the algorithm ends. For next steps we will need the greater common divisor of n and a and we will denote it d. Greatest common divisor could be found by Euclid's algorithm which has a polynomial complexity.

2. We could be extremely lucky when we chose the element a and it could be one of the divisors we were looking for. That happens exactly when the divisor d is greater than one. Unfortunately for us this does not happen very often.

3. We will try to find the order of a in the group $\mathbb{Z}_n^\times$. We have shown in Example 3.4.5 that this problem can be formulated as a hidden subgroup problem and we can solve it by quantum computers efficiently. This is exactly the part where quantum computers beat the ordinary computers. We denote the order by letter r.

4. We have successfully found the order r, that means

$$a^r \overset{n}{\equiv} 1$$

   If the order r is even we can rewrite the equation to a familiar form where factorization looks much simpler.

$$\begin{aligned} a^r - 1 &\overset{n}{\equiv} 0 \\ (a^{r/2} - 1)(a^{r/2} + 1) &\overset{n}{\equiv} 0 \\ (a^{r/2} - 1)(a^{r/2} + 1) &= k \cdot n \end{aligned}$$

5. We have figured out the number n divides $(a^{r/2} - 1)(a^{r/2} + 1)$ which should be elating. If n divides neither $(a^{r/2} - 1)$ nor $(a^{r/2} + 1)$ we almost have the factorization. Firstly n cannot divide $(a^{r/2} - 1)$ because that would mean $a^{r/2} \overset{n}{\equiv} 1$, which is in contradiction with the definition of order. Sadly n can divide $(a^{r/2} + 1)$ in some cases. Fortunately enough according to [18] and [14] we choose a fitting a with the probability greater than $1/2$. Finally we know that $(a^{r/2} + 1)$ or $(a^{r/2} - 1)$ must share a factor with the number n. We just have to find their greatest common divisor.

And here we have it, Shor's algorithm, the mighty tool Eve can use to break into Alice's and Bob's messages. We undertook a beautiful journey with Eve, we studied linear algebra in complex spaces, we learned postulates of quantum computation, we were introduced to groups, we discovered hidden subgroups and we admired the power of quantum computation. Is it not a shame to waste our fresh knowledge just on reading someone's love letters? We grew up and we can finally can have some fun on our own.

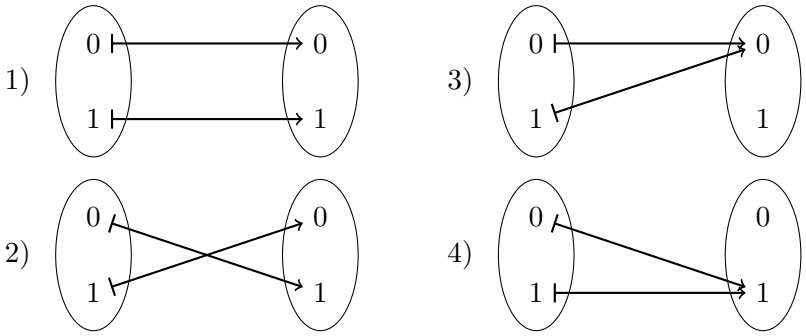
4.3 Formulation of the Deutsch's problem

We have to keep an old promise we gave to the reader at the beginning of Chapter 3. There we pledged that we would say what is hidden in the Deutsch's algorithm. After all we have seen that it seems reasonable to presume there is also a hidden subgroup in the Deutsch's problem. There indeed is one and we will discover it in this section. Finally we will answer why the Deutsch's algorithm works.

There are four possible functions f_1, f_2, f_3 and f_4 on two element set $\{0, 1\}$, namely

$$\begin{aligned} &1)\ f_1(x) = x && 3)\ f_3(x) = 0 \\ &2)\ f_2(x) = 1 - x && 4)\ f_4(x) = 1. \end{aligned}$$

We will just remind ourselves how these functions look like.



We are given one of the functions above and the task is to decide whether the function is constant or not. We already know how the decision process looks like from Chapter 2, this time we ask ourselves how we can interpret this problem as a hidden subgroup problem.

We observe that the set $\{0, 1\}$ can be interpreted as the group $\mathbb{Z}_2$. The group $\mathbb{Z}_2$ has two different subgroups.

1. The first subgroup is trivial, the group $H_1 = (\{0\}, +, -, 0)$. There are two cosets with respect to the subgroup H_1 which are $\{0\}$ and $\{1\}$.
2. The second subgroup is the whole $\mathbb{Z}_2$, which we will denote by H_2. The subgroup H_2 has only one coset which is the subgroup H_2.

The set of colours is in this case also the set $\{0, 1\}$. The function

$$f_i : \mathbb{Z}_2 \to \mathbb{Z}_2, \text{ where } i \in \{1, 2, 3, 4\} \tag{4.13}$$

fulfills the Simon's promise described in 3.4.1 and we should show that.

1. Choosing functions f_1 and f_2, which are not constant, they have different values for input 0 and for input 1. The hidden subgroup in this case is the subgroup H_1 and the cosets are $\{0\}$ and $\{1\}$. Indeed the Simon's promise is fulfilled.
2. The functions f_3 and f_4 hide the subgroup H_2 which is its only coset. It is almost trivial to say the functions satisfy the Simon's promise because they are constant (on the only coset).

If we are able to find the hidden subgroup, we also determine whether the function is constant or not. We can see the subgroup problem is in fact very easy, because the subgroups are the two simplest possible (the trivial group and the whole group). We can extend this example to a more challenging one and the Deutsch's algorithm can be extended to so-called *Deutsch-Jozsa's algorithm* [11], [18]. The Deutsch-Jozsa problem is: We have a function ρ and we can choose as an input any number between 0 and $2^n - 1$. The function ρ can be of two kinds. It can be *constant* or *balanced* (that means it gives 0 for one half of inputs and 1 for the second half of inputs). The goal is to determine whether the function ρ is constant or balanced. The Deutsch's problem is case of Deutsch-Jozsa's problem for $n = 1$. The solution to this problem is unsurprisingly almost the same we have seen in the Chapter 2 with just a multidimensional twist.

Summary

The story we began just few pages ago is at the end. Eve knows not only how to break RSA encryption but she knows much more now. The exploration of quantum computers opens doors to more previously unthinkable applications which may or may not be in cryptography.

In Chapter 1 we explored mathematics of a simple quantum system, most importantly we defined unitary matrices and tensor product. We enriched the knowledge in Chapter 3 where we were introduced to groups and subgroups. We touched the topic of characters of groups, which was necessary for development of the Fourier transform. We have written down postulates for quantum computation in Chapter 1, where one of them is the measurement postulate.

We also covered conventions for quantum computers in Chapter 1. We introduced the bra-ket notation and the computational basis. We simplified the notation of unitary matrices using quantum gates. We have demonstrated how to work with quantum gates in Deutsch's algorithm in Chapter 2. We formulated the Deutsch's problem as hidden subgroup problem by ourselves in Chapter 4 using the knowledge acquired from previous chapters.

The quantum Fourier transform from Chapter 3 is used in a general algorithm for solving hidden subgroup problem in Chapter 4. In the same chapter the reader can find Shor's algorithm for number factorization as a teaser of the application of quantum computers in more complex and less theoretical problems.

Despite the fact this thesis did not bring any new information I hope there is some place for it and it can be used for example as an introductory text for newcomers overwhelmed by quanta of much superior materials. I hope I was able to pass some the knowledge I acquired on this journey to the readers and maybe stimulate their curiosity in the topic of subgroups and quantum algorithms.

Index

Bibliography

[1] P. Aluffi, *Algebra: Chapter 0*, volume 104, American Mathematical Soc., 2009.

[2] S. Axler, *Linear algera done right*, volume 2, Springer, 2015.

[3] S. Axletr, *Measure, integration & real analysis*, Springer Nature, 2020.

[4] D. Bes, *Quantum mechanics: A modern and concise introductory course*, Springer Science & Business Media, 2012.

[5] D. Bohm, Quantum theory, Dover Publication, 1989.

[6] D. Boneh et al., Twenty years of attacks on the RSA cryptosystem, *Notices of the AMS*, 46(2):203-213, 1999.

[7] J. B. Carrel, *Groups, matrices and vector spaces*, Springer, 2017

[8] K. Conrad, Characters of finite abelian groups, *manuscript*, 2010, https://kconrad.math.uconn.edu/blurbs/grouptheory/charthy.pdf, accessed 14 May, 2021.

[9] R. Dandavati, The Fourier transform on finite groups: Theory and computation, *manuscript*, 2018, http://math.uchicago.edu/may/REU2018/REUPapers/Dandavati.pdf, accessed 14 May, 2021.

[10] R. de Wolf, *Quantum computing: Lecture notes*, arXiv preprint, 2021, https://arxiv.org/abs/1907.09415, accessed 14 May, 2021.

[11] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439 1907:553-558, 1992.

[12] A. L. Gorodentsev, *Algebra II*, Springer, 2017.

[13] B. C. Hall, *Quantum theory for mathematicians*, volume 267, Springer, 2013.

[14] M. Hirvensalo, *Quantum computing*, Springer, 1982.

[15] N. Koblitz, *A course in number theory and cryptography*, volume 114, Springer Science & Business Media, 1994.

[16] P. Kulhánek, *Vybrané Kapitoly z Teoretické Fyziky I*, Aldebaran Group for Astrophysics, 2020.

[17] J. Liesen and V. Mehrmann, *Linear algebra*, Springer, 2015.

[18] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.

[19] J. Pade, *Quantum Mechanics for Pedestrians 2 Applications and Extensions Second Edition*, Springer, 2018.

[20] N. P. Smart, *Cryptography made simple*, Springer, 2016.

[21] A. I. Kostrikin and Y. I. Manin, *Linear algebra and geometry*, CRC Press, 1989.

[22] C. P. Williams, *Explorations in quantum computing*, Springer Science & Business Media, 2010.