On Quantum Computation

Mika Hirvensalo

Turku Centre for Computer Science
TUCS Technical Report No 111
April 1997
ISBN 952-12-0001-4
ISSN 1239-1891

Contents

Introduction **1**

I. Classical models of computation **6**
- I.1 Finite automata . . . 6
- I.2 Turing machines . . . 6
- I.3 Nondeteministic automata . . . 10
- I.4 Probabilistic automata . . . 12
- I.5 On algorithmic complexity . . . 14
- I.6 On the classification of formal languages . . . 17

II. Quantum Turing machines **18**
- II.1 General overview . . . 18
- II.2 A mathematical formalism of quantum computation . . . 19
- II.3 Unitary operation on the configuration space . . . 23
- II.4 Simulation of a QTM in a finite-dimensional space . . . 28
- II.5 A simple physical model . . . 30
- II.6 The reversibility of the computation . . . 33

III. Finite quantum automata **39**
- III.1 Motivation . . . 39
- III.2 Quantum digits, registers and networks . . . 40
- III.3 Finite quantum automata . . . 41
- III.4 Quantum parallelism . . . 45
- III.5 The discrete Fourier transform . . . 46
- III.6 Shor's algorithm for factoring numbers . . . 51
- III.7 Discrete logarithm . . . 58

IV. Quantum complexity theory **65**
- IV.1 Quantum complexity classes . . . 65
- IV.2 The problem of Deutsch and Jozsa . . . 67
- IV.3 Oracle quantum computation . . . 69

V. Conclusions **73**
- V.1 Comparing quantum and classical computation . . . 73
- V.2 On possible implementations of a quantum computer . . . 74
- V.3 On an optical quantum computer . . . 75

VI. Appendix: Quantum physical background **77**
- VI.1 Quantum physical phenomena . . . 77
- VI.2 Mathematical tools . . . 78
- VI.3 Linear operators . . . 79
- VI.4 Spectral theory briefly . . . 81
- VI.5 States . . . 83

VI.6 Observables and the probability axiom 84
VI.7 The dynamical evolution of a quantum system 85
VI.8 Compound systems 86
VI.9 Measurement 88

Index **91**

References **93**

Abstract

As early as 1982 Richard P. Feynman suggested that it might be impossible to simulate quantum mechanical phenomena with a computer without an exponential slowdown in the efficiency of computation. He also proposed that this problem may be avoided by allowing the computer to run according to the laws of quantum mechanics, thus implicitely suggesting that a computer based on the quantum mechanical systems, a *quantum computer*, might have exponentially more computational efficiency than any classical one.

A quantum mechanical phenomenon that never occurs in classical physics and which actually makes quantum computation interesting and powerful, is the *superposition* of states, which, loosely speaking, means that instead of being totally in one single state, a particle can be "in between many states". Compare this to the elements of classical computer, bits. A bit takes values zero or one, but never anything between. In a quantum computer, one is able to store *both values* simultaneously in one quantum bit, equally weighted or not. Moreover, in the classical computer one can store a number from zero to $2^n - 1$ in a register of length n, whereas, in quantum computation, all the values could be stored simultaneously. The computation, which is performed via unitary transformations in the state vector space, then applies to all these values simultaneously. Thus the power of the quantum computation lies not in the absolute speed of the hypothetetical quantum computer but in the possibility to actually follow many computational path simultaneously, as a nondeterministic automaton does. Alas, the observation of the outcome will yield only one value destroying the rest irreversibly. However, using a procedure smart enough one could obtain some valuable information about the computation.

Keywords: Quantum computation, Unitary mapping, Superposition, Turing machine, Reversible computation, Computational Complexity, Factoring

TUCS Research Group
Theory Group: Mathematical Structures in Computer Science

Introduction

In 1982 R. Feynman pointed out in [14] that it appears to be impossible to simulate quantum physics with computers without an exponential slowdown. The reason to believe this is quite evident: If one, for instance, thinks about m interactive two-state quantum systems, the simulation of the compound system will be difficult since the compound system has 2^m states. Feynman also suggested that this difficulty may be avoided by allowing the computer to run according to the laws of quantum physics, thus introducing the idea of a *quantum computer* for the first time.

Classical physics is a deterministic theory: Any future position and momentum of a single body can be predicted, provided the initial state of the system is known well enough. Quantum mechanical systems obey different laws. Quantum mechanics is a *deterministically probabilistic* theory, meaning that although the behaviour of a single microsystem cannot be predicted, the *probability distribution* of the "future behaviour" will be uniquely determined.

The representation of quantum physics chosen for this thesis is the so called *Hilbert space formalism*, which seems to be especially well suited to the purposes of quantum computation. In this formalism, a physical system is associated to a Hilbert space, which, due to the purposes of quantum computation, will here be usually finite-dimensional. The *states* of the physical system are represented as unit-trace self-adjoint linear mappings (which are identified with unit-length vectors, so called *wave functions*) and the *observables* are represented as self-adjoint mappings. The next example is to give an illustration of this Hilbert space formalism:

A bit in a computer is implemented by two recognized states of voltages. A voltage around 0 V is interpreted as zero, and a voltage around 5 V represents one. As an analogue, one may concentrate on two-state quantum systems. For example, an atom may have a lower and a higher electron energy state, a photon may be polarized horisontally or vertically, etc. In this thesis we will not pay attention to the actual nature of the two-state systems, but the essential point is that such a system, a *quantum bit*, is associated to a two-dimensional complex Hilbert space $\mathcal{H}_2$. The inner product of vectors φ and ψ is denoted by $\langle \varphi \mid \psi \rangle$. The states of the system are in this thesis represented as unit-length vectors in $\mathcal{H}_2$. Actually, in quantum physics, these "vector states" are not sufficient to describe the behaviour of quantum systems. In general, the states are represented as unit-trace linear mappings. To illustrate how those mappings are identified with "vector states" in the Hilbert space, suppose that ψ is a unit-length vector in $\mathcal{H}_2$. Then the mapping $P[\psi]$ defined by $P[\psi]\eta = \langle \eta \mid \psi \rangle \psi$ is the projection onto the subspace spanned by ψ. Mapping $P[\psi]$ is now an example of a state, indeed, projections onto one-dimensional subspaces are called *pure* states. The other states are *mixed*. Usually we suppose that $\mathcal{H}_2$ has a fixed orthonormal basis

$\{|0\rangle, |1\rangle\}$ which we call a *computational basis* (the "ket"-notation $|\varphi\rangle \in \mathcal{H}_2$ is due to P. Dirac and becomes more understandable in the appendix). The element $|0\rangle$ should not be confused with the zero vector. We will say that $P[|0\rangle]$ and $P[|1\rangle]$ are *basis states* or *classical states* and all the other pure states are *superpositions* of the classical states. Now the name "two-state system" becomes justified: There are essentially two classical pure states (when the chosen computational basis is fixed). Space $\mathcal{H}_2$ is then called the *space of state vectors*. In this thesis, only pure states are considered, and such a state can be characterized by a single vector of unit length that is chosen to represent the projection subspace. For instance, instead of state $P[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)]$ we shall speak about a (vector) state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Of course the vector representative of a state is not unique, but all vectors of form $e^{i\theta}|\varphi\rangle$ with real θ represent the same state as $|\varphi\rangle$ does.

Observables, like momentum and position, are represented as self-adjoint mappings. In the finite-dimensional case the set of possible values of an observable equals the set of the eigenvalues of the observable, and in the infinite-dimensional case the possible values is exactly the spectrum of the observable. It is a well-known fact that the spectrum of a self-adjoint mapping is real, so an observable A and a state T together determine a measure on real numbers by

$$E_T^A(X) = \mathrm{Tr}(E^A(X)T),$$

where $E^A(X)$ is the *spectral projection* of A on set X (see appendix). Consider, for instance, an observable $A = \lambda_1 P[|0\rangle] + \lambda_2 P[|1\rangle]$. Then the measurement of observable A can yield either λ_1 or λ_2. The spectral projections are given by $E^A(\{0\}) = P[|0\rangle]$ and $E^A(\{1\}) = P[|1\rangle]$. In the Hilbert space formalism, the value $E_T^A(X)$ is interpreted to be the *probability* that the measurement of observable A in state T will yield a result in set $X \subseteq \mathbb{R}$. From this measurement scheme we can derive a simpler and more suitable concept for quantum computation, the *observation*. The idea is to choose a suitable observable for recognizing the *state* of the system. Some details are explained in the appendix, but the observation procedure will work as follows:

Suppose that the system is in a state represented by state vector $\alpha_1|0\rangle + \alpha_2|1\rangle$. Then the observation of the system yields $|0\rangle$ or $|1\rangle$, with probabilities $|\alpha_1|^2$ and $|\alpha_2|^2$ respectively (note that $|\alpha_1|^2 + |\alpha_2|^2 = 1$, since state vectors are of unit length).

The generalization of two-state systems into systems of arbitrary many states is straightforward, but now we will take a look into combining two-state systems, quantum bits, into *quantum registers*: Consider m two-state systems. Under certain assumptions of the physical nature of these systems (see [18] for instance), the Hilbert space associated to the entire system will be the m-fold tensor product

$$\mathcal{H}_{2^m} = \mathcal{H}_2 \otimes \mathcal{H}_2 \otimes \ldots \otimes \mathcal{H}_2.$$

Tensor product above is, by the definition, complex vector space whose basis consists of vectors $|\,x_1\rangle \otimes \ldots \otimes |\,x_m\rangle$ where sequence $x_1 \ldots x_m$ runs over all binary strings of length m. A reader not aware of tensor products can understand the notation $|\,x_1\rangle \otimes \ldots \otimes |\,x_m\rangle$ here as a formal product, where $\otimes$ is associative and distributive, but not commutative product. Clearly $\mathcal{H}_{2^m}$ is the complex vector space of dimension 2^m, so we can introduce a shorter notation

$$|\,x_1 \ldots x_m\rangle = |\,x_1\rangle \otimes \ldots \otimes |\,x_m\rangle.$$

The inner product in the space $\mathcal{H}_{2^m}$ is defined by

$$\langle |\,x_1 \ldots x_m\rangle \,||\, y_1 \ldots y_m\rangle\rangle = \langle |\,x_1\rangle \,||\, y_1\rangle\rangle \cdot \ldots \cdot \langle |\,x_m\rangle \,||\, y_m\rangle\rangle,$$

so the basis is orthonormal. It is natural to denote the basis vectors by $|\,0\rangle$, $|\,1\rangle$, $|\,2\rangle$, $\ldots$, $|\,2^m-1\rangle$. Space $\mathcal{H}_{2^m}$ is in this thesis usually referred to as the *binary quantum register of length m.*

So far we have outlined how the quantum information is represented. Now we would like to compute with this information. For this purpose, we will take a quick look at the time evolution of a quantum system. According to the so-called *Schrödinger picture* of quantum mechanics, a pure state represented by a state vector $|\,\varphi\rangle$ evolves via unitary mappings,

$$|\,\varphi(t)\rangle = U(t)\;|\,\varphi(0)\rangle.$$

Here, for each $t \in \mathbb{R}$, $U(t)$ is a unitary (see appendix) mapping in the space of state vectors. In quantum computation we aim at developing such a sequence of unitary mappings U_1, U_2, $\ldots$, U_n that in the beginning of the computation the state of the system is $|\,\varphi\rangle$, after the first step $U_1\,|\,\varphi\rangle$, after the second $U_2U_1\,|\,\varphi\rangle$, and so on. Finally, the state will be $U_n \ldots U_2U_1\,|\,\varphi\rangle$. Here $U_i = U(t_i)$ and $t_1 < t_2 < \ldots < t_n$ is a finite sequence of moments in time, so the computation that we consider goes through discrete time steps (as an analogue, think about an ordinary computer: A computer is thought to perform its action on discrete steps determined by an internal clock, but also the intemediate time between two consecutive steps is needed for some technical reasons). As an example, consider a quantum bit with the time evolution determined, in the beginning, by a unitary matrix

$$A = \frac{1}{2}\begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix}$$

(in basis $|\,0\rangle = (1,0)^T$ and $|\,1\rangle = (0,1)^T$). If the initial state was $|\,0\rangle$, it is easy to see that after the first step of the computation we will have

$$A\,|\,0\rangle = (\frac{1}{2} - \frac{i}{2})\;|\,0\rangle + (\frac{1}{2} + \frac{i}{2})\;|\,1\rangle.$$

If we now stopped the "computation" and observed the state $A\;|\;0\rangle$, we would get $|\;0\rangle$ or $|\;1\rangle$ both with probability $\frac{1}{2}$. The same result will come when starting with $|1\rangle$, since $A\,|1\rangle = (\frac{1}{2}+\frac{i}{2})\,|0\rangle + (\frac{1}{2}-\frac{i}{2})\,|1\rangle$ Let us continue the computation with time evolution A. Because $A^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, we have $A^2\,|0\rangle = |1\rangle$ ($|1\rangle$ will be observed with probability 1) and $A^2\,|1\rangle = |0\rangle$ ($|0\rangle$ will be observed). Thus we can consider A as a unary quantum gate representing the "square root of not".

Suppose now that there is a sequence of unitary mappings such that their product U actually computes a function that we are interested in:

$$U\,|x_{m-1}\rangle \otimes |x_{m-2}\rangle \otimes \ldots \otimes |x_0\rangle = |f(x_{m-1}2^{m-1} + x_{m-2}2^{m-2} + \ldots + x_0)\rangle.$$

If we then, instead of a basis state $|x\rangle$ could begin with a superposition

$$\varphi = \frac{1}{\sqrt{2^m}}\left(|0\rangle + |1\rangle + |2\rangle \ldots + |2^m - 1\rangle\right),$$

applying time evolution U we would obtain

$$U\varphi = \frac{1}{\sqrt{2^m}}\left(|f(0)\rangle + |f(1)\rangle + |f(2)\rangle + \ldots + |f(2^m - 1)\rangle\right),$$

where *all the values of* f on set $\{0, 1, 2, \ldots, 2^m - 1\}$ are computed simultaneously. In order to get a reasonable concept of time complexity and to get a realization of a quantum computer more obvious, we will require that the unitary mappings that produce U should act only on small subspaces of $\mathcal{H}_{2^m}$. This question will be studied in more details later.

The ability to compute on superpositions as well as in the classical states is exactly what the power of the quantum computation will be based on: A quantum computer would be able to follow exponentially many computational paths simultaneously. For instance, one could compare the described quantum computation of f to a nondeterministic Turing machine that in the beginning takes m steps such that in each step the computation divides into two branches, where the machine writes a bit 0 on the string, in one branch, and bit 1 in the other. After those m steps, all possible integers from 0 to $2^m - 1$ are written on the tape, and the function f is computed.

However, there is one "unwanted" feature in the quantum computation. Although exponentially many possible computational paths could be exhausted in the same time as one path takes, only one outcome can be seen and the information about the other paths will be irreversibly destroyed. This is a corollary from the fact that when a quantum system in the superposition is measured, the measurement disturbs the original system in a way that we presently do not completely understand. The experiment tells us that no superposition is ever observed, but we see only classical states. Moreover, if the observation procedure is repeated, one will not observe a classical state different from the one that has been seen in the first observation. Therefore, it has been inferred that in the observation procedure the wave function (state) *collapses into a classical state*. This however implies that the time evolution of the compound system formed of the system to be measured and of the measuring apparatus can not be unitary, since the (physical) information about the superposition is lost. This contradicts a fundamental structure of quantum mechanics: The unitary evolution must always be reversible. The problem of quantum measurement has not been resolved completely so far, but in this thesis we will adopt a naive interpretation of "collapsing states". This is formalized in the appendix as the projection postulate. However it should be emphasized that in those algorithms we will study, the projection postulate is *not actually needed*, since we are not interested of the outcome of observation in the middle of the computation but only when the computation ends. Anyway, looking at the superposition produced by a quantum computer irreducibly destroys information, but using a procedure smart enough, valuable information can be found faster than using known classical procedures.

In 1985 D. Deutsch was the first to formalize the concept of quantum computation [9]. He defined both quantum Turing machines and quantum circuits and studied some of their properties. However, quantum computation remained quite a marginal phenomenon in the theory of computing until 1994, when Peter W. Shor discovered his famous algorithms. In [22], Shor introduced polynomial time quantum algorithms for factoring integers and for discrete logarithm, tasks that are thought to be classically intractable. The importance of factoring and discrete logarithm lies in the field of cryptography: The security and reliability of many public-key cryptosystems and protocols are based on the assumed intractability of these two tasks. For instance, a polynomial time algorithm for factoring integers would remarkably threaten the security of the famous RSA cryptosystem [24, ch. 4].

The first chapter deals with the basic concepts. Some nonstandard formalism, which is generalized in the second chapter for the quantum computation is introduced. The second chapter offers a mathematical formalism of quantum computation. The definition of a quantum Turing machine will be given and some basic properties will be studied. In the end of the chapter, a more profound question occurs, the reversibility of classical computations. Since quantum computation is inherently reversible, in order to simulate classical computers with quantum ones, one has first to show that computation can be forced to be reversible. In this context, the result of Bennett in [1] is generalized to suit to the quantum computation. In chapter three a more "sophisticated" model of quantum computation is studied. Also some applications, such as the famous algorithms of Shor are introduced and put into more precise form. The fourth chapter is a collection of known results in quantum complexity theory, also the famous problem of Deutsch and Jozsa is studied. In the fifth chapter we compare classical and quantum computation. Also some possible implementations of quantum computers are studied. The last chapter is actually an appendix: The Hilbert space formalism of the quantum physics is briefly outlined. To read the thesis, it is not necessary to know quantum physics at all, but a look at the appendix may be recommended.

I. Classical models of computation

I.1. Finite automata

Classical models of computation are mathematical concepts that depict physical computations possible in a discrete, " Laplacean", world. One of these models is the *finite automaton.*

Definition I.1.1. A *Deterministic finite automaton* $\mathcal{A}$ is a quintuple

$$\mathcal{A} = (Q, \Sigma, \delta, q_0, F),$$

where Q is a finite set of *states*, Σ is a finite set called *alphabet*, δ is the *transition function* $\delta : Q \times \Sigma \to Q$, $q_0 \in Q$ is the *initial state* and $F \subseteq Q$ is the set of *final states.*

Deterministic finite automata (DFA for short) are often illustrated as *labelled directed graphs.* The vertices of the graph are the states of the automaton, and there is a labelled directed edge $p \xrightarrow{\sigma} q$ if and only if $\delta(p, \sigma) = q$.

Finite automata can intuitively thought to be recognizers: Given any finite word, i.e. sequence of elements (letters) of the alphabet, the device follows letter by letter the edges of the graph determined by δ beginning at the vertex corresponding to the initial state. If the computation ends in a vertex that corresponds to a final state, then the device is said to *accept* the given word, otherwise to *reject* that.

The definition of a finite automaton given here is purely abstract and mathematical, but if the set of the states is small, there is no question that one can really construct a physical device to perform the computation that this automaton does.

I.2. Turing machines

One way to measure the computational power of a DFA is to observe the family of *formal languages* recognized by DFAs. This family is called *the family of rational languages* (abbreviation **RAT**). One can of course question whether there are some devices more powerful in the sense that the family of languages recognized by them will be strictly greater than rational languages. The answer is well known to be positive, for instance, *Turing machines* are strictly more powerful.

Definition I.2.1. A *Turing machine*, $\mathcal{M}$ consists of a quintuple

$$\mathcal{M} = (Q, \Sigma, \delta, q_0, F),$$

and of a *tape*.

In the definition above Σ is a finite alphabet that is supposed to contain a special *blank symbol* $*$, Q is a finite set of *states* such that $\Sigma \cap Q = \emptyset$, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, 0, R\}$ is the completely specified *transition function*, $q_0 \in Q$ is the *initial state* and $F \subseteq Q$ is the set of *final states*. The set F is sometimes expected to contain two special states: State q_y, the "accepting state" and q_n, the "rejecting" state.

Definition I.2.2. The *tape* of a Turing machine $\mathcal{M}$ is the set $\mathcal{T}$ of all such mappings $T : \mathbb{Z} \to \Sigma \cup Q$ that $T(i) \neq *$ only for finitely many $i \in \mathbb{Z}$ and $T(k) \in Q$ for exactly one $k \in \mathbb{Z}$. For any $T \in \mathcal{T}$, $T(k)$ is the *contents* of the k:th cell. If $T(k) \in Q$, then $T(k+1)$ is called the *currently scanned symbol.*

Intuitively the tape of a Turing machine represents a sequence of memory cells that continues infinitely in both directions and each cell has a unique (integer) address. In each cell one can store a symbol chosen from $\Sigma \cup Q$.

The precise definition of the tape may sound unnecessarily ceremonial, but it will become convinient later in the chapter II.

Definition I.2.3. A *configuration of a Turing machine* is a mapping $T \in \mathcal{T}$.

Definition I.2.4. *Computation* of a Turing machine is a countable sequence of mappings in $\mathcal{T}$. Each computation is fully determined by the first member, say T_0 (which depends on the *input word*), and of the transition function. Configurations in the computation are defined as follows:

Initially symbol q_0 is stored in cell 0, ($T_0(0) = q_0$), and the letters of input word $x = \sigma_1\sigma_2 \ldots \sigma_m$ are stored in cells 1, 2, ..., m, i.e.

$$T_0(1) = \sigma_1,\ T_0(2) = \sigma_2, \ldots, T_0(m) = \sigma_m,$$

and all the other cells contain $*$. The unique element from Q on the tape works as the read-write head position and the current state marker simultaneuosly.

Suppose now that the mappings $T_0, T_1, \ldots, T_i$ of the computation path C are defined. By the definition of configuration, there is exactly one integer, say k, such that $T_i(k) \in Q$. We denote $T_i(k-1) = \sigma_1'$, $T_i(k) = q_1$ and $T_i(k+1) = \sigma_1$.

Case 1°. If $\delta(q_1, \sigma_1) = (q_2, \sigma_2, L)$, then T_{i+1} is defined by

$$\begin{aligned} T_{i+1}(k-1) &= q_2, \\ T_{i+1}(k) &= \sigma_1' \\ T_{i+1}(k+1) &= \sigma_2 \text{ and} \\ T_{i+1}(l) &= T_i(l) \text{ for all other values of } l. \end{aligned}$$

Thus the sequence $\sigma_1' q_1 \sigma_1$ on the tape at stage i is transformed into $q_2 \sigma_1' \sigma_2$ and all the other cells are unchanged.

Case 2°. If $\delta(q_1, \sigma_1) = (q_2, \sigma_2, R)$, then

$$\begin{aligned} T_{i+1}(k) &= \sigma_2, \\ T_{i+1}(k+1) &= q_2 \text{ and} \\ T_{i+1}(l) &= T_i(l) \text{ for all other values of } l. \end{aligned}$$

In this case, $q_1\sigma_1$ is transformed into $\sigma_2 q_2$ and the rest of the cells are unchanged.

Case 3°. If $\delta(q_1, \sigma_1) = (q_2, \sigma_2, 0)$, then

$$\begin{aligned} T_{i+1}(k) &= q_2, \\ T_{i+1}(k+1) &= \sigma_2 \text{ and} \\ T_{i+1}(l) &= T_i(l) \text{ for all other values of } l. \end{aligned}$$

In this third case $q_1\sigma_1$ transforms into $q_2\sigma_2$ and all the other cells remain unchanged.

Definition I.2.5. Let $\mathcal{M}$ be a Turing machine and

$$\mathcal{M}(x): \quad T_0 \longrightarrow T_1 \longrightarrow T_2 \longrightarrow T_3 \longrightarrow \ldots$$

a computation of $\mathcal{M}$ with input x. If there does not exist an i such that $T_i(k) \in F$ for some k, we say that the computation of $\mathcal{M}$ *does not halt* on input x and denote $\mathcal{M}(x) \uparrow$. Otherwise, let i be the smallest number such that $T_i(k) \in F$ for some k. Then we say that the computation of $\mathcal{M}$ *halts at stage* i.

The computation is formally described above in cases 1°, 2° and 3°, but intuitively computation goes as follows: The finite input string $x \in \Sigma^*$ is written on the tape, one symbol in one square. The read-write head marked by the unique element from Q on the tape is initially set to point at the first symbol of the input x, and the machine is set to the initial state q_0. The action of the machine is *local* in the sense that at each stage the behaviour of the automaton depends only on the value $\delta(q_1, \sigma_1)$, where σ_1 is the currently scanned symbol on the tape and q_1 is the current state of the machine. Let us mark $\delta(q_1, \sigma_1,) = (q_2, \sigma_2, d)$. Then σ_1 is replaced with σ_2, machine enters in state q_2, and the read-write -head moves one step to the left, remains unmoved or moves to the right depending on the possible values L, 0 or R of d respectively. Note that here we suppose that the transition function δ is *completely* defined for all possible pairs in $Q \times \Sigma$, even for such pairs (q, σ), where q is a halting state. In the formal definition the computational path continues infinitely, but it is thought that a final state is a "flag" that indicates the end of computation, so the result is read immediately after a final state is reached first time.

Definition I.2.6. Configuration T_1 *yields* T_2 *in* k *steps* in Turing machine M, if there is a computational path of length k developed according to rules

1°, 2° and 3° starting from T_1 and ending at T_2. If this happens, we denote $T_1 \vdash^k T_2$. Configuration T_1 *yields* T_2, denoted by $T_1 \vdash^* T_2$, if T_1 yields T_2 in k steps for some $k \geq 0$.

Turing machines (TM for short) can also be thought to be recognizers: A word is written on the tape and the computation is carried out. If the machine halts in the accepting state, we say that the machine has accepted that word. Otherwise the word is rejected. The set of all words accepted by a TM $\mathcal{M}$ is called the *formal language accepted by* $\mathcal{M}$. Note that the computation of a Turing machine need not halt at all. The family of formal languages that is accepted by TM's is called the *family of recursively enumerable languages* (abbreviation **RE**). Clearly "acceptance by a Turing machine" is not a satisfactory algorithmic concept; if L will be accepted by a Turing machine, then the computation on any $x \in L$ will end and answer $x \in L$ will be given, but one can not predict a priori how long the accepting computation will last, so a negative result $x \notin L$ cannot be seen.

A more suitable model of an algorithm is given by *always halting Turing machines*: If for any input $x \in L$, TM $\mathcal{M}$ halts in the accepting state and for any $x \notin L$ in the rejecting state, we say that $\mathcal{M}$ *decides* language L. The family of languages that are decided by Turing machines satisfying this condition is called *the family of recursive languages* (abbreviation **REC**). If there is no danger of confusion, we will use word *recognize* to mean both accepting and deciding.

On the other hand, Turing machines can also compute functions: Suppose that we are given a Turing machine $\mathcal{M}$ and an input string x is written on the tape. Furthermore, suppose that the computation ends in a final state and $y = \sigma_1 \ldots \sigma_n$ is the maximal string remaining on the tape with no blank symbols at ends and symbol $q_f \in F$ omitted. Then we denote $\mathcal{M}(x) = y$ and say that $\mathcal{M}$ carries out the computation $x \to y$. A Turing machine $\mathcal{M}$ thus determines a (partially) defined function $f_{\mathcal{M}} : \Sigma^* \to \Sigma^*$ whose domain consists of those inputs for which the machine halts. The functions that can be defined by always halting Turing machines are called *(total) recursive functions*.

There is another thing worth noticing: In addition to the final result of computation, *it is assumed that information of the state and the contents of the tape are available at every step of the computation.* This classical feature will no longer hold when studying the quantum Turing machines in the chapter II.

Comparing the power of a Turing machine to an ordinary computer, one sees that a Turing machine is a more powerful device. This is an immediate consequence of the fact that a Turing machine makes use of *potentially infinite tape*, while all physical computers must rely on finite but maybe very large memory. In fact, one also sees easily that the computational power of an ordinary computer with no incoming information is exactly that one of a deterministic finite automaton (with a large number of states).

It is also clear, due to the unlimited space resources, that no counterpart of a Turing machine exists in the physical world, so a Turing machine must be considered as a purely mathematical device. On the other hand, a physical device with a large memory may simulate a Turing machine for all inputs of "reasonable" size.

One can still ask whether the computational power of a Turing macine reaches the intuitive idea of an algorithm, i.e. a finite set of rules to handle a given input. Presently it is strongly believed that this really happens:

Church-Turing's Thesis. *The Turing machine is the mathematical counterpart of the intuitive notion of an algorithm.*

No counter-example has ever been found to contradict this thesis, but for the sake of the nature of the thesis it shall remain unproven: In order to prove it, we would have to know precicely what is an "intuitive algorithm". But to know it precisely is to lose the term "intuitive".

I.3. Nondeterministic automata

The concept of an automaton can be extended to include nondeterministic devices. A *nondeterministic finite automaton* (NFA) is defined in the same fashion as the deterministic one, but instead of the transition function there will be a *transition relation* $\delta \subseteq Q \times \Sigma \times Q$. Thus reading a sequence of letters the automaton can follow many *computation paths*, and this can be illustrated by a *computation tree* (this will be discussed later in more detail). The action of this automaton is now explained as follows: NFA accepts an input x, if there is *at least* one computational path $p \xrightarrow{x} q$, where p is an initial state and q a final state.

Definition I.3.1. A nondeterministic Turing machine, NTM, is a quintuple

$$(Q, \Sigma, \delta, q_0, F).$$

Components Σ, Q, q_0 and F and the tape are exactly same as in the definition of an ordinary Turing machine, but

$$\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{L, 0, R\}$$

is the *transition relation.*

Definition I.3.2. The *computation tree* C of a nondeterministic Turing machine is a labelled tree whose entries are mappings in $\mathcal{T}$. The root entry is determined by the input word x in the same fashion as the initial configuration was determined for the ordinary Turing machine and the descendants of each vertex are determined by the transition relation as explained below:

Let us take any vertex v with entry $T \in C$. Suppose that $T(k) = q_1 \in Q$ and $T(k+1) = \sigma_1$. The descendants of T are determined as follows:

For any pair $(q, \sigma) \in Q \times \Sigma$ satisfying $(q_1, \sigma_1, q, \sigma, L) \in \delta$ there is a descendant of v whose entry is defined as in case 1° in the previous section, for $(q_1, \sigma_1, q, \sigma, R) \in \delta$ there is a descendant as in case 2° and for $(q_1, \sigma_1, q, \sigma, 0) \in \delta$ there is a descendant as in 3°. Those vertices where the entry satisfies $T(k) \in F$ for some integer k are called *halting vertices*. Intuitively we can say that if machine is in state q_1 scanning symbol σ_1, then $(q_1, \sigma_1, q_2, \sigma_2, d) \in \delta$ determines that there is a *possibility* of entering state q_2, writing σ_2 over σ_1 and moving to direction d.

Definition I.3.3. A vertex v is at *level* i, if the length of the path from the root to v is of length i. Note that there is exactly one such path, since C is a tree. Since C is finitely branching, the number of vertices at each level i is finite.

Definition I.3.4. Configuration T_1 *yields* T_2 *in* k *steps* if one can develop a computation tree with root entry T_1 and entry T_2 in level k. Relation "yields" is defined in an obvious way. Notations $T_1 \vdash^k T_2$ and $T_1 \vdash^* T_2$ are used analogously.

Definition I.3.5. Let C be a computation tree with input x and L the set of entries of halting vertices of C. Set L is also called the set of *final configurations*. We say that computation C *halts* if and only if $L \neq \emptyset$.

If the transition relation is a function, i.e. for each pair (σ_1, q_1) there exists exactly one triplet (q_2, σ_2, d), where d is either L, 0 or R, the computation consists of one single path. We can then construct a (deterministic) Turing machine with all components equal to the original nondeterministic one, but transition function δ' will be defined as

$$\delta'(q, \sigma) = (q_1, \sigma_1, d) \text{ if and only if } (q, \sigma, q_1, \sigma_1, d) \in \delta.$$

It is clear that the computation path of this deterministic machine is exactly the same as the computation of the original nondeterministic one, thus the classical Turing machine can be regarded as a special case of NTM.

It is clear that nondeterministic automata in general do not correspond to any realistic model of computation, but one can always simulate a nondeterministic device by a deterministic one by exhausting the computation tree level by level. Thus the *computational power of a nondeterministic Turing machine is not greater than that one of the deterministic device*, but since the number of possible configurations may grow exponentially as a function of the level, the *simulation of nondeterministic computation may cause an exponential slowdown* in the efficiency.

One can then ask whether there exists any physical process that can simulate the computation of a nondeterministic device in a natural way. The answer is partially affirmative: Devices that are intuitively very closely related to nondeterministic ones, or to probabilistic ones, could be in principle built in quantum physical systems.

I.4. Probabilistic automata

One could also let the automaton "toss a coin" during the computation. This intuitive idea gives rise the following formal definition:

Definition I.4.1. A *finite probabilistic automaton* is defined in the same way as the deterministic one, but instead of the ordinary transition function there will be a transition function

$$\delta : Q \times \Sigma \times Q \to [0, 1]$$

such that

$$\sum_{q \in Q} \delta(q_1, \sigma_1, q) = 1$$

for any pair $(q_1, \sigma_1) \in Q \times \Sigma$. Assume that the automaton is in state q reading symbol σ. Then $\delta(q_1, \sigma, q_2)$ gives the *probability* that the automaton enters the state q_2. A *computation tree* edges labelled with the corresponding probabilities can be drawn in the obvious way. In each step of computation the automaton is in a *linear superposition* (defined later) of all possible configurations, each configuration is equipped with the coefficient that corresponds the probability of reaching that configuration. Moreover, it is required that the sum of all coefficients in the linear superposition equals to one.

Definition I.4.2. The probabilistic Turing machine, PTM, is a quintuple

$$(Q, \Sigma, \delta, q_0, F).$$

All the other components are same as before, but

$$\delta : Q \times \Sigma \times Q \times \Sigma \times \{L, 0, R\} \to [0, 1]$$

is the *transition probability function*. It is required that for each pair $(q_1, \sigma_1) \in Q \times \Sigma$

$$\sum_{\substack{(q,\sigma) \in Q \times \Sigma \\ d \in \{L,0,R\}}} \delta(q_1, \sigma_1, q, \sigma, d) = 1. \tag{1-1}$$

If a PTM is in the state q_1 scanning symbol σ_1, then intuitively $\delta(q_1, \sigma_1, q_2, \sigma_2, d)$ gives the probability of entering state q_2, overwriting σ_1 with σ_2 and moving according to d. It should be emphasized here that for each input a PTM determines the *probability distribution* of possible computations.

Definition I.4.3. The *computation* or the *computation tree* C of a probabilistic Turing macine is a labelled tree with entries in $\mathcal{T}$. The root entry of C is defined as in the nondeterministic Turing machine, and the labels of edges and the descendants of each vertex are defined as follows: Take an entry T and suppose that $T(k) = q_1 \in Q$. Denote $T(k+1) = \sigma_1$. For each pair $(q, \sigma) \in Q \times \Sigma$ there is a descendant with entry T_d if and only if $\delta(q_1, \sigma_1, q, \sigma, d) \neq 0$. The entry T_d is then defined as in the nondeterministic computation, and the edge to that descendant is labelled with number $\delta(q_1, \sigma_1, q, \sigma, d)$. Again vertices with entries that satisfy $T(k) \in F$ for some integer k are called *halting vertices*. Note that the entries of different vertices of C may coincide. What is meant by the *acceptance* of a probabilistic Turing machine will be studied in the chapter IV.

Definition I.4.4. The computation C *halts*, if there is at least one halting vertex in C. The *probability of a vertex v* is defined to be the product of all labels from the root to v (again the path mentioned is unique, C being a tree).

From the requirement (1-1) it follows easily by induction that the sum of probabilities of all vertices at level i equals one at each level.

Definition I.4.5. The configuration T_1 yields T_2 in k steps with probability p, if there is a computation tree, developed as described above, with root entry T_1 and a vertex of probability p with entry T_2 at level k. This situation is denoted by $T_1 \vdash_p^k T_2$. The relation "yields" is defined in the obvious way.

We could take reals instead of the complex numbers in the following definition, but then the definition would not apply directly to chapter II.

Definition I.4.6. Let $\mathcal{C}$ be the linear span of set $\mathcal{T}$ over complex numbers. Then obviously $\mathcal{C}$ becomes a vector space over complex numbers and each element in $\mathcal{C}$ can be represented as a finite sum

$$\sum_{i=1}^{n} a_i T_{j_i}.$$

Vector space $\mathcal{C}$ is called the *configuration space.*

Definition I.4.7. Let i be a level in the computation tree C, v_1, v_2, ..., v_m all vertices at level i, p_1, p_2, ..., p_m the probabilities of those vertices and T_1, T_2, ..., T_m the corresponding configurations. Element

$$C_i = p_1 T_1 + p_2 T_2 + \ldots + p_m T_m \in \mathcal{C}$$

that satisfies $p_1 + p_2 + \ldots + p_m = 1$ is called *the superposition of configurations at stage i* or *at step i* (this will be slightly different in chapter II). Those elements of the configuration space are either configurations or superpositions of configurations. This concept will become more important in chapter II in the definition of quantum Turing machine.

Definition I.4.8. Let the superposition of configurations at stage i be

$$C_i = p_{i_1} T_{i_1} + p_{i_2} T_{i_2} + \ldots + p_{i_n} T_{i_n} \tag{1-2}$$

written in such a way that $T_{i_k} \neq T_{i_l}$ in the representation above, when $k \neq l$. The *probability of reaching a configuration* T *at level* i, $p(T, i)$, is p_{i_k}, if $T = T_{i_k}$ for some k in (1-2), 0 otherwise. The *probability of reaching a final configuration* T is the sum of probabilities of halting vertices with entry T.

The idea of configuration space is quite useful and intuitively clear: In the computation tree it may happen that several entries at the same level coincide, so the probabilities of the corresponding vertices *all* contribute to the probability of that configuration. But in the configuration space one gets the essential information about the probabilities in the canonical basis vector representation (1-2). We can thus say that *the sequnece of superpositions of configurations at level* 1, 2, ... *reveals essential features of the computation of a probabilistic Turing machine.*

If for each pair (q_1, σ_1) there is exactly one triplet (q_2, σ_2, d) such that the probability $\delta(q_1, \sigma_1, q_2, \sigma_2, d)$ equals to 1 and for any other triplet (q', σ', d') always we have $\delta(q_1, \sigma_1, q', \sigma', d') = 0$, then the computation of C also consists of one single path with no branches. If the computation halts, then the probability of final configuration is 1. In this case, we can construct an ordinary Turing machine beginning with the probabilistic one in an analogous way as we did in the end of section I.3, where we begun with a nondeterministic machine. Therefore also PTM's can be considered as extensions of TM's. But even more: Suppose that we have a probabilistic Turing machine. Then we can turn it into a nondeterministic machine by defining the transition relation δ' by

$$(q_1, \sigma_1, q_2, \sigma_2, d) \in \delta' \text{ if and only if } \delta(q_1, \sigma_1, q_2, \sigma_2, d) \neq 0.$$

It is clear that nondeterministic automaton gained in this fashion halts if and only if the original one did. Thus we can consider a nondeterministic Turing machine as a loose form of a probabilistic one: A nondeterministic automaton just "ignores" the probability distribution.

I.5. On algorithmic complexity

An *algorithm* means here a Turing machine that always halts. An input word of $\mathcal{A}$ is breafly called the input. Reader may look at [20, section 2.2] to see how one can encode the instaces of algorithmic problems into words on some finite alphabet and use Turing machines to solve the problems. Depending on the nature of the problem, $\mathcal{A}$ is considered either as a recognizer or a function computer (see pages 8 and 9). We say that $\mathcal{A}$ *solves* the algorithmic problem P, if for each instance of P, $\mathcal{A}$ gives the correct answer.

A *complexity function* is a function $f : \mathbb{N} \to \mathbb{N}$. The notation $\mathcal{O}(f)$ means the set of those functions g for which there exists a positive constant M such that $g(n)$ is bounded by $Mf(n)$ for sufficiently large n, i.e. there is a natural number N_0 such that

$$g(n) \leq Mf(n) \text{ for all } n \geq N_0.$$

Analogously, $\Omega(f)$ is the set of functions h for which there is positive M and a natural number N_0 such that

$$h(n) \geq Mh(n) \text{ for all } n \geq N_0.$$

The notations $g = \mathcal{O}(f)$ and $h = \Omega(f)$ mean that $g \in \mathcal{O}(f)$ and $h \in \Omega(f)$.

The concept "size of the input" means here the number of the cells needed to encode the input on the tape of a Turing machine and "computational step" means one usage of the transition.

Definition I.5.1. Let $\mathcal{A}$ be an algorithm that solves the algorithmic problem P. Then

$$\begin{aligned} T_{\mathcal{A}}(n) = \max\{t \in \mathbb{N} \mid\ & a \text{ is an input of } \mathcal{A} \text{ of size } n \\ & \text{and } \mathcal{A} \text{ solves the problem with input } a \text{ in } t \text{ computational steps}\} \end{aligned}$$

is called the *time complexity* of algorithm $\mathcal{A}$. Here "using t computational steps" means naturally $T_1 \vdash^t T_f$, where T_1 is the initial configuration and T_f a final one.

The definition above means clearly the *worst case* complexity. It is clear that there may be an algorithm that can solve the problem much faster for some specific inputs of length n, but there exists at least one input of length n making the computation last $T_{\mathcal{A}}(n)$ steps.

Definition I.5.2. Let $\mathcal{P}$ be an algorithmic problem and $\mathcal{A}_i$, $i \in \mathcal{I}$ the algorithms solving that problem for any "legal" input. Any function $T_{\mathcal{A}_k}$ $k \in I$. that satisfies

$$(\forall i \in I)(\exists N \in \mathbb{N}) : T_{\mathcal{A}_k}(n) \leq T_{\mathcal{A}_i}(n) \quad \text{when } n \geq N$$

is called the *time complexity* of the algorithmic problem $\mathcal{P}$. It is evident that for two such functions $T_{\mathcal{A}_k}$ and $T_{\mathcal{A}_l}$ we have $T_{\mathcal{A}_k}(n) = T_{\mathcal{A}_l}(n)$ for big enough n. Notice also that nothing guarantees the existence of such a function for an arbitrary algorithmic problem.

Definition I.5.2, on the other hand, says that the complexity of an *algorithmic problem* is determined by the *best case* (vs. definition I.5.1). A time complexity function $T_{\mathcal{A}_k}(n)$ thus tells us that the problem $\mathcal{P}$ cannot be solved in *all cases* with input of lenght n any faster that $T_{\mathcal{A}_k}(n)$.

A *nondeterministic algorithm* is here a nondeterministic Turing machine whose each computational path reaches a final state. The time used by a nondeterministic algorithm with input a is defined to be the maximum length of a path in the computation tree from the inititial configuration to a final configuration. The time complexity function of a nondeterministic algorithm $\mathcal{N}$ is then defined as a natural analogue of the time complexity of a deterministic algorithm. Moreover, *probabilistic algorithm* and its time complexity is defined in an analogous way.

An algorithm (deterministic, nondeterministic or probabilistic) is said to be a *polynomial time algorithm*, if $T_{\mathcal{A}}(n) = \mathcal{O}(p(n))$ for some polynomial $p(n)$. Furthermore, an algorithm is called *exponential time algorithm*, if $T_{\mathcal{A}}(n) = \Omega(k^n)$ for some $k > 1$. Polynomial and exponential time algorithmic problems are defined analogiously.

Space complexity is defined in the same manner as the time complexity, but instead of the computational steps one needs to consider the memory resources needed. However, in this thesis we do not need the concept of space complexity.

An algorithmic problem is called *tractable*, if it is of polynomial complexity, and *intractable* otherwise. Tractable algorithmic problems are thought to be "solvable in practice", and intractable ones "practically unsolvable".

Definition I.5.3.

a) Set **P** is the class of the problems that can be solved by a deterministic algorithm in polynomial time.

b) Set **NP** consists of the problems that can be solved by a nondeterministic algorithm in polynomial time.

c) Set **BPP** (bounded probability of error) is the class of problems that can be solved in polynomial time with a probabilistic algorithm that gives the correct answer with probablity at least $\frac{2}{3}$.

In chapter IV more classes recognized by probabilistic machines are introduced.

It is obvious that $\mathbf{P} \subseteq \mathbf{NP}$ and $\mathbf{P} \subseteq \mathbf{BPP}$. One of the most famous open problems in complexity theory is that whether the inclusion $\mathbf{P} \subseteq \mathbf{NP}$ holds strictly. Presently it is expected that the inclusion is strict. Also is is unknown how classes **NP** and **BPP** are related.

The set **BPP** also defines a very important class of algorithmic problems. Suppose that we have a problem and an algorithm $\mathcal{A}$ solving that problem for each "legal" instance. Furthermore we assume that $\mathcal{A}$ gives the correct answer with probability at leat $\frac{2}{3}$, but this is not an important bound. Indeed, any constant greater than $\frac{1}{2}$ would do: Repeating the computation we can push the probability of correct answer *as close to one as we want.* The repeative use of the algorithm gives more reliable algorithm with only a *constant* slowdown.

Extended Church-Turing Thesis. *The class of problems that can be efficiently solved with a realizable automaton is exactly* **BPP**.

In the thesis above "efficiently solvability" means, as mentioned earlier, solvability in polynomial time. The concept "realizable automaton" is now understood intuitively, for instance an automaton computing with real numbers with *infinite precision* seems not to be realizable.

I.6. On the classification of formal languages

Let us consider an alphabet Σ of size n. A *formal language over* Σ or a *formal language*, if the alphabet is fixed, is a set of words over Σ.

Clearly the simplest way to (effectively) give a formal language is to make list of the words belonging to that language. Another way is to give a recognizer, which can be used for any word over Σ to check whether the given word belongs to the language.

One of the most obvious ways to classify formal languages is to order them according to how powerful an automaton is needed to recognize the language. It is clear that for any language recognized by some DFA there exists a Turing machine recognizing the same language, hence $\mathbf{RAT} \subseteq \mathbf{RE}$ (see pages 6 and 8). It is an easy task to show that the inclusion is strict. There are, of course, much more types of automata recognizing special types of formal languages. A well-known classification built this way is called *Chomsky hierarchy* (see [20, p.66] and the references within).

Consider now an algorithmic decision problem $\mathcal{P}$ (output at each instance is either "yes" or "no") and an arbitrary algorithm $\mathcal{A}$ solving that problem. Because of the finite description of an algorithm only finitely many *different* symbols are used to specify *all* inputs of $\mathcal{A}$. Moreover, any input is specified by using only a finite number of symbols, but of course an input can be arbitrarily long. Consequently the set of all *possible* inputs consists of all strings over a finite alphabet. If we denote by L the set of those instances where the answer is "yes", we see that *algorithmic decision problem turns into deciding a formal language.* Moreover, we can accept the concept of a "semialgorithm", which always gives the positive answer correctly, but on negative instaces does not stop at all. The task of this semialgorithm is simply the *acceptance a formal language.* Roughly speaking, the *classification of algorithmic problems turns into the classification of formal languages.* Therefore the definition I.5.3 can be stated more precisely by replacing "the class of problems" with "formal languages".

We have agreed that solvability in polynomial time is the mathematical counterpart for "solvability in practice". Of course this convention is open for criticism in each direction and there are many arguments both for and against it (for discussion, see [20, ch. 1]). Let us take an example of what is different in polynomial an exponential algorithm. Suppose that the time complexity functions of $\mathcal{A}_1$ and $\mathcal{A}_2$ are given by n^s and e^n respectively. Moreover, assume that both algorithms are implemented is computer whose "computational speed" is v, i.e. the results are given in times $\frac{n^s}{v}$ and $\frac{\mathrm{e}^n}{v}$

respectively. If the computational speed is improved k-fold, the new speed is $v' = kv$, then the results are given in times $\frac{n^s}{kv}$ and $\frac{e^n}{kv}$. Now we ask how much bigger instances could be solved in the "maximal" time T after the speed-up? For this purpose, we assume that the instances that can be solved in time T before the speed-up are of sizes n_1 and n_2. It is easy to see that after the speed-up instaces are of sizes $\sqrt[s]{k} \cdot n_1$ and $n_2 + \log k$ respectively, so the *capacity of an exponential algorithm grows only by an additive constant, while that of a polynomial algorithm increases by a multiplicative constant.*

The question of the efficiency of an algorithm raises another possible way to characterize formal languages: Here we keep the model of computation fixed, and ask how many computational steps are required to decide whether a word x of length n belongs to language L or not. More precisely, if a decision algorithm $\mathcal{A}$ has a time complexity function $f(n) = T_{\mathcal{A}}$, we say that the language decided by $\mathcal{A}$ belongs to the class $\mathbf{TIME}(f(n))$. Moreover, if an algorithmic decision problem $\mathcal{P}$ has an algorithm $\mathcal{A} \in \mathbf{TIME}(f(n))$ solving the problem, we say that $\mathcal{P} \in \mathbf{TIME}(f(n))$. In this way the decision algorithms and algorithmic decision problems are classified into *time complexity classes.*

II. Quantum Turing machines

II.1. General overview

The classical Turing machine model reflects very strongly the idea of a deterministic and discrete world. Therefore a Turing machine is thought to be able to totally capture the power of an "intuitive algorithm" - everything that can be computed, can be computed with a Turing machine. However, according to quantum mechanics, the "real world" does not behave that simply. Can one then describe an automaton built on the exotic behaviour of microsystems? Could such a device win the classical computer in the efficiency of computation? Feynman has pointed out in [14] that it seems to be impossible to simulate certain quantum physical systems with a Turing machine *without exponential slowdown.* The research interest in the field of quantum computation has raised after a surprising result: In 1994 Peter W. Shor described a probabilistic algorithm for factorizing numbers on quantum computer in polynomial time, a task thought to be impossible for the classical computers.

In addition to interesting research problems, there is another fundamental reason to investigate quantum computation deeply: The whole idea suggests, at least implicitly, that the *computation should be studied as a phenomenon related to the physical world* rather than as a purely abstract mathematical idea. Modern realizations of automata, like microcircuits, rely on the phenomena of electronics. They are built to perform discrete and deterministic computation. We can thus say that at least in the relizations, the efficiency of the computation is determined by the nature of the "real world", and, according to quantum mechanics, that world is different from the classical deterministic model. Before entering this chapter the reader who is not aware of the basic outlines of the quantum physics should get familiar with the appendix.

II.2. A Mathematical formalism of quantum computation

The definition of a quantum Turing machine given here was first formalized by D. Deutsch in [9]. Local requirements similar to 1a, 2 and 3 below were introduced in [3] to guarantee that the time evolution of the quantum Turing machine will be isometric and requirement 1b is introduced here to make sure that the time evolution will be unitary. Condition 4 ensures that a quantum Turing machine can be finitely described. This definition is, of course, a straightforward generalization of Turing machines introduced in chapter one. Therefore, it seems to be natural to think a quantum Turing machine as the basic model of quantum computation.

Definition II.2.1. A *Quantum Turing machine*, QTM for short, is a quintuple

$$(Q, \Sigma, \delta, q_0, F).$$

All other components are the same as in the definitions of TM, NTM and PTM in the chapter I, but δ is the *transition amplitude function*

$$\delta : Q \times \Sigma \times Q \times \Sigma \times \{L, 0, R\} \to \mathbb{C}$$

which satisfies the following conditions:

1 a) For all $(q_1, \sigma_1) \in Q \times \Sigma$ the sum of squared absolute values of the amplitudes leaving the current configuration equals one:

$$\sum_{\substack{(q,\sigma,d) \\ \in Q\times\Sigma\times\{L,0,R\}}} |\delta(q_1, \sigma_1, q, \sigma, d)|^2 = 1.$$

1 b) For any pair $(q_1, \sigma_1) \in Q \times \Sigma$

$$\sum_{\substack{(q,\sigma,d) \\ \neq(q',\sigma',d')}} \delta(q_1, \sigma_1, q, \sigma, d)\overline{\delta}(q_1, \sigma_1, q', \sigma', d') = 0.$$

2) For all different pairs $(q_1, \sigma_1) \neq (q_2, \sigma_2) \in Q \times \Sigma$ the corresponding sequences of the amplitudes are orthogonal:

$$\sum_{\substack{(q,\sigma,d) \\ \in Q\times\Sigma\times\{L,0,R\}}} \delta(q_1, \sigma_1, q, \sigma, d)\overline{\delta}(q_2, \sigma_2, q, \sigma, d) = 0.$$

3) For any fixed any $(q_1, \sigma_1, \sigma_1')$, $(q_2, \sigma_2, \sigma_2') \in Q \times \Sigma \times \Sigma$ and $d_1 \neq d_2 \in \{L, 0, R\}$,

$$\sum_{q \in Q} \delta(q_1, \sigma_1, q, \sigma_1', d_1)\overline{\delta}(q_2, \sigma_2, q, \sigma_2', d_2) = 0,$$

so the sequences of the amplitudes of reaching a state q from different directions must also be orthogonal.

4) $\delta(q_1, \sigma_1, q_2, \sigma_2, d)$ is always of form $re^{i\pi\theta}$ where $r \geq 0$ and r^2 and θ are rational.

The interpretation of δ is alike to that one of PTM: If the currently scanned symbol is σ_1 and the machine is in the state q_1, then $\delta(q_1, \sigma_1, q_2, \sigma_2, d)$ gives the *transition amplitude* that machine enters the state q_2, symbol σ_1 is overwritten with σ_2, and the head moves one step to the left, holds the current position or moves to the right according to the value $d \in \{L, 0, R\}$.

The requirements 1-3 above are due to the laws of quantum physics. They are here to guarantee that the *dynamical evolution* of the quantum system associated to the QTM will be unitary. This will be discussed later in more detail. Requirement 4 appears useful when we should approximate quantum Turing machines or simulate such one with an ordinary Turing machine. It also guarantees that there are no more than countably many quantum Turing machines.

Definition II.2.2. The *computation tree* C of a QTM, is defined inductively from the root (level 0) to level i in the same way as it was defined for the probabilistic Turing macine, but instead of probabilities, the labels of C are now the corresponding amplitudes. Moreover, the computation tree is not determined only by the transition amplitude, but also by the *observation.* The observation is usually thought to be the final step of the computation and will be discussed later.

Definition II.2.3. The *amplitude* of a vertex v at level i is the product of all the amplitudes on the path from root to v. Moreover, the *configuration space* $\mathcal{C}$ is defined analogously to definition I.4.6. The *superposition of configurations at the stage* i is defined to be a vector of unit length in the configuration space, where the length of a vector

$$C_i = \alpha_1 T_1 + \alpha_2 T_2 \ldots + \alpha_m T_m \tag{2-1}$$

is now $\left(|\alpha_1|^2 + |\alpha_2|^2 + \ldots + |\alpha_m|^2 \right)^{\frac{1}{2}}$. Let

$$C_i = \alpha_{i_1} T_{i_1} + \alpha_{i_2} T_{i_2} + \ldots + \alpha_{i_k} T_{i_k}$$

be the representation of (2-1) where T_{i_j} are distinct and $\alpha_{i_j} \neq 0$, then the *amplitude of a configuration T at stage i*, $a(T, i)$ is defined to be α_{i_l}, if $T = T_{i_l}$ for some l, and 0 otherwise. The *amplitude of reaching final configuration* $a_F(T)$ is the sum of the amplitudes of all halting vertices with entry T.

Definition II.2.4. Configuration T_1 *yields* T_2 *in* k *steps in a quantum Turing machine* $\mathcal{Q}$ *with amplitude* α, if there is a computation tree with root entry T_1 developed in the way explained above having vertex v in level k with probability α and entry T_2. Relation "yields" is defined and both relations are denoted as before.

Definition II.2.5. The value $|a(T,i)|^2$ gives the probability of reaching configuration T at stage i. Moreover, $|a_F(T)|^2$ gives the probability of reaching final configuration T.

It follows from the requirement 1) that the probability $|a(T,i)|^2$ is well-defined, i.e. it obeys the laws of probablility.

Definition II.2.6. (The standard observation) Let

$$C_i = \alpha_1 T_1 + \alpha_2 T_2 + \ldots + \alpha_m T_m$$

be the superposition of configurations at level i with all T_j distinct and α_i nonzero. The *(standard) observation at level* i chooses one configuration T_j with probability $|\alpha_j|^2$ and turns C_i into T_j. All information about the chosen configuration T_j is then available. On the other hand, information of all other configurations is lost.

Intuitively an observation chooses one configuration of the possible ones to be the "actual one" and destroys the superposition. In order to perform a more general observation we will introduce an equivalence in the set of configurations. The equivalence relation $\sim_I$ associated to a finite set $I \subseteq \{-N, \ldots, N\}$ is defined by

$$T_i \sim_I T_j \iff T_i(k) = T_j(k) \text{ whenever } k \in I.$$

The equivalence class determined by T_i will be denoted by $[T_i]$. The *probability* of an equivalence class $[T_i]$ is defined by

$$P([T_i]) = \sum_{T_{i_j} \in [T_i]} |\alpha_j|^2 ,$$

where α_j are the coefficients as above.

The *(standard) observation of cells* I yields equivalence class $[T_i]$ with probability $P([T_i])$ and the post-observation superposition will be

$$\frac{1}{\sqrt{\sum_{T_j \in [T_i]} |\alpha_j|^2}} \sum_{T_j \in [T_i]} \alpha_j T_j$$

provided that the class $[T_i]$ was observed.

Quantum mechanics allows a still more general type of observation. For this purpose we shall use a general notion of *observables*.

Definition II.2.7. Let $\mathcal{C}$ be the configuration space with Euclidian inner product

$$\langle \mathbf{c_1} \mid \mathbf{c_2} \rangle = \sum_{i=i}^{l} \overline{\alpha_i}\beta_i \text{ where } \mathbf{c_1} = \sum_{i=1}^{l} \alpha_i T_{j_i} \text{ and } \mathbf{c_2} = \sum_{i=1}^{l} \beta_i T_{j_i}.$$

Thus $\mathcal{C}$ becomes a complex vector space and $\mathcal{T}$ can be chosen to be an orthonormal basis. An *observable* $\mathcal{E}$ is a finite set $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ of subspaces of $\mathcal{C}$ such that

$$\mathcal{C} = E_1 \oplus E_2 \oplus \ldots \oplus E_m,$$

and E_i and E_j are orthogonal whenever $i \neq j$.

Let now $C \in \mathcal{C}$ be an arbitrary superposition and $\mathcal{E}$ an observable as above. Then there is unique representation of C

$$C = \alpha_1 C_1 + \alpha_2 C_2 + \ldots + \alpha_m C_m$$

such that each C_i is a superposition and $C_i \in E_i$.

Definition II.2.8. Let C and $\mathcal{E}$ be as above. A *(nonstandard) observation of C with respect to observable $\mathcal{E}$* chooses C_i with probability $|\alpha_i|^2$. The post-observation configuration will be C_i provided C_i was observed.

Let now I be a subset of $\mathbb{Z}$ of cardinality l. Consider any fixed string $S = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_l}$ in $(\Sigma \cup Q)^*$ such that at most one σ_{i_j} belongs to Q. There are only finitely many such strings and each one induces a subspace of $\mathcal{C}$ spanned by configurations of form

$$T(k) = \sigma_{i_j} \text{if } k \text{ is the } j\text{:th element of } I.$$

In other words, $T(k)$ is determined by string S if k is an element of I. For each such string S_v we have a subspace $E_v \subseteq \mathcal{C}$. Since all configurations are orthonormal, E_v and E_u are also orthonormal whenever $S_u \neq S_v$. Clearly $\mathcal{C}$ is direct sum of these subspaces, so these form an observable $\mathcal{E}_S$. It is not hard to see that observation with respect to observable $\mathcal{E}_S$ is essentially the same procedure as the general standard obervation in the definition II.2.6.

If we allowed also infinite observables, we could choose $\mathcal{T}$ to be one. Clearly the observation with respect to $\mathcal{T}$ would correspond to the standard observation.

Definition II.2.9 Let T be a configuration and N a natural number such that $T(k) = *$ whenever $k \geq N$. Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ and $Q = \{q_0, \ldots, q_f\}$. Consider then a $d = |\Sigma| + |Q|$-dimensional Hilbert space $\mathcal{H}$ with orthonormal basis

$$\{|\sigma_1\rangle, \ldots, |\sigma_n\rangle, |q_0\rangle, \ldots, |q_f\rangle\},$$

where we have a basis vector for each element of Σ and Q. A *finite model of the tape* is then a $2N+1$-fold tensor product of the space $\mathcal{H}$,

$$\hat{\mathcal{H}} = \bigotimes_{i=-N}^{N} \mathcal{H}_i,$$

where each $\mathcal{H}_i$ is isomorphic to $\mathcal{H}$. The *projection mapping* Π from $\mathcal{C}$ into $\hat{\mathcal{H}}$ is defined in the natural way:

$$\begin{aligned}\Pi : \ldots * **\sigma_{i_{-N}}\sigma_{i_{-N+1}} \ldots \sigma_{i_{N-1}}\sigma_{i_N} *** \ldots \\ \to |\,\sigma_{i_{-N}}\rangle \otimes |\,\sigma_{i_{-N+1}}\rangle \otimes \ldots \otimes |\,\sigma_{i_{N-1}}\rangle \otimes |\,\sigma_{i_N}\rangle.\end{aligned}$$

Definition II.2.10. Let C_i be a superposition as in the definition II.2.6. If $\Pi(C_i)$ can be written as a tensor product of $2N+1$ vectors in $\mathcal{H}$, we say that tape is *decomposable*. Otherwise the tape is said to be *entangled* (cf. the appendix).

Notice that observation as first defined in II.2.6 always destroys the superposition rotating vector C_i most probably to the closest basis vector, but "partial" observations in the definition II.2.6 and will cause only a "partial collapse" of the superposition (cf. the appendix).

Note that now there is an essential difference between the computations of PTM and QTM: If there is a halting vertex with entry T in the computation of a PTM, then the probability of reaching T as a final configuration is inevitably non-negative. On the other hand, suppose that there are exactly two halting vertices in the computation tree of a QTM with entries T. If the amplitudes are α and $-\alpha$ respectively, then the amplitude of reaching a final configuration T equals to zero, and so *one can never reach* the final configuration T. In physics, this phenomenon is called *destructive interference.*

What is especially important, is that since this we consider that *no information about the computation is available before an observation.*

II.3. Unitary operation on the configuration space

Let a quantum Turing machine $\mathcal{Q}$ now be fixed and $\mathcal{C}$ the configuration space equipped with standard Euclidean inner product

$$\langle \mathbf{c_1} \mid \mathbf{c_2} \rangle = \sum_{i=i}^{l} \overline{\alpha_i}\beta_i \quad \text{where } \mathbf{c_1} = \sum_{i=1}^{l} \alpha_i T_{j_i} \text{ and } \mathbf{c_2} = \sum_{i=1}^{l} \beta_i T_{j_i}.$$

Note that the indices in the notation above are chosen in such a way that the set $\{\alpha_i, \beta_i \mid i \in \{1, \ldots, l\}\}$ includes all non-zero coefficients in the representation of $\mathbf{c_1}$ and $\mathbf{c_2}$. Then $\mathcal{C}$ becomes a complex vector space with orthonormal basis $\mathcal{T}$. Note also that $\mathcal{C}$ is *not* complete with respect to the topology induced by the inner product. It is clear that the inner product is well defined on $\mathcal{C}$.

Definition II.3.1. The *time evolution mapping* $U : \mathcal{C} \to \mathcal{C}$ is defined on $\mathcal{T}$ by

$$U(T) = \sum_{i=1}^{k} \alpha_i T_{j_i},$$

where T_{j_i} are configurations such that $T \vdash^1_{\alpha_i} T_{j_i}$ and is extended to $\mathcal{C}$ in the only possible way to be a linear mapping.

Next we notice an obvious relation between the computation tree and mapping U to generalize the computation tree. For this purpose we fix a computation tree of quantum Turing machine $\mathcal{Q}$.

Theorem II.3.2. *Suppose that the superposition of configurations at level i is $\mathbf{c}$. Then the superposition of configurations at next level is $U(\mathbf{c})$.*

Proof. This is obvious: Let $v_1, \ldots, v_m$ be the vertices at level i, $T_1, \ldots, T_m$ their entries and $\alpha_1, \ldots, \alpha_m$ the corresponding amplitudes. Then the superpositition of configurations at level i is given by

$$\mathbf{c} = \alpha_1 T_1 + \ldots + \alpha_m T_m.$$

To obtain level $i+1$, each v_j is equipped with descendants whose entries are those configurations that can be yielded from T_j in one step, and the labels between are the corresponding amplitudes. So the superposition of configurations at level $i+1$ is

$$\alpha_1 U(T_1) + \ldots + \alpha_m U(T_m) = U(\mathbf{c})$$

by the linearity of U. □

Definition II.3.3. The *general computation* of a quantum Turing machine $\mathcal{Q}$ is a sequence of superpositions

$$C : \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \ldots ,$$

where $\mathbf{c}_1$ is a configuration called *input* and for $i > 1$ $\mathbf{c}_i$ is either $U(\mathbf{c}_{i-1})$ or is obtained from $\mathbf{c}_{i-1}$ via an observation procedure. Here U is the time evolution mapping of $\mathcal{Q}$. To avoid the explicit use of the projection postulate, it is assumed that no information of the observed superposition is obtained before the final observation.

Lemma II.3.4. *Mapping U is injective. More precisely, $U^*U = I$, where U^* is the adjoint mapping of U (see the appendix).*

Proof. To handle the mappings in $\mathcal{C}$ more efficiently, we choose a fixed enumeration for all configurations. To find U^* we write

$$U(T_i) = \sum_{l=1}^{\infty} \alpha_{li} T_l$$

for each configuration T_i. Recall that the coefficient α_{li} is the amplitude of reaching T_l from T_i in one step. The sum above is actually finite, since there are only a finite number of configurations reachable from T_i in one step. It is straightforward to verify that the most obvious candidate for the adjoint mapping of U really is the adjoint mapping, namely

$$U^*(T_i) = \sum_{l=1}^{\infty} \overline{\alpha}_{il} T_l.$$

Here also the sum is finite, since any T_i can be reachable only from finitely many configurations. Straightforwardly we see that

$$U^*(U(T_i)) = \sum_{l=1}^{\infty} \Big(\sum_{k=1}^{\infty} \alpha_{ki} \overline{\alpha}_{kl} \Big) T_l,$$

but the right hand side is required to be T_i for each i. This is satisfied, if

$$\sum_{k=1}^{\infty} \alpha_{ki} \overline{\alpha}_{kl} = 0$$

whenever $l \neq i$, and

$$\sum_{k=1}^{\infty} \alpha_{ki} \overline{\alpha}_{ki} = \sum_{k=1}^{\infty} |\alpha_{ki}|^2 = 1.$$

The latter requirement is satisfied because of the condition 1 a) in the definition of a quantum Turing machine.

The former requirement holds if when $i \neq l$, the sequences of the amplitudes $\{\alpha_k \mid T_i \vdash^1_{\alpha_k} T_k, k \in \mathbb{N}\}$ and $\{\alpha_k \mid T_l \vdash^1_{\alpha_k} T_k, k \in \mathbb{N}\}$ are orthogonal. From different configurations T_i and T_l it is possible to obtain T_k only if the head positions in T_i and T_l differ at most two cells. If the head is in the same position in T_i and T_l, then the orthogonality follows from condition 2 and if the head position in T_i and T_l differs one or two cells, the orthogonality follows from condition 3. □

It is worth emphasizing that the previous lemma states that U preserves the inner product; thus $\{U(T_i) \mid i \in \mathbb{N}\}$ is an orthonormal set. We can think that mapping U is represented by an "infinite matrix"

$$U = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \dots \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \dots \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Theorem II.3.4 then essentially states that the columns of matrix U are orthonormal. But this is not everything we would like to say about U. The next theorem gives the final characterization of the time evolution mapping of a quantum Turing machine.

Theorem II.3.5. *Mapping U is unitary.*

The idea in [3] is completed in the proof. Before entering the proof it is worth emphasizing that the sketch in [3] shows that there are no zero rows in the matrix of U. From this it does not directly follow that U is surjective; a counter-example is given by infinite-dimensional matrix

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \dots \\ \frac{1}{\sqrt{2}} & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

which has an orthonormal set of columns and all rows nonzero. However, the matrix above does not determine a surjective mapping, as is easily seen.

Proof of the theorem. It remains to show that the mapping U is surjective, since then U has the inverse mapping $U^{-1} = U^*$. We say that a configuration T_i is reachable from some other if T_i appears in some representation

$$U(T_k) = \alpha_1 T_{i_1} + \ldots + \alpha_i T_i + \ldots + \alpha_m T_{i_m},$$

which happens if the i:th row of the matrix of U is not a zero row. We are first going to show that there are no zero rows. Suppose, on the contrary that there is a configuration T_N that is not reachable from any other configuration. We then denote the symbols surrounding the currently scanned, σ_2, by σ_1 and σ_3 and the state by q, so there is a sequence $\sigma_1 q \sigma_2 \sigma_3$ somewhere on the tape in consecutive cells. Now it is clear that *all the configurations having that sequence somewhere on the tape are also not reachable.* The configurations having this sequence somewhere are said to *locally look like* T_N.

Let $n \geq 4$ and construct $K = n\,|Q|\,|\Sigma|^n$ configurations $T_{i_1}, \ldots, T_{i_K}$ such that in these configurations all the cells outside $0, 1, \ldots, n$ are blank and the symbol from the state set is not in n. For the K-dimensional subspace $V \subset \mathcal{C}$ generated by $T_{i_1}, \ldots, T_{i_K}$ we have $\dim U(V) = K$, because U is injective.

On the other hand, from *any* T_{i_k} one can yield in one step another T_{i_l} or exit the chosen n cells (there are at most $2\,|Q|\,|\Sigma|^n$ configurations to exit into), so yield at most $K + 2\,|Q|\,|\Sigma|^n$ configurations. But at least $(n-2)\,|\Sigma|^{n-3}$ of T_{i_k} locally look like T_N, so they cannot be yielded from anywhere in one step. Consequently we get a representation

$$U(T_{i_k}) = \sum_{j_l \in \mathcal{J}} \alpha_l T_{j_l}$$

with index set $\mathcal{J}$ of cardinality at most $K + |Q|\,|\Sigma|^n - (n-2)\,|\Sigma|^{n-3}$. Therefore

$$K = \dim U(V) \leq K + |Q|\,|\Sigma|^n - (n-2)\,|\Sigma|^{n-3},$$

which is equivalent to $n \leq 2\,|Q|\,|\Sigma|^3 + 2$. But n can be chosen arbitrarily large, which leads to a contradiction.

Without loss of generality we can assume, for the sake of contradiction, that T_1 is not in the range of U. From the derivation above, it follows that the first row of the time evolution matrix is not a zero row. On the other hand, from condition 1b we can infer easily that any two rows of the time evolution matrix must be orthogonal. Let us choose N such that $\alpha_{1j} = 0$ whenever $j > N$. Since rows are orthogonal, we have

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1N} & \dots \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2N} & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \overline{\alpha}_{11} \\ \overline{\alpha}_{12} \\ \vdots \\ \overline{\alpha}_{1N} \\ 0 \\ \vdots \end{pmatrix} = \begin{pmatrix} C \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

where $C = |\alpha_{11}|^2 + \ldots + |\alpha_{1N}|^2 > 0$. Written in another form,

$$U(\overline{\alpha}_{11} T_1 + \ldots + \overline{\alpha}_{1N} T_N) = C T_1.$$

Dividing by C we obtain the preimage of T_1, a contradiction. □

Corollary II.3.6. *The computation of a quantum Turing machine is reversible, i.e. at each step of the computation the superposition of configurations uniquely determines the previous superposition.*

Proof. As an unitary mapping U has an inverse which in this case is U^*. □

Corollary II.3.6 contains something that sounds paradoxical: It sounds not unfamiliar that the future computation is fully determined when the initial configuration is known, but *also the past computation is known*, so we can at least in principle determine the superposition before the initial one! Therefore we will begin thinking of the computation of a quantum Turing machine as a sequence of superpositions that continues infinitely in both directions rather than a tree. However, there is one thing that violates the determinism - the observation.

There also rises a fundamental problem in the termination of the computation. Since each configuration determines the previous one uniquely, we cannot have any "true final configuration" which would not change any longer. We can however avoid this problem by taking the occurrence of a final state as a sign given by the machine, indicating that the result is "complete enough" for the final observation. This usually requires more standardization in the model of a quantum Turing machine, for example it could be required that the cell number zero should contain symbol q_f. Another suitable model is to predestine the number of the steps of the computation, and observe after those steps have been done. This point of view may sound more realizable when we want to consider inputs of certain length, and leads naturally to the idea of a finite quantum automaton. This will be discussed later on.

II.4. Simulation of a QTM in a finite-dimensional space

We fix two natural numbers $M \leq N$ and assume that a quantum Turing machine $\mathcal{Q} = (Q, \Sigma, \delta, q_0, F)$ satisfies the following condition: For any input of length at most M the machine never scans cells other than $\{-N+1, \dots, N\}$ in forward or backward computation. This assumption, made for quantum Turing machines or for classical ones, makes the model of computation more realistic: The memory of any imaginable physical device is bounded; in this particular case the size of memory is thought to be $2N+1$ and inputs longer than M cannot be handled with certainty. This assumption also leads to ideas of a finite quantum automaton and quantum gates. Both will be introduced in the next chapter, but we will see first where this formalism leads to.

Let $\mathcal{T}'$ be the set of those configurations that occur in some superposition that is visited when the length of the input is at most M, more precisely, we define

$$\begin{aligned}\mathcal{I} = \{\mathbf{c} \in \mathcal{T} \mid \mathbf{c} &\text{ is a configuration of form} \\ &\dots * * * \, q_0 x_1 x_2 \dots x_m * * * \dots \text{ with } q_0 \text{ in cell } 0 \text{ and } m \leq M\},\end{aligned}$$

$$\mathcal{S} = \bigcup_{\mathbf{c} \in \mathcal{I}} \{U^k(\mathbf{c}) \mid k \in \mathbb{Z}\} \text{ and finally}$$

$$\mathcal{T}' = \{T \in \mathcal{T} \mid \langle T \mid S \rangle \neq 0 \text{ for some } S \in \mathcal{S}\}.$$

By hypothesis, any $q \in Q$ can occur only in cells $-N, \dots N-1$, so the blank symbol in cells outside of range $\{-N, \dots, N\}$ *can never be changed.* It follows that cardinality of $\mathcal{T}'$ is at most $2N\,|Q|\,|\Sigma|^{2N}$. Next we will consider the finite-dimensional subspace $\mathcal{C}' \subset \mathcal{C}$ generated by $\mathcal{T}'$.

Lemma II.4.1. *The restriction of U on subspace $\mathcal{C}'$ is a unitary mapping.*

Proof. Choose any $\mathbf{c} \in \mathcal{C}'$ and let

$$\mathbf{c} = \alpha_1 T_{i_1} + \alpha_2 T_{i_2} + \dots + \alpha_m T_{i_m}$$

be the canonical representation of $\mathbf{c}$ in base $\mathcal{T}' \subset \mathcal{T}$. To see that $\mathcal{C}'$ is closed under U is simple: If $T_{i_k} \in \mathcal{C}'$ but $U(T_{i_k}) \notin \mathcal{C}'$, there would exist a superposition $\mathbf{c}' \in \mathcal{C}'$ such that T_{i_k} appears in the representation of $\mathbf{c}'$:

$$\mathbf{c}' = \beta_1 T_{i_1} + \dots + \beta_k T_{i_k} + \dots + \beta_n T_{i_n}.$$

By definition $\beta_1 U(T_{i_1}) + \ldots + \beta_k U(T_{i_k}) + \ldots + \beta_n U(T_{i_n}) = U(\mathbf{c}') \in S \subseteq \mathcal{C}'$, so counting the inner products with $U(T_{i_k})$ in both sides we get $\beta_k = 0$ since all $U(T_i)$ are orthogonal, a contradiction.

Moreover, since U is injective, so must be the restriction of U and because $\mathcal{C}'$ is of finite dimension, the restriction is surjective also. It is easily seen that the restriction of U^* is the inverse of the restriction of U. $\square$

Recall the vector space $\hat{\mathcal{H}}$ in the section II.2. The projection mapping $\Pi : \mathcal{C} \to \hat{\mathcal{H}}$ can also be restricted to $\mathcal{C}'$ and maps initial configuration

$$T : \ldots * * * \, q_0 \sigma_{i_1} \sigma_{i_2} \ldots \sigma_{i_m} * * * \ldots$$

into

$$\Pi(T) = \mid * \rangle \otimes \ldots \otimes \mid * \rangle \otimes \mid q_0 \rangle \otimes \mid \sigma_{i_1} \rangle \otimes \mid \sigma_{i_2} \rangle \otimes \ldots \otimes \mid \sigma_{i_m} \rangle \ldots \otimes \mid * \rangle \ldots \otimes \mid * \rangle$$

Clearly Π becomes an isometric mapping; it maps orthonormal basis of $\mathcal{C}'$ into orthonormal basis of $\Pi(\mathcal{C}')$. Therefore the restriction of U on $\mathcal{C}'$ also induces a unitary mapping $\hat{U} : \Pi(\mathcal{C}') \to \Pi(\mathcal{C}')$ defined by $\hat{U}(\mid \varphi\rangle) = \Pi(U(\Pi^{-1}(\mid \varphi\rangle)))$.

Theorem II.4.2. *Mapping $\hat{U} : \Pi(\mathcal{C}') \to \Pi(\mathcal{C}')$ can be extended into unitary mapping $\widetilde{U} : \hat{\mathcal{H}} \to \hat{\mathcal{H}}$ whose restriction on subspace $\Pi(\mathcal{H})$ is $\hat{U}$.*

Proof. We define

$$\widetilde{U}(\mid \psi\rangle) = \begin{cases} \hat{U}(\mid \psi\rangle) & \text{if } \mid \psi\rangle \in \Pi(\mathcal{C}') \\ \mid \psi\rangle & \text{otherwise.} \end{cases}$$

To see that $\widetilde{U}$ is unitary it suffices to check that $\widetilde{U}$ preserves the norm: If it does, it follows from the *polarization equation* (straightforward to verify)

$$\langle \varphi \mid \psi \rangle = \frac{1}{4} \sum_{k=0}^{3} i^k \langle \psi + i^k \varphi \mid \psi + i^k \varphi \rangle$$

that $\widetilde{U}$ preserves the inner product, too. If $\widetilde{U}$ preserves norm, it is injective, and the space being of finite dimesion, surjective too. Then

$$\langle \varphi \mid \widetilde{U}(\psi) \rangle = \langle \widetilde{U}\widetilde{U}^{-1}(\varphi) \mid \widetilde{U}(\psi) \rangle = \langle \widetilde{U}^{-1}(\varphi) \mid \psi \rangle$$

for all $\varphi, \psi \in \hat{\mathcal{H}}$. It follows that $\widetilde{U}^* = \widetilde{U}^{-1}$. To see that $\widetilde{U}$ preservs the norm is a trivial task, since $\hat{U}$ does, being an unitary mapping. $\square$

So we have obtained

Theorem II.4.3. *Suppose that for a quantum Turing machine $\mathcal{Q}$ there are two natural numbers $M \leq N$ such that when proceeding with inputs no longer than M $\mathcal{Q}$ never scans other cells than $-N+1, \ldots, N$. Then there* *exists a* $(|Q|+|\Sigma|)^{2N}$*- dimensional complex Euclidian vector space* $\hat{\mathcal{H}}$ *and a unitary mapping* $\widetilde{U}:\hat{\mathcal{H}}\to\hat{\mathcal{H}}$ *such that* $\Pi\phi(U^k(\mathbf{c}))=\widetilde{U}^k(\Pi(\mathbf{c}))$ *for all* $k\in\mathbb{Z}$, *where* $\mathbf{c}$ *is an input configuration of form* $\mathbf{c}=\ldots***q_0\sigma_1\sigma_2\ldots\sigma_m***\ldots$ *with* q_0 *in cell* 0 *and* $m\leq M$.

The theorem says intuitively that for an always halting quantum Turing machine $\mathcal{Q}$ there is a "finite model" that operates with a finite number of inputs - with those that are no longer than M - exactly in the same fashion that $\mathcal{Q}$ did. In fact, this result should not be a surprise: Consider that we have a special case of quantum Turing machine, an ordinary reversible Turing machine M that also satisfies the conditions of theorem II.4.3. Then M simply permutes finitely many configurations, and we could establish a finite-dimensional vector space with orthonormal basis labelled with those configurations and embed this in $\hat{\mathcal{H}}$. Then the time evolution mapping of this space would be a bijective permutation mapping, *which is always unitary.* Theorem II.4.3 simply generalizes this fact.

Despite the triviality of the theorem II.4.3 there is one interesting thing in the construction of $\hat{\mathcal{H}}$. First, $\hat{\mathcal{H}}$ is bigger than really needed since, for example, elements of form $|\,q_0\rangle\otimes|\,q_0\rangle\otimes\ldots$ do not have any counterpart in $\mathcal{C}'$. However, $\hat{\mathcal{H}}$ is big enough to give room for intuition of a memory cell: Each component in the inner product actually servers as a memory cell where one can store an element of $Q\cup\Sigma$. On the other hand, the action of the mapping $\widetilde{U}$ is local as the action of the original QTM is: *at most three cells are affected at the same time.* The structure of $\hat{\mathcal{H}}$ suits very well to this purpose: At each stage of computation $\widetilde{U}$ changes vectors in $(|Q|+|\Sigma|)^3$-dimensional subspace $\mathcal{H}_l\otimes\mathcal{H}_{l+1}\otimes\mathcal{H}_{l+2}$ and and leaves all other components unchanged. In addition to this we know that we can build unitary mappings from known ones by a tensor product construction: If $U_{-N},\ldots,U_N$ are unitary mappings $U_i:\mathcal{H}_i\to\mathcal{H}_i$, then the tensor product of mappings U_i defined by

$$(U_{-N}\otimes\ldots\otimes U_N)(|\,\varphi_{-N}\rangle\otimes\ldots\otimes|\,\varphi_N\rangle)=U_{-N}(|\,\varphi_{-N}\rangle)\otimes\ldots\otimes U_N(|\,\varphi_N\rangle)$$

is known to be a unitary mapping $\hat{\mathcal{H}}\to\hat{\mathcal{H}}$. This raises a natural question whether we can build the unitary mapping $\widetilde{U}$ from simpler ones. Of course it would be convinient if one could represent the unitary transform matrix of a quantum Turing machine by a product of very simple unitary matrices or even as a tensor product of smaller matrices.

Before investigating this question more precisely, we will give a meaning to the superposition configurations in terms of quantum physics.

II.5. A simple physical model

We will first adopt two conventions:

1) For any Hilbert space $\mathcal{H}$ of finite dimension d one can find a d-state quantum physical system S such that the Hilbert space associated to S "really" is $\mathcal{H}$.

2) For any physical system S and the associated Hilbert space $\mathcal{H}$ there is a mapping $s : \mathbb{Z} \to \mathcal{H}^1 = \{\varphi \in \mathcal{H} \mid ||\varphi|| = 1\}$. Vector $s(t)$ is called the *state vector of system S at time t*. Note that domain set $\mathbb{Z}$ instead of $\mathbb{R}$ is used to indicate that the development of state vector is performed in discrete steps. Recall from the appendix that a quantum state may now be a basis state or a superposition of the basis states.

Consider now the finite configuration space $\hat{\mathcal{H}}$ in the previous section. This raises naturally a physical model: By convention 1 one can find a physical $d = |\Sigma| + |Q|$-state system S_1 such that $\mathcal{H}$ is the Hilbert space associated to this system. We suppose furthermore that we can find $2N+1$ such systems and set them in interaction with each other. Of course this is a very strong assumption and it is not clear that it can be satisfied at all in the "true world", but in this chapter we will stay in the pure formalism.

According to the appendix the Hilbert space associated to the compound system of all $2N+1$ subsystems is the tensor product $\hat{\mathcal{H}}$. Furthermore, we assume that we can prepare the physical conditions in such a way that the time evolution of this system goes via unitary transformation $\widetilde{U}$. In other words we suppose now that there really exists a physical system that can first be prepared into pure state

$$|*\rangle \otimes \ldots \otimes |*\rangle \otimes |q_0\rangle \otimes |\sigma_{i_1}\rangle \otimes \ldots \otimes |\sigma_{i_m}\rangle \otimes \ldots \otimes |*\rangle,$$

and that the time evolution of this system runs via unitary transformation $\widetilde{U}$. This requires that any three "consecutive" system interact.

Let us now look at the influence of measurement. We suppose that the pre-measurement state of the composite system is

$$\sum_{i=1}^{m} \alpha_i \, |\sigma_{i_{-N}}\rangle \otimes \ldots \otimes |\sigma_{i_N}\rangle$$

with $|\alpha_1|^2 + \ldots + |\alpha_m|^2 = 1$. We use the observable A as in the appendix. If the cells to be observed are $i_1, \ldots, i_n$, one can check that measuring observable

$$\bigotimes_{j=-N}^{N} T_j,$$

where $T_j = I$ if $j \notin \{i_1, \ldots, i_n\}$ and $T_j = A$ otherwise, corresponds to the observation defined in II.2.9. Moreover, the superposition "collapses" according to the observed value.

We will now reconsider the problem of termination of the computation. If this realization can be built, one could observe *after each step* the cell $-N$ (which is supposed to remain unused in the computation), but when q_f is observed, the user also observes the cells where the output is expected to be. Another way to control the computation is to run it for a pre-determined number of steps and then observe.

There is another problem that rises with the unitary evolution $\widetilde{U}$, which is now a unitary mapping in $(|\Sigma| + |Q|)^{2N+1}$-dimensional Hilbert space. As in Feynman's idea we should be able to construct a Hamiltonian operator H_i on $\hat{\mathcal{H}}$ which will determine the time evolution on each computational step by $U_i(t) = \mathrm{e}^{-itH_i}$, but there may be difficulties with the practical implementation of an arbitrary H. On the other hand, it is known how to implement some unitary evolutions in two-state systems in optical networks [25].

So we will study the problem of splitting the unitary transformation matrix into smaller or somehow simpler ones. A trivial result can easily be obtained:

Theorem II.5.1. *Any $d \times d$ unitary matrix can be represented as a product of d^2 unitary $d \times d$ matrices that each affect only a two-dimensional subspace spanned by two natural basis vectors.*

Proof. Let U be a fixed unitary $d \times d$ matrix. First we notice that the eigenvectors of U, say $\varphi_1, \dots, \varphi_d$ can be chosen in such a way that they form an orthogonal basis of the complex vector space $\mathbb{C}^d$. This fact can be proved easily by induction. If we could operate in the basis $B = \{\varphi_1, \dots, \varphi_d\}$, the mapping U would be easy to perform:

$$U(\alpha_1\varphi_1 + \ldots + \alpha_d\varphi_d) = \alpha_1\lambda_1\varphi_1 + \ldots + \alpha_d\lambda_d\varphi_d,$$

where λ_j is the eigenvalue associated to φ_i. So in basis B the matrix of mapping induced by U is merely a diagonal unitary matrix that can certainly be represented as a product of d unitary matrices $U_1', \dots, U_d'$ that each operate on one-dimensional subspace spanned by some φ_i.

We will show that there exist $k = \frac{d(d-1)}{2}$ unitary rotations $U_1, \dots, U_k$ whose product rotates the basis $\{\varphi_1, \dots, \varphi_d\}$ into the natural basis $\{\mathbf{e}_1, \dots, \mathbf{e}_d\}$ (note that unitary transforms always preserve orthonormality). If there are such ones, then $U_k^{-1} \cdot \ldots \cdot U_1^{-1}$ turns the natural basis into B. Thus

$$U = U_1 \cdot \ldots \cdot U_k U_1' \cdot \ldots \cdot U_d' U_k \cdot \ldots \cdot U_1^{-1}$$

is the representation required. To show that $U_1, \dots, U_k$ exist, we notice that if $\varphi = (c_1, \dots, c_l, \dots, c_d)$, then the unitary mapping induced by matrix

$$U_l = \frac{1}{\sqrt{|c_1|^2 + |c_l|^2}} \begin{pmatrix} \overline{c_1} & 0 & \dots & \overline{c_l} & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ c_l & 0 & \dots & -c_1 & \dots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{pmatrix}$$

acts only on the two-dimensional subspace spanned by $\mathbf{e}_1$ and $\mathbf{e}_l$ and makes the l:th component of φ zero. Thus we can find suitable unitary matrices U_2, $\ldots$, U_d such that all but the first component of $U_1'\varphi_1 = U_d \cdot \ldots \cdot U_2\varphi_1$ are zero. Now U' maps B into an orthonormal basis, so the first components of the vectors $U_1'(\varphi_2)$, $\ldots$, $U_1'(\varphi_d)$ must be zero. Now delete the first component and consider the orthogonal basis of $\mathbb{C}^{d-1}$ thus obtained. By induction we can find $\frac{(d-1)(d-2)}{2}$ unitary $(d-1)\times(d-1)$ matrices whose product maps this basis into $\{\lambda_2\mathbf{e}_2', \ldots, \lambda_d\mathbf{e}_d'\}$, where $\lambda_i = e^{i\theta_i}$. All these matrices can be extended into $d\times d$-matrices by adding one zero column to the left and the top row with first component one and all other zero. Totally we have $\frac{(d-1)(d-2)}{2} + d - 1 = \frac{d(d-1)}{2}$ unitary matrices to perform the rotation of basis B into $\{\lambda_1\mathbf{e}_1, \ldots, \lambda_d\mathbf{e}_d\}$. Since all λ_i:s are on the unit circle, we can take them in account in unitary diagonal matrices U_i'. □

Alike theorem II.4.3, this theorem generalizes the fact that any permutation matrix can be written as a product of two-dimensional permutation matrices; it is a well-known fact that any permutation group is generated by the transpositions.

Theorem II.5.1 provides a decomposition result, but not a satisfactory one, even though it provides a decomposition into polynomially many two-dimensional matrices: For an input of length N the finite simulation of a Turing machine makes use of $(|Q| + |\Sigma|)^{2N+1}$-dimensional Hilbert space, so the dimension is exponential in N. For a satisfactory result we should decompose U into a tensor product of unitary matrices or in some other way yielding polynomially (in N) many matrices. It is an open question if there are such unitary matrices U on $\hat{\mathcal{H}}$ that the decomposition of $U = U_1 \ldots U_m$ into matrices operating on two-dimensional subspaces will require that m is always exponential in $2N+1$. For a unitary evolution induced by a quantum Turing machine this question is solved in chapter III.

We have seen that for some quantum Turing machines one can build a physical model to perform the computation. Next we will consider the question of simulation of an ordinary Turing machine with a quantum Turing machine. The problem arising is that a quantum Turing machine is always reversible, but an ordinary Turing machine need not be.

II.6. The reversibility of the computation

Definition II.6.1. A Turing machine is *reversible*, if the current configuration determines the previous one uniquely.

Lemma II.6.2. *Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ be an ordinary Turing machine. Then $\mathcal{M}$ is reversible if and only if δ is injective and satisfies*

$$\left.\begin{aligned}\delta(q_1,\sigma_1) &= (q,\sigma_1',d_1)\\ \delta(q_2,\sigma_2) &= (q,\sigma_2',d_2)\end{aligned}\right\} \implies d_1 = d_2.$$

Proof. Suppose that $\mathcal{M}$ is reversible. Then clearly δ must be injective. Suppose then that there are pairs $(q_1, \sigma_1) \neq (q_2, \sigma_2)$ such that

$$\begin{aligned}\delta(q_1, \sigma_1) &= (q, \sigma_1', d_1) \text{ and} \\ \delta(q_2, \sigma_2) &= (q, \sigma_2', d_2).\end{aligned}$$

Suppose also that $d_1 = L$ and $d_2 = R$. Then the configurations T_1 and T_2 which are identical everywhere else except the four consequtive cells, where T_1 has $\sigma_2' \eta q_1 \sigma_1$ and T_2 has $q_2 \sigma_2 \eta \sigma_1'$ both yield a configuration where in those four cells $\sigma_2' q \eta \sigma_1'$ is written. Thus $\mathcal{M}$ is not reversible. Similary one can check for any other combination of d_1 and d_2 that $\mathcal{M}$ cannot be reversible if the condition of the lemma does not hold.

Suppose then that δ satisfies the condition of the lemma and consider any configuration T. Assume also that there are consecutive symbols $\sigma_1 q_1 \sigma_2 \sigma_3$ on the tape. Since we supposed that δ is completely specified, there are exactly $|Q|\,|\Sigma|$ elements in the image of δ. On the other hand, for each state q there is at most one d, so if (q_1, σ, d) is not in the image of δ for any choice of σ and d, the image would be too small. Without loss of generality we can suppose that (q_1, σ, R) lies in the image of δ. With a similar argument we infer that also (q_1, σ_1, R) is in the image of δ. Since δ is injective, there is unique pair (q_2, σ') such that $\delta(q_2, \sigma') = (q_1, \sigma_1, R)$. This reveals the former configuration uniquely. □

Definition II.6.3. A quantum Turing machine $\mathcal{Q}$ *simulates* an ordinary Turing machine $\mathcal{M}$, if the alphabets and state sets are the same and the computation tree of $\mathcal{Q}$ beginning at any configuration is a single path, exactly the same as the computation of $\mathcal{M}$ beginning at the same configuration.

Lemma II.6.4. *A Turing machine can be simulated with a quantum Turing machine if and only if it is reversible.*

Proof. Suppose that $\mathcal{M}$ is not reversible, but there is a quantum Turing machine $\mathcal{Q}$ that simulates $\mathcal{M}$. Then there are two configurations $T_1 \neq T_2$ such that $T_1 \vdash T$ and $T_2 \vdash T$. As $\mathcal{Q}$ simulates $\mathcal{M}$, using U^{-1} we obtain $T_1 = T_2$.

Assume then that $\mathcal{M}$ is reversible. We define a QTM $\mathcal{Q} = (Q, \sigma, \delta', q_0, F)$, where

$$\delta'(q_1, \sigma_1, q_2, \sigma_2, d) = 1 \text{ if and only if } \delta(q_1, \sigma_1) = (q_2, \sigma_2, d)$$

and $\delta'(q_1, \sigma_1, q_2, \sigma_2, d) = 0$ otherwise. For a Turing machine $\mathcal{M}$ we assumed that the transition function δ is completely defined. Therefore δ' becomes a completely defined function, too. It is a straightforward task to see that δ' satisfies conditions 1-4 on the transition amplitude function. Thus $\mathcal{Q}$ becomes a well-defined quantum Turing machine that clearly simulates $\mathcal{M}$.
□

In general a Turing machine $\mathcal{M}$ does not have to be reversible, but a quantum Turing machine always is. Therefore it is not clear that a computation of $\mathcal{M}$ can in every case be simulated by any quantum Turing machine. However this problem can be solved by introducing a two-tape quantum Turing machine and using the idea of Bennett in [1]. Bennett's theorem informally states that for each 1-tape Turing machine $\mathcal{M}$ there exists a reversible 3-tape Turing machine $\mathcal{M}'$ that simulates $\mathcal{M}$ with a constant slowdown. On the other hand, it is well known that a k-tape Turing machine can be simulated by an 1-tape machine with quadratic slowdown (see [20, p.30]). One question still remains: Can this *whole process, the simulation* of a reversible Turing machine be done reversibly?

Bennett's idea was to establish a 3-tape machine that simulates a 1-tape machine on its first tape and keeps on writing the history list of the computation on the second tape. The third tape is only for copying the outcome of the computation (this can be done reversibly) before running the computation backwards to erase the history tape. Bennett was at least partially motivated by remarkable consequences that the reversibility of the computation has: It was mentioned in [1] that an irreversible overwriting of a bit of an information must dissipate energy at least $kT \ln 2$, where k is Boltzmann's constant. A reversible computation can be performed with much less energy dissipation, in fact, the ideal would be thermodynamically reversible computation with zero energy dissipation.

As mentioned above, there is one thing open for criticism in Bennett's result: It guarantees that each computation *with legal input* will be reversible, that means a computation beginning with input on the first tape and both other tapes empty. We will however need a stricter concept of irreversibility: Each configuration, no matter shall it occur on *any computation* must determine the previous one uniquely. Whether it is possible to simulate generally irreversible Turing machine with a reversible multitape one, is not a trivial problem a priori. Let us now investigate with this stricter point of view how to convert an ordinary Turing machine $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ into an equivalent reversible multitape machine. For this purpose we will introduce a *two-tape Turing machine*.

Definition II.6.5. A two-tape Turing machine is of a sixtuple

$$(Q, \Sigma, \Gamma, \delta, q_0, F).$$

Tape alphabets are Σ and Γ respectively. It is required that the unique symbol in Q on both tapes is equal at each step. The function

$$\delta : Q \times \Sigma \times \Gamma \to Q \times \Sigma \times \Gamma \times \{L, 0, R\}^2$$

is again called the transition function.

The action of a two-tape Turing machine is intuitively described as follows: If the machine is in state q, heads are scanning symbols σ and γ

respectively, and $\delta(q, \sigma, \delta) = (q', \sigma', \gamma', d_1, d_2)$, then the machine enters state q', σ' and γ' are printed in the currently scanned squares, and the heads move in directions d_1 and d_2 respectively.

Definition II.6.6. The *configuration* of a two-tape Turing machine is an ordered pair $(T^{(1)}, T^{(2)})$, where $T^{(1)}$ and $T^{(2)}$ are ordinary configurations as in definition I.2.4 such that if $T^{(1)}(k_1) \in Q$ and $T^{(2)}(k_2) \in Q$, then $T^{(1)}(k_1) = T^{(2)}(k_2)$. It is obvious that there are only countably many configurations, hence we can choose a fixed numbering and denote by T_1, T_2, $\ldots$ all the configurations. Configuration T_1 *yields* configuration T_2 in one step if the latter is obtained from the former one as the result of the action of the machine. In this case we again denote $T_1 \vdash^1 T_2$. Relations "yields in k steps" and "yields" are defined in the obvious way.

Again we take complex numbers instead of reals to obtain a general concept suitable also for quantum computation.

Definition II.6.7. The *configuration space* $\mathcal{C}$ of a two-tape Turing machine is the linear span of all configurations over the field of complex numbers. Relation "yields in one step" here also defines a linear mapping M on the configuration space by

$$M(T_1) = T_2, \text{ where } T_1 \vdash^1 T_2.$$

For future use, we introduce now a generalization of a two-tape Turing machine in the quantum computation.

Definition II.6.8. A *two-tape quantum Turing machine* is a sixtuple

$$(Q, \Sigma, \Gamma, \delta, q_0, F),$$

where all the other components and notations are as in a two-tape Turing machine, but δ is a transition amplitude function

$$\delta : Q \times \Sigma \times \Gamma \times Q \times \Sigma \times \Gamma \times \{L, 0, R\} \times \{L, 0, R\} \to \mathbb{C}.$$

The interpretation of δ is analogous to the one-tape quantum Turing machine. Configuration space is defined in the same way as in two-tape Turing machines. It is required that δ satisfies the following conditions:

Denote $T_j \vdash^1_{\alpha_{ij}} T_i$, if T_j yields T_i in one step. Then

1 a) The squared sum of absolute values of amplitudes leaving a single configuration adds up to one:

$$\sum_{k=1}^{\infty} |\alpha_{ki}|^2 = 1 \text{ for each } i \text{ and}$$

1 b) The rows of the time evolution matrix are orthogonal: If $k \neq l$, then

$$\sum_{j=1}^{\infty} \overline{\alpha}_{kj} \alpha_{lj} = 0.$$

2) For all $i \neq l$ we have

$$\sum_{k=1}^{\infty} \overline{\alpha_{ki}} \alpha_{kl} = 0.$$

3) The same as the requirement 4 (p. 19) for one-tape machines.

Note that the sums in 1 and 2 are actually finite as in the case of ordinary quantum Turing machine.

Conditions 1 and 2 will guarantee that the time evolution mapping on configuration space determined by the transition function is unitary and the computation therefore reversible. The proof is obtained by slightly modifying lemma II.3.4 and theorem II.3.5.

Analogously we say that a two-tape Turing machine is *reversible*, if the current configuration always determines the previous one uniquely.

Lemma II.6.9. *For any one-tape Turing machine there exists a reversible two-tape Turing machine that simulates the original one on the first tape.* Transition rules of the new machine are listed below in seven groups. The idea is to simulate the original machine on the first tape and simultaneously write the history of the computation on the second one. Groups 1 and 2 are for this purpose. However, if the machine finds a nonempty cell on the history list, the computation cannot continue. In this case, our new machine obeys a transition in group 3 and begins to compute backwards according to the rules in 4 and 6. The backwards computation may fail for two reasons: The "end" of the history list is found (case 5) or the history item do not match (case 6). In both cases, the ordinary computation begins again. It is clear that this construction simulates correctly the original machine if the computation begins with empty history list. In this case it is also easy to see that the time usage is redoubled.

Proof of the lemma. The tape alphabet of the two-tape machine $\mathcal{M}'$ for the first tape is Σ and for the second tape $\Gamma = (Q \times \Sigma \times \{L, 0, R\}) \cup \{*\}$. The state set of $\mathcal{M}'$ is $Q' = Q \cup Q_w \cup Q_r \cup Q_{rw}$, where Q_w, Q_r and Q_{rw} are pairwise disjoint copies of the original state set Q. Copies of a state q are denoted by q_w, q_r, q_{rw}.

Transition function

$$\delta' : Q \times \Sigma \times \Gamma \to Q \times \Sigma \times \Gamma \times \{L, 0, R\} \times \{L, 0, R\}$$

of machine M' is defined by following groups of rules: For any q, σ, q_1, σ_1 and d

1) $\delta'(q, \sigma, \mathbf{x}) = (q_w, \sigma, \mathbf{x}, 0, R)$ for any $\mathbf{x} \in \Gamma$.

2) $\delta'(q_w, \sigma, *) = (q_1, \sigma_1, (q, \sigma, d_1), d_1, 0)$, where q_1, σ_1 and d_1 are determined by $\delta(q, \sigma) = (q_1, \sigma_1, d_1)$.

3) $\delta'(q_w, \sigma, (p, \eta, d)) = (q_r, \sigma, (p, \eta, d), 0, L)$ for each $(p, \eta, d) \in \Gamma \setminus \{*\}$.

4) $\delta'(q_r, \sigma, (p, \eta, d)) = (q_{rw}, \sigma, (p, \eta, d), -d, 0)$

5) $\delta'(q_r, \sigma, *) = (q, \sigma, *, 0, 0)$

6) $\delta'(q_{rw}, \sigma, (p, \eta, d))$

$$= \begin{cases} (p_r, \eta, *, 0, L) & \text{if } \delta(p, \eta) = (q, \sigma, d) \\ (q, \sigma, (p, \eta, d), d, 0) & \text{otherwise} \end{cases}$$

7) $\delta'(q_{rw}, \sigma, *) = (q_{rw}, \sigma, *, 0, 0)$

To show that any configuration determines a unique predecessor, we study four cases.

In the first case we have a configuration T with state q in Q. Looking at the groups we see that the only possibilities for the state of the former configuration is a state in set Q_w (group 2), Q_r (group 5) or Q_{rw} (group 6). Let now $\mathbf{x}$ be the symbol that is currently being scanned on the second tape. If $\mathbf{x} = *$, then the previous state must have been q_r. Then also the previous configuration is clearly uniquely determined. Otherwise, let $\mathbf{x} = (p, \eta, \sigma)$ and σ be the symbol that would be scanned on the first tape if the cursor were moved one step in direction $-d$. Then we must check if $\delta(p, \eta) = (q, \sigma, d)$. If the answer is positive, then case 6) is excluded (machine would have gone to a state in Q_r), and 2) is the only possibility. In that case, the previous state must be p_w. Also the previous configuration is easy to construct. If $\delta(p, \eta) \neq (q, \sigma, d)$, then the previous state must be q_{rw} and again, the previous configuration is clearly unique.

In the second case, we study a configuration with state $q_w \in Q_w$. Here the only possibility for the previous state is q and the previous configuration is easy to construct.

In the third case, we have state $p_r \in Q_r$. The possible previous states are either a state in Q_w (group 3) or a state in Q_{rw}. Then let $\mathbf{x}$ be the symbol on the second state that is on the right hand side of the currently scanned one. If $\mathbf{x} \neq *$, then 6) is excluded and 3) remains. In case 3) the previous configuration is easy to tell, so suppose that $\mathbf{x} = *$. Let then η be the symbol scanned on the first tape. Then find out $\delta(p, \eta) = (q, \sigma, d)$. The previous configuration is obtained by writing (q, σ, d) instead of $*$, σ instead of η and setting the state into q_{rw}.

In the last case, the current state is $q_{rw} \in Q_{rw}$. The only possible previous states are q_r (group 4) and q_{rw} (group 7). These are easy to distinguish by looking at the symbol $\mathbf{x}$ scanned on the second state. If $\mathbf{x} = *$, then the previous configuration must equal to the current one. If $\mathbf{x} \neq *$. The previous configuration is easy to tell.

It should be emphasized that group 7) is here only for a technical reason; to make δ' completely specified. It is clear that no such transition appears whatever the initial configuration is and no such configuration can be entered from any of the other forms, but these are fixed points of the time evolution mapping. □

Now any reversible two-tape Turing machine can be simulated with a two-tape quantum Turing machine by defining the transition amplitude function by an analogue of lemma II.4.4. It is a very well-known fact that any two-tape Turing machine can be simulated with a one-tape machine with quadratic loss of efficiency [20, p.30], but it remains an open question whether this simulation can be done reversibly in general. I would like to conjecture that there are two-tape machines that can not be simulated reversibly. *We will therefore consider a two-tape Turing machine as a basic model of quantum computation.* From lemma II.6.9 and the observation above we conclude

Theorem II.6.10. *Any recursively enumerable language can be recognized by a quantum Turing machine.*

Also the converse is true, any language recognized by a two-tape quantum Turing machine can also be recognized by an ordinary Turing machine. This is quite clear: One can describe a Turing machine that exhausts the computation tree of a QTM level by level thus simulating the QTM however losing exponentially in efficiency. There is still a technical question how an ordinary Turing machine can keep record of the amplitudes, but this is easy to solve: for an amplitude $re^{i\pi\theta}$ the machine remembers only rational numbers r^2 and θ. It is then clear how to multiply those amplitudes and how to get an arbitrarily strict approximation.

Of course, the converse statement can also be confirmed by the Church-Turing thesis. We get

Theorem II.6.11. *Let* **QRE** *be the family of formal languages accepted by the quantum Turing machines with probability* $1-\epsilon$, *where* $\epsilon \in (0,1)$ *is a constant. Then* **QRE** $=$ **RE**.

III. Finite quantum automata

III.1. Motivation

The definition of a QTM is very formal and, as the classical Turing machine, a QTM is really difficult to design for a special algorithmic problem. Moreover, it would be convenient to extend the concepts of gates and circuits to quantum computation, since building a complex machine becomes much easier if one can operate with simple basic units. Moreover, an automaton having an *actually* infinite workspace does not seem to be realizable. In addition to this, a finite quantum automaton corresponds better to a physical object than a quantum Turing machine.

The simulation of a Turing machine with a finite model gives rise to the definition of a finite quantum automaton. Actually, this definition will lead us to a somehow more "sophisticated" model of quantum computation. We are also able to formulate the termination of the computation in a natural way.

III.2. Quantum digits, registers and networks

In this section we will generalize the concept of classical bits and registers into quantum computation.

Let $\Sigma = \{\sigma_1, \ldots \sigma_d\}$ be a finite alphabet and $\mathcal{H}_d$ the d-dimensional complex Hilbert space with orthonormal basis

$$\{|\,\sigma_1\rangle, \ldots, |\,\sigma_d\rangle\}.$$

The set of unit length elements in $\mathcal{H}_d$ is denoted by $\mathcal{H}_d^1$.

Definition III.2.1. A *d-ary quantum bit* q is a mapping $q : \mathbb{Z} \to \mathcal{H}_d^1$. The value $q(t)$ is called the *value of the quantum bit at time* t. We use here domain $\mathbb{Z}$ instead of $\mathbb{R}$ to indicate that the computation process is performed in discrete steps. Binary quantum digits are called *qubits*.

Definition III.2.2. A *d-ary quantum register* of length m is a mapping

$$q_m : \mathbb{Z} \to \underbrace{\mathcal{H}_d^1 \otimes \ldots \otimes \mathcal{H}_d^1}_{m \text{ factors}} = \mathcal{H}_{d^m}^1$$

Space $\mathcal{H}_{d^m}$ has orthonormal basis

$$\{|\,\sigma_{i_1}\rangle \otimes \ldots \otimes |\,\sigma_{i_d}\rangle \mid \sigma_{i_j} \in \Sigma\},$$

so we can identify a d-ary quantum register of length m by a d^m-ary quantum digit defined on alphabet Σ^m by

$$|\,\sigma_{i_1}\rangle \otimes \ldots \otimes |\,\sigma_{i_m}\rangle \mapsto |\,\sigma_{i_1} \ldots \sigma_{i_m}\rangle.$$

Definition III.2.3. A *quantum gate with m inputs and outputs* is a unitary mapping $U : \mathcal{H}_{d^m} \to \mathcal{H}_{d^m}$ or an observation "window" gate as the observation rule in the definition of a finite quantum automaton (in the next section). Notice that a quantum gate always has equal number of inputs and outputs. This may lead to idea that some of the input wires are thought to be "dummy" wires that are set to 0 in the beginning of the computation. Also some output wires may be ignored.

Definition III.2.4. A *quantum network* is an acyclic collection of quantum gates connected with wires. Some of the wires are specified to be *input wires* and *output wires*.

D. Deutsch has shown [10] that gates of form

$$D = \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & i\cos\theta & \sin\theta \\ 0 & 0 & \sin\theta & i\cos\theta \end{matrix} \end{pmatrix}$$

where $\mathbf{1}$ and $\mathbf{0}$ are 4×4 unity and zero matrices respectively and θ/π is irrational, are universal in the sense that all quantum networks can be constructed of them (to be precise, all quantum networks can be approximated by networks constructed of them with arbitrary precision). Matrix D operates on computational basis

$$\{|\,000\rangle, |\,001\rangle, |\,010\rangle, \ldots, |\,111\rangle\},$$

so the gate is a three-bit gate. Indeed, almost all quantum gates that operate on two or more bits are universal [11]. However, in this thesis we shall not concentrate on quantum networks but on finite quantum automata. Quantum networks are studied in more details in [4] and [26], for instance.

III.3. Finite quantum automata

Definition III.3.1. A *finite quantum automaton*, QFA for short, is a quintuple

$$\mathcal{F} = (\Sigma, \Gamma, M, W, R),$$

where Σ and Γ are alphabets of finite cardinalities d and d', M is *memory of constant size* K, W is the *workspace of constant size* L and R is a finite sequence consisting of the rules of computation. Memory M is the K-fold tensor product

$$M = \mathcal{H}_d \otimes \ldots \otimes \mathcal{H}_d$$

of the d-dimensional complex Hilbert space $\mathcal{H}_d$, that is a d^K-dimensional complex Hilbert space. Space $\mathcal{H}_d$ is assumed to be equipped with orthonormal basis

$$B = \{|\,\sigma_1\rangle, \ldots, |\,\sigma_d\rangle\}.$$

We call B the *computational basis* of $\mathcal{H}_d$. Similary, the workspace W is an L-fold tensor product of d'-dimensional vector space $\mathcal{H}_{d'}$ that is also assumed to have a fixed orthonormal basis.

The rules of the computation, R consists of a sequence whose each member is either a unitary mapping $U_i : M \otimes W \to M \otimes W$ or a subset $O_j \subseteq \{1, \dots, K\}$, an *observation rule.* The matrix elements of the unitary mappings are restricted by requirement 4 in the definition of quantum Turing machine (p. 19) and the last rule is usually assumed to be an observation rule. Moreover we assume that the unitary mappings and observation rules are enumerated independently and that the first member is not allowed to be an observation rule. Also an agreement must be made on computational rules that are unitary mappings $U_i : M \otimes W \to M \otimes W$. *We assume that each such rule is of form*

$$U_i : \mathcal{H}_d^{(1)} \otimes \mathcal{H}_d^{(2)} \otimes \mathcal{H}_d^{(3)} \otimes \mathcal{H}_{d'}^{(1)} \otimes \mathcal{H}_{d'}^{(2)} \to \mathcal{H}_d^{(1)} \otimes \mathcal{H}_d^{(2)} \otimes \mathcal{H}_d^{(3)} \otimes \mathcal{H}_{d'}^{(1)} \otimes \mathcal{H}_{d'}^{(2)},$$

$\mathcal{H}_d^{(1)}$, $\mathcal{H}_d^{(2)}$, $\mathcal{H}_d^{(3)}$ represent some three memory cells and $\mathcal{H}_{d'}^{(1)}$ and $\mathcal{H}_{d'}^{(2)}$ some two workspace cells. These cells does not need to be consequtive. So U_i is defined on three memory cells and two workspace cells. This assumption is made to obtain a reasonable implementation of the unitary evolution. Furthermore, according to Theorem II.5.1. mappings U_i can be represented as a product of $d^6 d'^4$ unitary matrices acting only on two-dimensional subspaces. Indeed, this assumption seems to make finite quantum automata equivalent to a reasonable quantum circuit model based on three-bit quantum gates.

Definition III.3.2. A *configuration* $\mathbf{c}$ of a finite quantum automaton is a vector in $M \otimes W$ of form

$$|\,\sigma_{i_1}\rangle \otimes |\,\sigma_{i_2}\rangle \otimes \ldots \otimes |\,\sigma_{i_K}\rangle \otimes |\,\gamma_{j_1}\rangle \otimes \ldots \otimes |\,\gamma_{j_L}\rangle.$$

The configurations clearly form an orthonormal basis of $M \otimes W$. A *superposition of configurations* is a vector in $M \otimes W$ of unit length. A superposition is *decomposable* if it can be written as a tensor product of vectors in $\mathcal{H}_d$ and $\mathcal{H}_{d'}$. Otherwise the superposition is *entangled.*

Definition III.3.3. The *computation* C of a finite quantum automaton is a finite sequence of superpositions of configurations determined by the rules of the computation. The first member of the computation, $\mathbf{c}_1$ is called the input and it is assumed to be a configuration of form

$$|\,\sigma_{i_1}\rangle \otimes \ldots \otimes |\,\sigma_{i_K}\rangle \otimes |\,*\rangle \otimes \ldots \otimes |\,*\rangle.$$

All other members are determined by the rules of the computation:

If $\mathbf{c}_i$ is the i:th member of C and U_i, the i:th member of R is a unitary mapping, then $\mathbf{c}_{i+1}$ is defined by $\mathbf{c}_{i+1} = U_i(\mathbf{c}_i)$. If the i:th member of R is an observation rule, then $\mathbf{c}_{i+1}$ is determined by the observation on the memory as in the definitions II.2.8 and II.2.9. The computation is said to halt at the last member of $\mathcal{C}$. The *time* used by a finite quantum automaton is defined to be the number of computational rules applied.

Theorem III.3.4. *Let $\mathcal{F}$ be a finite quantum automaton having no observation rules with memory and workspace of length l and 0 respectively, i.e. all operation of $\mathcal{F}$ is done via unitary mappings $U_i : M \to M$ and observations. Then there exists a two-tape quantum Turing machine $\mathcal{Q}$ that performs the same computation as $\mathcal{F}$ does, more precisely, if $\mathcal{F}$ beginning with configuration*

$$|\sigma_1\rangle \otimes \ldots \otimes |\sigma_l\rangle$$

ends in superposition

$$\alpha_1 |x_1\rangle + \ldots + \alpha_m |x_m\rangle$$

in N computational steps, then $\mathcal{Q}$ beginning with configuration

$$\ldots * * * q_0\sigma_1 \ldots \sigma_l * * * \ldots$$

ends up in superposition

$$\alpha_1 |q_f x_1\rangle + \ldots + \alpha_m |q_f x_m\rangle$$

using $\mathcal{O}(N^2)$ computational steps. Here each $x_i \in \Sigma^l$.

Proof. Let $d = |\Sigma|$. Each U_i acts only on three cells, which means that each U_i can essentially be represented by $d^3 \times d^3$ unitary matrix. U_i can be decomposed into d^6 unitary matrices each acting only on a two-dimensional subspace, so actually we can assume that each U_i acts only at two memory cells thus suffering only from constant slowdown.

We will first describe a one-tape quantum machine $\mathcal{Q}'$ without requirements 2 and 3. Machine $\mathcal{Q}'$ is constructed as follows: For any computational step U_i there will be states $(i, k, \sigma_1, \sigma_2, d)$ for each combination of $\sigma_1, \sigma_2 \in \Sigma \cup \{*\}$, $k \in \{0, 2, \ldots, l\}$ and $d \in \{r, w\}$ (we can suppose now that the alphabet of $\mathcal{F}$ does not contain symbol $*$). Consider a computational step like

$$U_i |\sigma_1\rangle \otimes |\sigma_2\rangle = \alpha_1 |\sigma_{11}\sigma_{12}\rangle + \ldots + \alpha_m |\sigma_{m1}\sigma_{m2}\rangle$$

where each $x_i \in \Sigma^2$ and U_i operates on cells t and u.

Machine $\mathcal{Q}'$ will perform this as follows: We suppose that the simulation of each computational step begins with the read-write head marker of the first tape $(i, 1, *, *, r)$ scanning the first symbol on the tape. Then the read-write head travels to the location t and remembers the symbol there, say σ_1 by going to state $(i, t, \sigma_1, *, r)$. Then the second symbol σ_2 at location u is gathered to obtain state $(i, u, \sigma_1, \sigma_2, r)$. Then the machine goes into states $(i, u, \sigma_{j1}, \sigma_{j2}, w)$ with amplitudes α_j, where $j \in \{1, 2, \ldots, m\}$. In each such state the machine writes σ_{j2} and σ_{j1} in positions u and t respectively when travelling back to the beginning and then goes to state $(i+1, 1, *, *, r)$. Finally, when the last rule is performed, the machine goes into state q_f. The initial state will be $q_0 = (1, 1, *, *, r)$.

Now the unitarity of matrices U_i will guarantee that requirement 1 in Definition II.6.8 (p. 35) will hold for all pairs (q, σ) for which the transition is defined. For all other pairs the transition may be defined in any way preserving property 1; this is always obtained by giving deterministic definition. This machine is then converted into a two-tape machine as in lemma II.6.9 to force the requirement 2 in definition II.6.8. $\square$

Naturally theorem III.3.4 offers no useful method for finding any quantum Turing machine. Actually, in the construction the number of the states of the machine will be terrible. A corresponding reverse of theorem III.3.4 is more obvious:

Theorem III.3.5. *Assume that a two-tape quantum Turing machine Q satisfies the conditions of theorem II.4.3. Then there exists a finite quantum automaton that performs the same computation with quadratic loss of efficiency.*

Proof. This sounds like only a restatement of theorem II.4.3, since the operation of a QTM is local: It acts only on at most three cells. There is still a problem: A configuration of a quantum Turing machine may during the computation develop into a superposition

$$\alpha_1 T_1 + \ldots \alpha_m T_m$$

in such a way that the read-write -head travels in configurations T_i in totally different places. Then we do not know which is the low-dimensional subspace of the finite model that we should act upon with the induced unitary mapping. The problem can be although avoided by requiring that the string in the tape is always marked with a special endmarker at both ends, and the original computation is simulated by using a special "quasi-marker" of the read-write head. The read-write-head on each computational step then travels the string from the beginning to the end and at the first travel gathers the information, and on the second travel performs the computational step. This requires only quadratic slowdown. $\square$

We shall reconsider theorems III.3.4 and III.3.5. They indeed say that *quantum Turing machines and finite quantum automata are equally powerful devices in the field of quantum computation.* This is not clear, since a QFA can handle only a finite number of inputs. We shall therefore consider that *for any QTM there exists a family of QFA's* (not finite) that can, all together, perform the computation of the QTM.

III.4. Quantum parallelism

Suppose that we have a computable (i.e. recursive) function $f:\mathbb{Z}\to\mathbb{Z}$. Consider then the restriction

$$f:\{0,1,\dots,2^m-1\}\to\{0,1,\dots,2^n-1\}.$$

All numbers in the domain and range can be represented by using m- and n-bit registers respectively. If $n<m$, the restricted function f is not injective, and computation therefore cannot be reversible. This problem can however be solved by allowing extra workspace large enough: By theorem II.6.7 there exists a two-tape quantum Turing machine that can compute f in the first tape and store the "history" of the computation on the second tape. We let S be the set of those inputs where any number from set $\{0,1,\dots,2^m-1\}$ is printed on the first tape, and on the second tape there are enough blank symbols to complete all computations of f, but then "illegal" non-blank symbol to guarantee that the computation uses only finitely many squares on both tapes. Consequently there exists a finite quantum automaton that can compute function f on memory of size m and using a finite workspace (which may be very large). The output is then read on n first qubits.

Assume now that there is some nontrivial property of f of great interest. Depending on the property, usully one has to find out all possible values of f and to do this, perform 2^m-1 computations. It is also usually assumed that f is easily (in polynomial time) computable, but the number of possible values grows exponentially in m, so the problem lacks practical solvability when m is big enough.

In quantum computation we can however take advantage of quantum superposition: Suppose that we have a quantum automaton computing a function

$$f:\{0,1,\dots,2^m-1\}\to\{0,1,\dots,2^m-1\}.$$

By our convention this means that there exists a unitary mapping $U:M\otimes W\to M\otimes W$ such that

$$U\left(x\otimes|*\rangle\otimes\ldots\otimes|*\rangle\right)=|f(x)\rangle\otimes|w\rangle$$

and U can be decomposed into unitary matrices each acting only on three memory cells.

We will now extend this quantum automaton by adding m computational rules to be performed before U. The input of the new automaton is assumed to be constant

$$|0\rangle\otimes\ldots\otimes|0\rangle\otimes|*\rangle\otimes\ldots\otimes|*\rangle.$$

The new rules added are of form

$$U_i=I\otimes\ldots\underbrace{\otimes A\otimes}_{i\text{:th position}}\ldots\otimes I\otimes\underbrace{I\otimes\ldots\otimes I}_{\substack{\text{the cells of}\\ \text{the workspace}}},$$

where A is a unitary matrix given by

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

Mapping with $U_1 \ldots, U_m$ transforms the first m qubits in the state $\frac{1}{\sqrt{2}} \mid 0\rangle + \frac{1}{\sqrt{2}} \mid 1\rangle$, equally weighted superposition of $\mid 0\rangle$ and $\mid 1\rangle$, and the register contains

$$\begin{aligned} & \frac{1}{\sqrt{2}}(\mid 0\rangle + \mid 1\rangle) \otimes \frac{1}{\sqrt{2}}(\mid 0\rangle + \mid 1\rangle) \otimes \ldots \otimes \frac{1}{\sqrt{2}}(\mid 0\rangle + \mid 1\rangle) \otimes \mid *\rangle \otimes \ldots \otimes \mid *\rangle \\ = \quad & \frac{1}{\sqrt{2^m}} (\mid 0\rangle + \mid 1\rangle + \ldots + \mid 2^m - 1\rangle) \otimes \mid *\rangle \otimes \ldots \otimes \mid *\rangle, \end{aligned}$$

but this is, ignoring the workspace, equally weighted superposition of $\mid 0\rangle, \mid 1\rangle, \ldots, \mid 2^m - 1\rangle$. Because of linearity, the unitary mapping U yields now

$$\frac{1}{\sqrt{2^m}} (\mid f(0)\rangle \otimes \mid w_0\rangle + \mid f(1)\rangle \otimes \mid w_1\rangle + \ldots + \mid f(2^m - 1)\rangle \otimes \mid w_{2^m-1}\rangle), \quad (3\text{-}1)$$

which is an equally weighted superposition of *all values* of f. This phenomenon is called *quantum parellelism.* The effect of the workpace can be ignored by performing the computation of f backwards modified so that the value of f is first saved in an auxiliary register. Therefore, from this on, the workspace is usually not written explicitly.

However, observing the state of (3-1) yields only one of possible values chosen equiprobably, and *all the other information is lost*, as discussed in the chapter II. Thus it is worth emphasizing that the *quantum parallelism does not mean parallel computation.* However, some interesting information of function f can be found in the superposition (3-1). This will be seen in more details later.

III.5. The discrete Fourier transform

Here $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ is the residue class ring of n elements. It is well-known that the mapping $\mathrm{e} : \mathbb{Z}_n \to \mathbb{C}$, $\mathrm{e}(x) = \mathrm{e}^{\frac{2\pi i \cdot x}{n}}$ defines an additive character of $\mathbb{Z}_n$ and that all the additive characters are of form $\psi_a(x) = e(ax)$ (A good introduction to the theory of characters is offered in [17, ch. 7]). Moreover, if $a \neq b$, then $\psi_a \neq \psi_b$. Very well-known are also the *orthogonality relations*

$$\sum_{a \in \mathbb{Z}_n} \mathrm{e}(ax) = \begin{cases} 0, & \text{if } x \neq 0 \\ n, & \text{if } x = 0. \end{cases}$$

Definition III.5.1. Let $A : \mathbb{Z}_n \to \mathbb{C}$ be any function. The *discrete Fourier transform* of A is a function $\mathcal{F}A : \mathbb{Z}_n \to \mathbb{C}$ defined by

$$\mathcal{F}A(x) = \frac{1}{\sqrt{n}} \sum_{y \in \mathbb{Z}_n} A(y)\mathrm{e}(xy).$$

The *inverse discrete Fourier transform* of A is a function $\mathcal{F}^{-1}A : \mathbb{Z}_n \to \mathbb{C}$ defined by

$$\mathcal{F}^{-1}A(x) = \frac{1}{\sqrt{n}} \sum_{y \in \mathbb{Z}_n} A(y)\mathrm{e}(-xy).$$

Using the orthogonality relations it is easy to verify that $\mathcal{F}^{-1}\mathcal{F}A = \mathcal{F}\mathcal{F}^{-1}A = A$. The discrete Fourier transform can also be defined for finite fields in an analogous way. It is obvious that in fields with characteristic two we have $\mathcal{F}^{-1} = \mathcal{F}$.

In the next example we will illustrate how the Fourier transform extracts some information about the periodicity.

Example. Let $A : \mathbb{Z}_{mn} \to \mathbb{C}$ be defined by $A(a) = \sin \frac{2\pi n a}{mn}$, where a is any representative of the coset $a + mn\mathbb{Z}$. By the periodicity of sin, A is a well-defined function and has period m. Straightforwardly one can compute

$$\begin{aligned} \mathcal{F}A(a) &= \frac{1}{\sqrt{mn}} \sum_{b=0}^{mn-1} \mathrm{e}^{\frac{2\pi i a b}{mn}} \sin \frac{2\pi n b}{mn} \\ &= \frac{i}{2\sqrt{mn}} \Big(\sum_{b=0}^{mn-1} \mathrm{e}^{\frac{2\pi i (a-n) b}{mn}} - \sum_{b=0}^{mn-1} \mathrm{e}^{\frac{2\pi i (a+n) b}{mn}} \Big) \end{aligned}$$

Using the orthogonality relations of characters we get the final result

$$\mathcal{F}A(a) = \begin{cases} \frac{i\sqrt{mn}}{2} & \text{if } a = n \\ -\frac{i\sqrt{mn}}{2} & \text{if } a = -n \\ 0 & \text{otherwise.} \end{cases}$$

We see that $\mathcal{F}A$ equals to zero everywhere else but at the "needle points" $\pm n$. This tells us that the function A is formed of a very simple periodic function.

We will consider now $\mathbb{F}_2^m$, m-dimensional vector space over binary field. We establish mapping $\mathbb{F}_2^m \to \mathbb{Z}$ by

$$(x_{m-1}, \ldots, x_1, x_0) \to x'_{m-1} \cdot 2^{m-1} + \ldots + x'_1 \cdot 2 + x'_0,$$

where $x'_i = 0$ if x_i is the zero element of $\mathbb{F}_2$ and 1 otherwise. We can thus consired elements of $\mathbb{F}_2^m$ as integers in the interval $[0, 2^m - 1]$. It is easy to see that $\psi_{\mathbf{y}}(\mathbf{x}) = (-1)^{\mathbf{x} \cdot \mathbf{y}}$ gives all the characters of $\mathbb{F}_2^m$, where $\mathbf{x} \cdot \mathbf{y}$ means the standard inner product in $\mathbb{F}_2^m$.

The discrete Fourier transform takes now form

$$\mathcal{F}A(\mathbf{x}) = \frac{1}{\sqrt{2^m}} \sum_{\mathbf{y}\in\mathbb{F}_2^m} A(\mathbf{y})(-1)^{\mathbf{x}\cdot\mathbf{y}}.$$

This is called the *Fourier-Hadamard transform.*

The discrete Fourier transform reveals an important property of a function, namely the approximate Fourier coefficients. Therefore, $\mathcal{F}A$ is often called the *spectrum* of A.

Problem III.5.2. (Fourier-Hadamard sampling problem) [3]. *Suppose that a mapping $A : \mathbb{F}_2^m \to \mathbb{C}$ that satisfies $\sum_{\mathbf{x}\in\mathbb{F}_2^m} |A(\mathbf{x})|^2 = 1$ is given (again we consider elements of $\mathbb{F}_2^m$ as integers in $[0, 2^m - 1]$). Sample from $\mathbb{F}_2^m$ so that any $\mathbf{x}$ is found with probability $|\mathcal{F}A(\mathbf{x})|^2$.*

Claim: With a quantum automaton the Fourier-Hadamard sampling problem can be solved in time $\mathcal{O}(m)$ assuming that the values of A are given in a very tricky way, namely encoded in the weights of the state vector.

Proof of the claim. The idea is introduced in [12]. This encoding will be used later in the factorization problem. Note that if, in the classical computation, all the values of $\mathcal{F}A$ are computed, it requires at least 2^m operations.

We begin with a quantum register of length m prepared in superposition of all $x \in \{0, 1, \ldots 2^m - 1\}$ weighted with values of A:

$$|\varphi\rangle = A(0)\,|0\rangle + A(1)\,|1\rangle + \ldots + A(2^m - 1)\,|2^m - 1\rangle.$$

Let $A : \mathcal{H}_2 \to \mathcal{H}_2$ be as before:

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

It is an easy task to show by induction that applying unitary operator A to each qubit of the register we get

$$\underbrace{A \otimes \ldots \otimes A}_{m \text{ times}} |\mathbf{x}\rangle = \frac{1}{\sqrt{2^m}} \sum_{\mathbf{y}\in\mathbb{F}_2^m} (-1)^{\mathbf{x}\cdot\mathbf{y}}\,|\mathbf{y}\rangle.$$

Applying A to each qubit of φ we thus obtain

$$\sum_{\mathbf{y}\in\mathbb{F}_2^m} \Big(\frac{1}{\sqrt{2^m}} \sum_{\mathbf{x}\in\mathbb{F}_2^m} A(\mathbf{x})(-1)^{\mathbf{x}\cdot\mathbf{y}} \Big)\,|\mathbf{y}\rangle = \sum_{\mathbf{y}\in\mathbb{F}_{2m}} \mathcal{F}A(\mathbf{y})\,|\mathbf{y}\rangle.$$

Now observation of the register yields $\mathbf{x}$ with probability $|\mathcal{F}A(\mathbf{x})|^2$. The computational time needed is clearly proportional to m. $\square$

We will now return to the ordinary discrete Fourier transform on $\mathbb{Z}_n$ which takes form

$$\mathcal{F}A(x) = \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} \mathrm{e}^{\frac{2\pi i\cdot xy}{2^m}} A(y)$$

and see how this can be performed with a quantum automaton. This is introduced in [13]. We assume that the input sequence is encoded in the weights of superposition of a register of length m, i.e. initially register contains

$$A(0)\,|0\rangle + A(1)\,|1\rangle + \ldots + A(2^m-1)\,|2^m-1\rangle.$$

The desired outcome is

$$\frac{1}{\sqrt{2^m}}\big(\mathcal{F}A(0)\,|0\rangle + \mathcal{F}A(1)\,|1\rangle + \ldots + \mathcal{F}A(2^m-1)\,|2^m-1\rangle\big).$$

Clearly it suffices to establish a unitary mapping that performs the operation

$$U(|a\rangle) = \frac{1}{\sqrt{2^m}}\sum_{b=0}^{2^m-1} \mathrm{e}^{\frac{2\pi i\cdot ab}{2^m}}\,|b\rangle.$$

For convenience we denote the Hilbert space representing the register by

$$\mathcal{H} = \mathcal{H}_{m-1}\otimes\mathcal{H}_{m-2}\otimes\ldots\otimes H_0,$$

and we assume that the most significant bit is represented by $\mathcal{H}_{m-1}$. We define mapping $A_i : \mathcal{H}\to\mathcal{H}$ for each $i\in\{0,\ldots,m-1\}$ by

$$A_i = I\otimes\ldots\underbrace{\otimes A\otimes}_{i\text{:th position}} I\otimes\ldots\otimes I$$

and mappings $B_{jk} : \mathcal{H}_j\otimes\mathcal{H}_k\to\mathcal{H}_j\otimes\mathcal{H}_k$ for $0\le j<k\le m-1$ by

$$B_{jk} = \begin{pmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & \mathrm{e}^{\frac{\pi i}{2^{k-j}}}\end{pmatrix}.$$

Matrices B_{jk} are written in the basis $\{|0\rangle\otimes|0\rangle, |0\rangle\otimes|1\rangle, |1\rangle\otimes|0\rangle, |1\rangle\otimes|1\rangle\}$. In other words, bits only at locations j and k are affected if they both are 1.

Lemma III.5.3. *Let*

$$U = \prod_{i=0}^{m-1} A_i \prod_{j=i+1}^{m-1} B_{i,j}$$

be a product of $\frac{m(m+1)}{2}$ *matrices and* $a_{m-1}2^{m-1}+a_{m-2}2^{m-2}+\ldots+a_0$ *be the binary representation of number* a. *Then*

$$U(|a\rangle) = \frac{1}{\sqrt{2^m}}\sum_{b=0}^{2^m-1} e^{\frac{2\pi i\cdot ab}{2^m}}\,|b_0\rangle\otimes|b_1\rangle\otimes\ldots\otimes|b_{m-1}\rangle,$$

where $b_0 b_1 \dots b_{m-1}$ *is the inverse binary representation of number* $b = b_0 2^{m-1} + b_1 2^{m-2} + \dots + b_{m-1}$.

Proof. We prove the claim by induction on m. When $m = 1$ the claim takes form

$$U(|\,0\rangle) = A_0(|\,0\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}(|\,0\rangle + |\,1\rangle)$$

and

$$U(|\,1\rangle) = A_0(|\,1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}(|\,0\rangle - |\,1\rangle)$$

that are straightforward to verify. We assume now that the claim holds for value $m-1$. In case 1° the least significant bit of a equals to zero. Then

$$U(|\,a\rangle) = (A_0 \prod_{j=1}^{m-1} B_{0,j})(\frac{1}{\sqrt{2^{m-1}}}\sum_{b=0}^{2^{m-1}-1} \mathrm{e}^{\frac{2\pi i \frac{a}{2} b}{2^{m-1}}}\,|\,b_0 b_1 \dots b_{m-2}\rangle \otimes |\,0\rangle),$$

where $b_0 b_1 \dots b_{m-2}$ is the inverse binary representation of b. Applying mappings $B_{0,j}$ and keeping in mind that the bit $m-1$ equals to zero this yields

$$\begin{aligned}
U(|\,a\rangle) &= A_0 \frac{1}{\sqrt{2^{m-1}}}\sum_{b=0}^{2^{m-1}-1}\Big(\mathrm{e}^{\frac{2\pi i a b}{2^m}}\,|\,b_0 b_1 \dots b_{m-2}\rangle \otimes |\,0\rangle\Big)\\
&= \frac{1}{\sqrt{2^{m-1}}}\sum_{b=0}^{2^{m-1}-1}\mathrm{e}^{\frac{2\pi i a b}{2^m}}\,|\,b_0 b_1 \dots b_{m-2}\rangle \otimes \Big(\frac{1}{\sqrt{2}}\,|\,0\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle\Big)\\
&= \frac{1}{\sqrt{2^m}}\Big(\sum_{b=0}^{2^{m-1}-1}\mathrm{e}^{\frac{2\pi i a b}{2^m}}\,|\,b_0 b_1 \dots b_{m-2}\rangle \otimes |\,0\rangle\\
&+ \sum_{b=0}^{2^{m-1}-1}\mathrm{e}^{\frac{2\pi i a b}{2^m}}\,|\,b_0 b_1 \dots b_{m-2}\rangle \otimes |\,1\rangle\Big)\\
&= \frac{1}{\sqrt{2^m}}\Big(\sum_{b=0}^{2^{m-1}-1}\mathrm{e}^{\frac{2\pi i a b}{2^m}}\,|\,b\rangle + \sum_{2^{m-1}}^{2^m-1}\mathrm{e}^{\frac{2\pi i a b}{2^m}}\,|\,b\rangle\Big) = \frac{1}{\sqrt{2^m}}\Big(\sum_{b=0}^{2^m-1}\mathrm{e}^{\frac{2\pi i a b}{2^m}}\,|\,b\rangle\Big)
\end{aligned}$$

which completes the case 1°. For case 2° we assume that the least significant bit of a equals to 1. Then

$$\begin{aligned}
U(|\,a\rangle) &= \Big(A_0 \prod_{j=1}^{m-1} B_{0,j}\Big)\Big(\frac{1}{\sqrt{2^{m-1}}}\sum_{b=0}^{2^{m-1}-1}\mathrm{e}^{\frac{2\pi i \frac{a-1}{2} b}{2^{m-1}}}\,|\,b_0 b_1 \dots b_{m-2}\rangle \otimes |\,1\rangle\Big)\\
&= A_0 \prod_{j=1}^{m-1} B_{0,j}\Big(\frac{1}{\sqrt{2^{m-1}}}\sum_{b=0}^{2^{m-1}-1}\mathrm{e}^{\frac{2\pi i a b}{2^m}}\mathrm{e}^{-\frac{2\pi i b}{2^m}}\,|\,b_0 b_1 \dots b_{m-2}\rangle \otimes |\,1\rangle\Big)
\end{aligned}.$$

The action of each $B_{0,j}$ depends now on the actual value of bits b_j. We gain

$$
\begin{aligned}
U(|a\rangle) &= A_0 \frac{1}{\sqrt{2^{m-1}}} \sum_{b=0}^{2^{m-1}-1} \mathrm{e}^{\frac{2\pi iab}{2^m}} \mathrm{e}^{-\frac{2\pi ib}{2^m}} \mathrm{e}^{\frac{\pi ib_{m-2}}{1}} \cdot \ldots \cdot \mathrm{e}^{\frac{\pi ib_0}{2^{m-1}}} \, |b_0 b_1 \ldots b_{m-2}\rangle \otimes |1\rangle \\
&= A_0 \frac{1}{\sqrt{2^{m-1}}} \sum_{b=0}^{2^{m-1}-1} \mathrm{e}^{\frac{2\pi iab}{2^m}} \mathrm{e}^{-\frac{2\pi ib}{2^m}} \mathrm{e}^{\frac{2\pi i}{2^m}(2^{m-2}b_{m-2}+\ldots+b_0)} \, |b_0 b_1 \ldots b_{m-2}\rangle \otimes |1\rangle.
\end{aligned}
$$

The action of A_0 yields

$$
U(|a\rangle) = \frac{1}{\sqrt{2^m}} \sum_{b=0}^{2^{m-1}-1} \mathrm{e}^{\frac{2\pi i(ab-b)}{2^m}} \mathrm{e}^{\frac{2\pi i}{2^m}(2^{m-2}b_{m-2}+\ldots+b_0)} \, |b_0 b_1 \ldots b_{m-2}\rangle \otimes |0\rangle
$$

$$
-\frac{1}{\sqrt{2^m}} \sum_{b=0}^{2^{m-1}-1} \mathrm{e}^{\frac{2\pi i(ab-b)}{2^m}} \mathrm{e}^{\frac{2\pi i}{2^m}(2^{m-2}b_{m-2}+\ldots+b_0+2^{m-1}-2^{m-1})} \, |b_0 b_1 \ldots b_{m-2}\rangle \otimes |1\rangle,
$$

which takes form

$$
\begin{aligned}
U(|a\rangle) &= \frac{1}{\sqrt{2^m}} \sum_{b=0}^{2^{m-1}-1} \mathrm{e}^{\frac{2\pi i(ab-b+b)}{2^m}} \, |b_0 b_1 \ldots b_{m-2}0\rangle \\
&\quad - \frac{1}{\sqrt{2^m}} \sum_{b=2^{m-1}}^{2^m-1} \mathrm{e}^{\frac{2\pi i(ab-b+b-2^{m-1})}{2^m}} \, |b_0 b_1 \ldots b_{m-2}1\rangle \\
&= \frac{1}{\sqrt{2^m}} \sum_{b=0}^{2^{m-1}-1} \mathrm{e}^{\frac{2\pi iab}{2^m}} \, |b\rangle - \frac{1}{\sqrt{2^m}} \sum_{b=2^{m-1}}^{2^m-1} \mathrm{e}^{\frac{2\pi iab}{2^m}} \mathrm{e}^{-\frac{2\pi i 2^{m-1}}{2^m}} \, |b\rangle \\
&= \frac{1}{\sqrt{2^m}} \sum_{b=0}^{2^m-1} \mathrm{e}^{\frac{2\pi iab}{2^m}} \, |b\rangle,
\end{aligned}
$$

and the proof is complete. □

Notice that this quantum Fourier transform (QFT) deals with the coefficients of the quantum superposition, not with the actual information stored in quantum bits. Like Fourier-Hadamard transform, the QFT does not solve any classical problem.

III.6. Shor's algorithm for factoring numbers

It is presently believed that there exists no efficient classical algorithm for solving the following problem $\mathcal{F}$: Given $n = pq$, a product of two primes, find out p and q. Efficiency here means that the number of algorithmic steps should be polynomially bounded by binary representation of n. This problem is very important in cryptography, for instance, public key cryptosystem

RSA is based on the assumed non-existence of an efficient algorithm solving $\mathcal{F}$. Using RSA also requires an efficient method to solve the inverse problem $\mathcal{P}$: Given two primes p and q in binary representation, compute the binary representation of product pq. The latter problem is easily solved in time $\mathcal{O}(\max\{|p|, |q|\}^2)$.

Peter W. Shor descibed in 1994 [22] a probabilistic quantum algorithm solving $\mathcal{P}$ in polynomial time. The original idea for the factoring algorithm comes from D. Simon [23]. Shor's algorithm makes use of a quantum automaton. This algorithm (slightly modified) will be described in this section.

Divisibility by a prime number is a periodic property; every p:th number is divisible by p. It should not then appear as a surprise that solving $\mathcal{P}$ reduces to determining the period of a certain function. In what follows, the order of an element a in the multiplicative group $\mathbb{Z}_N^*$ will be denoted by $\mathrm{ord}_N(a)$.

Shor's division algorithm. The representation of [13] is here modified. The input is an odd composite natural number N which is not a pure prime power, and the aim is to find a nontrivial factor of N. By the size of the input we understand the length of the binary representation of N, which is $\lfloor \log_2 N + 1 \rfloor$.

1° Choose an arbitrary natural number $a < N$.

2° Compute $\gcd(a, N)$ (This can be done in cubic time with the classical Euclidean algorithm).

3° If $\gcd(a, N) > 1$, then it is a nontrivial factor of N. In this case output $\gcd(a, N)$ and stop. Otherwise continue at 4°.

4° Compute the period r of function $f_N(x) = a^x \pmod{N}$.

5° If r is odd or $a^{\frac{r}{2}} \equiv \pm 1 \pmod{N}$, then output "failure" and stop, else compute $c_\pm = \gcd(a^{\frac{r}{2}} \pm 1, N)$, output $c_\pm$ and stop.

Each step but 4° can be done with a classical computer in polynomial time. Note also that if N is even, the powers of 2 can be easily recognized and divided. There are also polynomial time classical probabilistic algorithms for recognizing whether a number is a pure prime power [27, p.41]. We will next show how to use a quantum automaton to perform 4° with a polynomial number of simple unitary transformations. First of all we shall estimate the probability of failure at step 5°.

Lemma III.6.1. *Let $N = p_1^{a_1} \cdot \ldots \cdot p_k^{a_k}$ be the decomposition of an odd integer into prime factors and a a random number such that $\gcd(N, a) = 1$. Then the probability $P(r = \mathrm{ord}_N(a)$ is even and $a^{\frac{r}{2}} \not\equiv \pm 1 \pmod{N})$ is greater than $1 - \frac{1}{2^{k-1}}$.*

Proof. [13] modified. We will demonstrate that inequality

$$P(r = \mathrm{ord}_N(a) \text{ is odd or } a^{\frac{r}{2}} \equiv \pm 1 \pmod N) \geq \frac{1}{2^{k-1}}$$

always holds when N is an odd integer. In fact, it suffices to reduce the latter condition into $a^{\frac{r}{2}} \equiv -1 \pmod N$ since r is the least positive number such that $a^r \equiv 1 \pmod N$. We first notice that equivalence

$$a^{\frac{r}{2}} \equiv -1 \pmod N \iff a^{\frac{r}{2}} \equiv -1 \pmod{p_i^{a_i}} \text{ for each } i$$

holds. But this is trivial since all numbers $p_i^{a_i}$ are coprime to each other. We denote $r_i = \mathrm{ord}_{p_i^{a_i}}(a)$ and notice that $r = \mathrm{lcm}(r_1, \dots, r_k)$. We also denote $r_i = 2^{t_i}s_i$ with s_i odd and

$$\begin{aligned} s &= \mathrm{lcm}(s_1, \dots, s_k) \\ t &= \max(t_1, \dots, t_k). \end{aligned}$$

Then it is easy to check that $r = 2^t s$, where s is odd. Next we show that the implications

$$a^{\frac{r}{2}} \equiv -1 \pmod{p_i^{a_i}} \implies t_i = t \tag{3-2}$$

also hold. Let us suppose, for the sake of contradiction, that $a^{\frac{r}{2}} \equiv -1$ but $t_i < t$. This means that 2 divides r at least one time more than it divides r_i, so $r_i \mid \frac{r}{2}$, because each s_i divides s. We can now write $\frac{r}{2} = r_i \cdot c$ and compute:

$$1 \equiv a^{r_i} \equiv a^{cr_i} = a^{\frac{r}{2}} \equiv -1 \pmod{p_i^{a_i}},$$

which is a contradiction, since no even prime factor was allowed in the decomposition of N.

We can conclude that also implications

$$r \text{ odd} \iff \text{each } r_i \text{ odd} \iff t_i = 0 \text{ for all } i$$

hold, and using (3-3) we see that

$$P(r \text{ odd or } a^{\frac{r}{2}} \equiv -1 \pmod N) \leq P(\text{all the numbers } t_i \text{ are equal}).$$

Finally we claim that the probability $P(t_i = j)$ cannot be greater than $\frac{1}{2}$ for any numbers i and j. In order to prove this claim we make use of the fact that any group of form $\mathbb{Z}_{p^a}^*$ is cyclic. Let us denote the generator of $\mathbb{Z}_{p^a}^*$ by γ. It follows from the assumption $p \neq 2$ that $|\mathbb{Z}_{p^a}| = \varphi(p^a) = p^{a-1}(p-1)$ is an even number. Let us denote this by $\varphi(p^a) = 2^n l$ with l odd. All elements in $\mathbb{Z}_{p^a}^*$ can be represented uniquely in the form γ^i, where $i \in \{1, 2, \dots, 2^n l\}$. Moreover we have

$$\mathrm{ord}_{p^a}(\gamma^i) = \frac{2^n l}{\gcd(i, 2^n l)},$$

so an order of form $2^m k$, where $2 \nmid k$ occurs if and only if

$$\gcd(i, 2^n l) = 2^{n-m} l,$$

and this happens if and only if $i = 2^{n-m} k$ where k is odd. But multiples of 2^{n-m} with an odd multiplier occur in the sequence 1, 2, ..., $2^n l$ at most

$$\frac{1}{2} \cdot \frac{2^n l}{2^{n-m}} = 2^{m-1} l \le \frac{|\mathbb{Z}_{p^a}^*|}{2}$$

times. As a conclusion, we can count the probability

$$\begin{aligned} P(\text{all the numbers } t_i \text{ are equal}) &= \sum_{j=0}^{n} \prod_{i=1}^{k} P(t_i = j) \\ &= \sum_{j=0}^{n} P(t_1 = j) \prod_{i=2}^{k} P(t_i = t_1) \\ &= \prod_{i=2}^{k} P(t_i = t_1) \sum_{j=0}^{n} P(t_1 = j) \\ &\le \prod_{i=2}^{k} \frac{1}{2} = \frac{1}{2^{k-1}}, \end{aligned}$$

which completes the proof. □

The next lemma is to show that the exclusion of the pure prime powers is really necessary.

Lemma III.6.2. *Let $N = p^\alpha$ be an odd prime power and a a number such that $p \nmid a$ and $r = \mathrm{ord}_N(a)$. Then either r is odd or $a^{\frac{r}{2}} \equiv -1 \pmod{N}$.*

Proof. Assume on the contrary that $r = 2r'$ is even and $a^{\frac{r}{2}} \not\equiv -1 \pmod{p^\alpha}$. In any case $\alpha^r \equiv 1 \pmod{p^\alpha}$, so

$$p^\alpha \mid (a^{\frac{r}{2}} + 1)(a^{\frac{r}{2}} - 1).$$

It follows from the hypothesis that p^α cannot divide either of the factors above, so p divides them both. Let $1 < n < \alpha$ be the greatest number such that $p^n \mid a^{\frac{r}{2}} - 1$. Then $a^{\frac{r}{2}} - 1 = p^n \cdot m$, and squaring this we get

$$a^r - 2a^{\frac{r}{2}} + 1 = p^{2n} \cdot m^2.$$

But this can be written as $a^r - 1 - 2(a^{\frac{r}{2}} - 1) = p^{2n} m^2$. But $a^r - 1$ is divisible by p^α, so $p^{\min\{2n,\alpha\}}$ divides $a^{\frac{r}{2}} - 1$, which contradicts the choice of n. □

We will now show how to perform step 4° by a quantum automaton. First we choose a number $M = 2^L$ such that $N^2 \le M \le 2N^2$ and use a quantum automaton whose memory size is $2L$. Then clearly L is of the same order as the size of the input, and we will show that the algorithm works in polynomial time with respect to L.

The memory is thought to be divided into two registers of length L, say q_1 and q_2. The input of the algorithm is a configuration

$$\mathbf{c} = \underbrace{|0\rangle \otimes \ldots \otimes |0\rangle}_{q_1} \otimes \underbrace{|n_1\rangle \otimes \ldots \otimes |n_L\rangle}_{q_2} \otimes \underbrace{|*\rangle \otimes \ldots \otimes |*\rangle}_{W}, \qquad (3\text{-}3)$$

where $n_1 \ldots n_L$ consists the binary representation of the arbitrarily chosen a and N. We ignore the workspace and denote this input by $|0\rangle \otimes |a, N\rangle$.

The first rules of the computation are as follows: Mapping A is used to all L qubits in q_1. The outcome is then

$$\frac{1}{\sqrt{M}}\big(|0\rangle \otimes |a, N\rangle + |1\rangle \otimes |a, N\rangle + \ldots + |M-1\rangle \otimes |a, N\rangle\big), \qquad (3\text{-}4)$$

thus q_1 contains the equally weighted superposition of all values in $\{0, 1, \ldots, M-1\}$. The next set of computational rules makes use of the reversible Turing machine $\mathcal{M}$ computing function $f(c, a, N) = a^c \pmod N$. Input c is stored in q_1, a and N in q_2. The result is stored in q_2. It is a well-known fact that there exists machine $\mathcal{M}$ working in time $\mathcal{O}(L)$ for this task. The space needed in this operation determines the size of the workspace. We know that there is a unitary mapping $U : M \otimes W \to M \otimes W$ perfoming the same computation. Ignoring the workspace, the contents of the memory is then

$$\frac{1}{\sqrt{M}}\Big(\sum_{c=0}^{M-1} |c\rangle \otimes |\overline{a^c}\rangle\Big), \qquad (3\text{-}5)$$

where $\overline{a^c}$ means $a^c \pmod N$. The next rule is an observation rule determined by set $\{L+1, \ldots, 2L\}$, i.e. an observation on the register q_2. If r is the order of a mod N, then the post-observation value of the memory is

$$\frac{1}{\sqrt{K+1}} \sum_{j=0}^{K} |jr + l\rangle \otimes |\overline{a^l}\rangle, \qquad (3\text{-}6)$$

where l has been chosen probabilistically in the observation procedure and K is the unique integer satisfying $M - r \leq l + Kr < M$ which is equivalent to $\frac{M-l}{r} - 1 \leq K < \frac{M-l}{r}$. For future use we still estimate the value of K. First we know that $K < \frac{M}{r} - \frac{l}{r}$. Writing this in form $K < \lfloor \frac{M}{r} \rfloor + \delta - \frac{l}{r}$ where $\delta = \frac{M}{r} - \lfloor \frac{M}{r} \rfloor \in [0, 1)$ and noticing that $\frac{l}{r} \in [0, 1)$ we see that inequality $K \leq \lfloor \frac{M}{r} \rfloor - 1$ must hold, since K is an integer. Therefore $K + 1 \leq \frac{M}{r}$. We can then write the state of the register q_1 as

$$\sum_{b=0}^{M-1} f(b)\, |b\rangle, \qquad (3\text{-}7)$$

where

$$f(b) = \begin{cases} \frac{1}{\sqrt{K+1}} & \text{if } b = jr + l \text{ for some } j \\ 0 & \text{otherwise.} \end{cases}$$

In other words, f is a periodic function with period r. However, the periodicity may be slightly ruined in the end of the "cycle" when we identify M and 0. We will now apply the discrete Fourier transform to get q_1 in the state

$$\sum_{b=0}^{M-1} \underbrace{\left(\frac{1}{\sqrt{M}} \sum_{a=0}^{M-1} f(a) \mathrm{e}^{\frac{2\pi i a b}{M}} \right)}_{\tilde{f}(b)} |b\rangle. \tag{3-8}$$

Furthermore, we can write

$$\tilde{f}(b) = \frac{1}{\sqrt{M}} \sum_{j=0}^{K} \frac{1}{\sqrt{K+1}} \mathrm{e}^{\frac{2\pi i (jr+l) b}{M}}. \tag{3-9}$$

Next we perform an observation at q_1. The probability of seeing b is given by

$$P(b) = \left| \frac{1}{\sqrt{M}} \sum_{j=0}^{K} \frac{1}{\sqrt{K+1}} \mathrm{e}^{\frac{2\pi i j r b}{M}} \mathrm{e}^{\frac{2\pi i b l}{M}} \right|^2 \tag{3-10}$$

which can be estimated by

$$P(b) = \frac{1}{M} \frac{1}{K+1} \left| \sum_{j=0}^{K} \mathrm{e}^{\frac{2\pi i j r b}{M}} \right|^2 \geq \frac{r}{M^2} \left| \sum_{j=0}^{K} \mathrm{e}^{\frac{2\pi i j r b}{M}} \right|^2. \tag{3-11}$$

We interrupt now the analysis of the general case and, in order to understand the situation better, consider a special case where r happens to divide M exactly. Then clearly $K = \frac{M}{r} - 1$ and (3-9) becomes

$$\tilde{f}(b) = \frac{1}{\sqrt{M}} \sum_{j=0}^{\frac{M}{r}-1} \frac{1}{\sqrt{\frac{M}{r}}} \mathrm{e}^{\frac{2\pi i (jr+l) b}{M}}, \tag{3-12}$$

and the probability of seeing value b becomes

$$\frac{1}{M} \frac{r}{M} \left| \sum_{j=0}^{\frac{M}{r}-1} \mathrm{e}^{\frac{2\pi i j r b}{M}} \mathrm{e}^{\frac{2\pi i l b}{M}} \right|^2 = \frac{r}{M^2} \left| \sum_{j=0}^{\frac{M}{r}-1} \mathrm{e}^{\frac{2\pi i b j}{M/r}} \right|^2 = \begin{cases} \frac{1}{r} & \text{if } b = k\frac{M}{r} \text{ for some } k \\ 0 & \text{otherwise.} \end{cases}$$

Thus the *observation of (3-8) can not yield any other values but* $k\frac{M}{r}$, *where* k *is chosen from the set* $\{0, 1, \ldots, r-1\}$. Moreover, any such k value will be obtained with probability $\frac{1}{r}$. Further, *if the observed value* $b' = k'\frac{M}{r}$

happens to satisfy $\gcd(k', r) = 1$, we can find r by cancelling $\frac{b'}{M} = \frac{k'}{r}$ into an irreducible fraction (recall that the values b' and M are now known). In the analysis of this special case it still remains (in addition to number-theoretical investigations) to show that the probability of observing b' such that $\gcd(k', r) = 1$ is great enough.

Instead of continuing the special case $r \mid M$ we return to the general case. Notice that in the special case only such values b for which $br - kM = 0$ for some $k \in \{0, 1, \ldots, r-1\}$ could be observed. In the general case we will ask what would be the probability of observing a b value that satisfies

$$|br - kM| \leq \frac{r}{2} \tag{3-13}$$

for some k. It is clear that for each $k \in \{0, 1, \ldots, r-1\}$ there exists unique $b \in \{0, 1, \ldots, M-1\}$ that satisfies the condition (3-13) and that different k values will yield different b values. It follows that as in the special case handled above, there will be r b values satisfying (3-13). The probability of seeing a number b that satisfies (3-13) becomes

$$P(b) \geq \frac{r}{M^2} \left| \sum_{j=0}^{K} \mathrm{e}^{\frac{2\pi i r b j}{M}} \right|^2 = \frac{r}{M^2} \left| \sum_{j=0}^{K} \mathrm{e}^{\frac{2\pi i \varphi_b j}{M}} \right|^2, \tag{3-14}$$

where we have denoted $\varphi_b = rb - kM$. Evaluating the exponential sum (3-14) we get a lower bound for the probability:

$$P(b) \geq \frac{r}{M^2} \frac{\sin^2 \frac{\pi \varphi_b}{M}(K+1)}{\sin^2 \frac{\pi \varphi_b}{M}} \tag{3-15}$$

Noticing that the lower bound (3-15) considered as a function of $\frac{\pi \varphi_b}{M}$ receives its local maximum at zero and according to the estimation (3-13) of φ_b the bound (3-15) gets the lowest value at the points $\varphi_b = \pm \frac{r}{2}$. Thus

$$P(b) \geq \frac{r}{M^2} \frac{\sin^2 \frac{\pi r}{2M}(K+1)}{\sin^2 \frac{\pi r}{2M}}. \tag{3-16}$$

We still need to estimate (3-16). From previous bounds for K we get $\frac{M}{r} - 1 < K + 1 \leq \frac{M}{r}$, therefore

$$\frac{\pi}{2}(1 - \frac{r}{M}) < \frac{\pi r}{2M}(K+1) \leq \frac{\pi}{2}.$$

Because $\frac{r}{M} \leq \frac{N}{M} \leq \frac{1}{\sqrt{N}}$, the numerator of (3-16) is very close to 1. For instance, if we require that $N > 2^{10}$, one can easily check that $\sin^2 \frac{\pi r}{2M}(K+1) \geq 1 - 10^{-10}$. Because of the same reason, $\frac{\pi r}{2M}$ is close to zero and the estimation $\sin x \leq x$ holds. Thus

$$P(b) \geq \frac{r}{M^2} \frac{1 - 10^{-5}}{(\frac{\pi r}{2M})^2} = \frac{4}{\pi^2} \frac{1}{r}(1 - 10^{-5}) > \frac{4}{10} \frac{1}{r}.$$

Since there are r different values of b that satisfy (3-13), the probability of observing one is greater than $\frac{2}{5}$.

The probability that an arbitrary number $k \in \{1, \dots, r\}$ satisfies $\gcd(k, r) = 1$ is clearly $\frac{\varphi(r)}{r}$, where φ is Euler's function. This probability can be estimated by inequality (see [15, p.267], for instance)

$$\frac{\varphi(r)}{r} \geq c \frac{1}{\log \log r},$$

where c is a constant independent of r. Thus the probability of having b that satisfies both (3-13) and $\gcd(k, r) = 1$ is greater than

$$\frac{2}{5} \frac{c}{\log \log r} \geq \frac{2c}{5 \log \log N}.$$

Both conditions (3-13) and $\gcd(k, r) = 1$ can be tested in cubic time with a classical algorithm, and the probability of failure is

$$1 - \frac{2c}{5 \log |N|}.$$

Assume that the probability of failure is less than ε after t independent repeats. A lower bound for t is obtained from inequality

$$\left(1 - \frac{2c}{5 \log |N|}\right)^t \leq \varepsilon. \tag{3-17}$$

Using estimate $\log(1 - x) \leq -x$ when x is close to zero we see that (3-17) holds when

$$t \geq \frac{5 \log |N| \log \varepsilon^{-1}}{2c},$$

so repeating the algorithm $\mathcal{O}(\log N \log \varepsilon^{-1})$ times we find a suitable b with probability at least $1 - \varepsilon$.

Finally, condition (3-13) can be written as

$$\left| \frac{b}{M} - \frac{k}{r} \right| \leq \frac{1}{2M}. \tag{3-18}$$

Now b and M are known, and there is exactly one fraction $\frac{k}{r}$ with denominator at most N in the range (3-18). This can be found efficiently by using the continued fraction expansion of $\frac{b}{M}$, see [15, ch. 10] or [17, ch. 10], for instance.

III.7. Discrete logarithm

Let $q = p^m$ be a fixed prime power and $\mathbb{F}_q$ the finite field of q elements. It is well-known that the multiplicative group of a finite field is cyclic, so let γ be a fixed generator of $\mathbb{F}_q^*$. All nonzero elements of $\mathbb{F}_q$ can then be uniquely represented as γ^i with $0 < i \leq q - 1$. Notice that here 1 is represented as γ^{q-1}, not as γ^0 as usually.

Definition III.7.1. Discrete logarithm in base γ is a mapping $\log_\gamma : \mathbb{F}_q^* \to \{1, 2, \ldots, q-1\}$ defined by $\log_\gamma(\gamma^i) = i$, where γ^i is the unique representation mentioned above.

A special case of discrete logarithm appears when $q = p$. Then $\mathbb{F}_p$ is a prime field, and the discrete logarithm can be seen as a permutation of set $\{1, 2, \ldots, p-1\}$, since this set can naturally be identified with the multiplicative group of $\mathbb{F}_p$ (this is why 1 is represented as γ^{p-1}). If a generator γ and another element $x \in \{1, 2, \ldots, p-1\}$ are given, the computational problem associated to the discrete logarithm is to find out an exponent $r \in 1, 2, \ldots, p-1$ such that $x = \gamma^r$. In number theory, this r is also called the *index of* x with respect to γ. Here the length of p is considered as the size of the input.

Presently many cryptosystems and protocols are based on the hypothesis that discrete logarithm is a computationally intractable problem, so any polynomial time algorithm for this problem would also be a breakthrough on cryptoanalysis.

In [22] Peter W. Shor also presented a polynomial time probabilistic quantum algorithm for discrete logarithm. The problem was divided in two cases, the case where $p-1$ is smooth turns out to be essentially easier (an infinite set of numbers is a *set of smooth numbers* if there is a constant c such that any number x in the set has no prime factors greater than $(\log x)^c$). The heart of the algorithm is to obtain a superposition of form

$$\frac{1}{(p-1)^2} \sum_{a=0}^{p-2} \sum_{b=0}^{p-2} \sum_{c=0}^{p-2} \sum_{d=0}^{p-2} \mathrm{e}^{\frac{2\pi i(ac+bd)}{p-1}} \, |\, c\rangle \otimes |\, d\rangle \otimes |\, \overline{\gamma^a x^{-p}}\rangle, \tag{3-19}$$

where $\overline{\gamma^a x^{-b}}$ means the least non-negative residue of $\gamma^a x^{-b}$ modulo p. In [22] it has been shown how to obtain (3-19) in polynomial number of unitary operations operating on two dimensions for *smooth* p, whereas the general case is based on finding a smooth q close enough to p and then beginning with superposition of form

$$\frac{1}{p-1} \sum_{a=0}^{p-2} \sum_{b=0}^{p-2} |\, a\rangle \otimes |\, b\rangle \, |\, \overline{\gamma^a x^{-b}}\rangle.$$

But it now remains unclear how to obtain this superposition if $p-1$ is not smooth. First, it is not realistic to allow the machine to start in such a superposition, since in a realization of a quantum computer we have, in the beginning of the computation, to ensure that the initial superposition was the one we wanted. This essentially requires an observation, which destroys the superposition. Therefore we shall agree that *any computation of a quantum automaton begins in a classical configuration.* We can then consider, for example a quantum Turing machine which, in the beginning goes into $p-1$ states $q_0, q_1, \ldots, q_{p-2}$ equally weighted and then writes i

on the tape in state q_i and continues computation. This quantum Turing machine could compute the discrete logarithm for only one number p and would require exponentially many states, which is undesirable. Next we will represent a modified version of Shor's algorithm for discrete logarithm, but first an easy estimation result is given.

Lemma III.7.2. *Suppose that complex numbers C_i lie on the unit circle and numbers δ_i are not far away from 1, say $|\delta_i - 1| \leq \delta \leq \frac{1}{2}$ for any δ_i. If additionally $\left|C_0 + \ldots + C_{2^L-1}\right| \geq 2^L\delta$, then*

$$\left|\frac{1}{2^{2L}}(C_0\delta_0 + \ldots + C_{2^L-1}\delta_{2^L-1})\right|^2 \geq \left|\frac{1}{2^{2L}}(C_0 + \ldots + C_{2^L-1})\right|^2 - \frac{\delta(2+\delta)}{2^{2L}}.$$

Proof. First, triangle inequality gives

$$\begin{aligned}
& \left|\frac{1}{2^{2L}}(C_0\delta_0 + \ldots + C_{2^L-1}\delta_{2^L-1})\right|^2 \\
\geq\ & \frac{1}{2^{4L}}(\left|C_0 + \ldots + C_{2^L-1}\right| - \left|(\delta_0 - 1)C_0 + \ldots + (\delta_{2^L-1} - 1)C_{2^L-1}\right|)^2,
\end{aligned}$$

and the right hand side still remains non-negative. This can be estimated by the Cauchy-Schwarz inequality to get

$$\begin{aligned}
& \frac{1}{2^{4L}}(\left|C_0 + \ldots + C_{2^L-1}\right| - \left|(\delta_0 - 1)C_0 + \ldots + (\delta_{2^L-1} - 1)C_{2^L-1}\right|)^2 \\
\geq\ & \frac{1}{2^{4L}}\Big(\left|C_0 + \ldots + C_{2^L-1}\right| \\
-\ & (\left|\delta_0 - 1\right|^2 + \ldots + \left|\delta_{2^L-1} - 1\right|^2)^{\frac{1}{2}}(\left|C_0\right|^2 + \ldots + \left|C_{2^L-1}\right|^2)^{\frac{1}{2}}\Big)^2 \\
\geq\ & \frac{1}{2^{4L}}\left(\left|C_0 + \ldots + C_{2^L-1}\right| - \sqrt{2^L\delta^2}\sqrt{2^L}\right)^2.
\end{aligned}$$

Since $\left|C_0 + \ldots + C_{2^L-1}\right| \leq 2^L$, the claim is obtained straightforwardly by developing the square in the last line. □

Shor's algorithm for discrete logarithm

Suppose that a prime number p and a fixed generator of $\mathbb{Z}_p$, say γ are given. Then, given an element $x \in \mathbb{Z}_p$ we should compute a number r such that $x \equiv \gamma^r \pmod{p-1}$ (notice that r is determined uniquely modulo $p-1$).

The algorithm is here described as a finite quantum automaton. First a number L is chosen such that $2^{L-1} < p - 1 \leq 2^L$. The automaton will use memory of length $3L+1$, and the initial configuration will be (workspace is not written, as usually)

$$|0\rangle \otimes |0\rangle \otimes |p\rangle,$$

where two first parts are L qubits and the last one $L+1$ qubits long. Then, unitary mapping

$$(A \otimes \ldots \otimes A) \otimes (A \otimes \ldots \otimes A) \otimes (I \otimes \ldots \otimes I),$$

where A is as in the previous subchapter is used to obtain superposition

$$\frac{1}{2^L} \sum_{a=0}^{2^L-1} \sum_{b=0}^{2^L-1} \mid a\rangle \otimes \mid b\rangle \otimes \mid p\rangle. \tag{3-20}$$

This will require time proportional to L. Exponentiation modulo p can be performed in cubic time, so there exists a unitary transformation U_E performing operation

$$U_E \mid a\rangle \otimes \mid b\rangle \otimes \mid p\rangle = \mid a\rangle \otimes \mid b\rangle \otimes \mid \overline{\gamma^a x^{-b}}\rangle$$

such that U_E can be decomposed into cubic number of unitary matrices each operating on a 6-dimensional subspace. The bar over means here the least positive residue modulo p. After operation of U_E, the discrete Fourier transform is done (in quadratic time) for both indices a and b to obtain

$$\frac{1}{2^{2L}} \sum_{a=0}^{2^L-1} \sum_{b=0}^{2^L-1} \sum_{c=0}^{2^L-1} \sum_{d=0}^{2^L-1} e^{\frac{2\pi i}{2^L}(ac+bd)} \mid c\rangle \otimes \mid d\rangle \otimes \mid \overline{\gamma^a x^{-b}}\rangle.$$

Then observation of all the $3L+1$ memory bits is done to see a triplet (c, d, γ^k). Remembering that $\gamma^a x^{-b} = \gamma^{a-rb}$ we can find out the probability P of seeing (c, d, γ^k) is

$$P = \left| \frac{1}{2^{2L}} \sum_{a=0}^{2^L-1} \sum_{\substack{b=0 \\ a-rb\equiv k \\ (\mathrm{mod}\ p-1)}}^{2^L-1} e^{\frac{2\pi i}{2^L}(ac+bd)} \right|^2.$$

Writing $a - rb = k - M(p-1)$ we see easily that M is either $\lfloor \frac{rb+k}{p-1} \rfloor$ or $\lfloor \frac{rb+r}{p-1} \rfloor - 1$ and the probability becomes

$$\begin{aligned} P &= \left| \frac{1}{2^{2L}} \sum_{b=0}^{2^L-1} \exp\Big(\frac{2\pi i}{2^L}(rbc + kc - M(p-1)c + bd)\Big) \right|^2 \\ &= \left| \frac{1}{2^{2L}} \sum_{b=0}^{2^L-1} \exp\Big(\frac{2\pi i}{2^L}(rbc + bd - M\overline{(p-1)c})\Big) \right|^2, \end{aligned} \tag{3-21}$$

where the bar now means the least non-negative residue modulo 2^L. Factor $e^{\frac{2\pi ikc}{2^L}}$ has been taken out and ignored, since it has absolute value one.

Consider now a special case where $p = 2^L + 1$. Then $\overline{c(p-1)} = 0$, and the probability becomes

$$P = \left| \frac{1}{(p-1)^2} \sum_{b=0}^{p-2} \exp\Big(\frac{2\pi i \cdot b}{p-1}(rc+d) \Big) \right|^2 ,$$

and we see that the exponential sum is now $p-1$ or 0 depending on if $rc + d \equiv 0 \pmod{p-1}$ or $rc + d \not\equiv 0 \pmod{p-1}$. Thus we can see only those states where $rc + d \equiv 0 \pmod{p-1}$ and each such state will be seen with probability $\frac{1}{(p-1)^2}$ (it is easy to see that there are actually $(p-1)^2$ possible triplets (c, d, γ^k) such that $rc + d \equiv 0$, so the probability of seeing one will add up to one). If the observed number c now satisfies $\gcd(c, p-1) = 1$, we can solve r in cubic time, $r \equiv -dc^{-1} \pmod{p-1}$. As in the factoring algorithm, the probability that $\gcd(c, p-1) = 1$ is estimated by using inequality

$$\frac{\varphi(p-1)}{p-1} \geq \frac{C}{\log\log(p-1)}$$

to ensure that a polynomial number of repeats is sufficient to give r with high probability.

From this on, we suppose that $p-1$ is not a power of 2. Then we could even modify the automaton with dropping one bit from the last register. We also have estimations $2^{L-1} + 1 < p < 2^L$. The first thing to do now is to write (3-21) in form where b can be separated:

$$\begin{aligned} P &= \left| \frac{1}{2^{2L}} \sum_{b=0}^{2^L-1} \exp\Big(\frac{2\pi i}{2^L}(rbc + bd - M\overline{(p-1)c} \right. \\ &\quad \left. - \frac{rb}{p-1}\overline{(p-1)c} + \frac{rb}{p-1}\overline{(p-1)c}) \Big) \right|^2 \\ &= \left| \frac{1}{2^{2L}} \sum_{b=0}^{2^L-1} \exp\Big(\frac{2\pi i b}{2^L} \underbrace{(rc + d - \frac{r}{p-1}\overline{c(p-1)})}_{\varphi} \Big) \right. \\ &\quad \left. \exp\Big(\frac{2\pi i}{2^L}(\frac{br}{p-1} - M)\overline{c(p-1)} \Big) \right|^2 . \end{aligned} \tag{3-22}$$

The idea will now begin to show: Term $\frac{br}{p-1}\overline{c(p-1)}$ was added and substracted, since M is expected to be close to $\frac{br}{p-1}$ and therefore, provided that $\overline{c(p-1)}$ is small enough, the factor

$$\exp\Big(\frac{2\pi i}{2^L}(\frac{br}{p-1} - M)\overline{c(p-1)} \Big)$$

is expected to be close to one. Then (3-22) can be estimated by using lemma III.7.2. If also the residue of $\varphi = rc + d - \frac{r}{p-1}\overline{c(p-1)}$ modulo 2^L having the least absolute value is small enough, a lower bound for P will be obtained.

Inspired by this, we say that triplet (c, d, γ^k) is *good*, if there is an integer K such that

$$\left| rc + d - \frac{r}{p-1}\overline{c(p-1)} - K2^L \right| \leq \frac{1}{2} \tag{3-23}$$

and if

$$\overline{c(p-1)} \leq \frac{1}{40\pi} 2^L. \tag{3-24}$$

A triplet that is not good is *bad*.

Then we will ask how many good triplets there are. First, we can see that for any c there is exactly one d such that (c, d, γ^k) is good, namely, d is obtained by rounding off $\frac{r}{p-1}\overline{c(p-1)} - rc$ to the nearest integer (it is easy to see that the case $\frac{r}{p-1}\overline{c(p-1)} = N_1 + \frac{1}{2}$ leads to the excluded case $p = 2^{L_1} + 1$) so the closest integer is unique.

We then find out how many c's there are such that (3-24) holds. To recover this we denote $p - 1 = 2^{L_1} N_1$, where N_1 is odd. Then it is easy to see that $\overline{c_1(p-1)} = \overline{c_2(p-1)}$ holds if and only if 2^{L-L_1} divides $c_1 - c_2$. Thus, there are $\frac{2^L}{2^{L-L_1}}$ c's that will map into the same $\overline{c(p-1)}$. On the other hand $\overline{c(p-1)} = cN_1 2^{L_1} - N2^L$ is always a multiple of 2^{L_1}, so the number of c's that satisfy (3-24) is at least $\frac{1}{40\pi}2^{L-L_1} \cdot 2^{L_1} = \frac{1}{40\pi}2^L$. Taking account of the last component we see that the total number of good triplets is at least $\frac{1}{40\pi}2^{2L}$.

Next task will be the estimation of the probability of observing a good triplet (c, d, γ^k). First estimate the expression

$$S = \left| \frac{1}{2^{2L}} \sum_{b=0}^{2^L - 1} \mathrm{e}^{\frac{2\pi i b}{2^L}\varphi} \right|^2,$$

which can be evaluated:

$$S = \frac{1}{2^{4L}} \frac{1 - \cos 2\pi\varphi}{1 - \cos \frac{2\pi\varphi}{2^L}}.$$

Since we assumed that (c, d, γ^k) is good, S gets the minimum at $\varphi = \frac{1}{2}$, so

$$S \geq \frac{1 - \cos\pi}{1 - \cos\frac{\pi}{2^L}} \geq \frac{1}{2^{4L}} \frac{4}{(\frac{\pi}{2^L})^2} = \frac{4}{\pi^2} \cdot \frac{1}{2^L}.$$

To ensure that the conditions of the lemma III.7.2 will hold we estimate

$$\begin{aligned}\left| \frac{br}{p-1} - M \right| &= \left| \frac{br}{p-1} - \frac{br+k}{p-1} + \frac{a}{p-1} \right| \\ &= \left| \frac{a-k}{p-1} \right| \leq \frac{2^L + p - 2}{p-1} < 2.\end{aligned}$$

Therefore,

$$\left|\frac{2\pi}{2^L}(\frac{br}{p-1}-M)\overline{c(p-1)}\right| \leq \frac{2\pi}{2^L}\cdot 2\cdot\frac{1}{40\pi}\cdot 2^L = \frac{1}{10},$$

so

$$\left|\exp(\frac{2\pi i}{2^L}(\frac{br}{p-1}-M)\overline{c(p-1)})-1\right| \leq \frac{1}{10} = \delta < \frac{4}{\pi^2},$$

so the conditions of lemma VII.3.2 hold. Therefore,

$$P \geq (\frac{4}{\pi^2}-\frac{1}{10}(2+\frac{1}{10}))\frac{1}{2^{2L}}.$$

But there are at least $\frac{1}{40\pi}2^{2L}$ good states, so the probability of seeing one is at least $\frac{1}{644}$. This bad estimation can of course improved with advanced estimation techniques, but here we are interested only of a positive lower bound.

It still remains to explain how we can recover r from a good triplet. First, denoting $c(p-1)-\overline{c(p-1)} = N2^L$, condidtion (3-23) can be rewritten as

$$-\frac{1}{2} \leq d + r\frac{N2^L}{p-1} - K2^L \leq \frac{1}{2}. \qquad (3\text{-}25)$$

Notice that after the observation we know c and N can be computed in cubic time. Since $p-1 < 2^L$, we obtain

$$K(p-1)-\frac{1}{2} < \frac{d}{2^L}(p-1)+rN < K(p-1)+\frac{1}{2}. \qquad (3\text{-}26)$$

Next we denote $\frac{d}{2^L} = \frac{x}{p-1}+\frac{\alpha}{p-1}$, where $|\alpha| \leq \frac{1}{2}$ (this x can be obtained by rounding $\frac{d(p-1)}{2^L}$ to the closest integer and therefore computed in quadratic time). We now claim that x, r and N satisfy $x+rN \equiv 0 \pmod{p-1}$, but this is easy to show: Substituting the representation of $\frac{d}{2^L}$ into (3-26) we get

$$K(p-1)-\frac{1}{2}-\alpha < x+rN < K(p-1)+\frac{1}{2}-\alpha.$$

This leads to $K(p-1)-1 < x+rN < K(p-1)+1$, so

$$x+rN = K(p-1) \equiv 0 \pmod{p-1}. \qquad (3\text{-}27)$$

Finally, if $\gcd(N, p-1) = 1$, we can solve r from congruence (3-27). But this depends totally on c, since $c(p-1) = N_c 2^L + \overline{c(p-1)}$. It is clear that if $c_1 \leq c_2$, then also $N_{c_1} \leq N_{c_2}$. On the other hand, the maximum value that N can get is the found by setting $c = 2^L - 1$. Then

$$N_{2^L-1} = \frac{(2^L-1)(p-1)-\overline{c(p-1)}}{2^L} \geq \frac{(2^L-1)2^{L-1}-(2^L-1)}{2^L} \geq 2^{L-1}-1,$$

so there are at least 2^{L-1} possible values of N ($c = 0$ will give 0). Therefore,

$$P(\gcd(N, p-1) = 1) \geq \frac{2\varphi(p-1)}{p-1} \geq \frac{C_1}{\log\log p}.$$

As a conclusion we see that the probability that we will get the discrete logarithm of x using the method described is bounded below by

$$P(\text{ succesful run }) \geq \frac{C}{\log |p|},$$

so repeating this algorithm a number proportional to the logarithm of the length of p times we will get the discrete logarithm of x with arbitrarily high probability.

IV. Quantum complexity theory

IV.1. Quantum complexity classes

In section I.4 we defined the concept of probabilistic Turing machine, but we did not define what is meant by the *acceptance* of a probabilistic automaton. In section I.5 one model was given: Language L is decided by a probabilistic Turing machine $\mathcal{M}$, if $\mathcal{M}$ gives the correct answer for any input with probability at least $\frac{2}{3}$. The class of languages decided by such probabilistic Turing machines was called **BPP**. In the definition of **BPP** the cases "the input word is in the language" and "the input word is not in the language" are treated symmetrically, so it should be clear that **BPP** is closed under complementation.

Other natural acceptance models can be defined for probabilistic Turing machines:

Definition IV.1.1. Let L be a language and $\mathcal{M}$ a probabilistic Turing machine that on any input halts in polynomial time. Machine $\mathcal{M}$ is a *Monte Carlo Turing machine* deciding L if for all inputs $\mathbf{x} \in L$ machine $\mathcal{M}$ gives answer "yes" with probability at least $\frac{1}{2}$ and for all inputs $\mathbf{x} \notin L$ machine gives answer "no" with probability 1. The set of languages decided by Monte Carlo Turing machines is denoted by **RP** (Randomized Polynomial time).

If a language L is in **RP**, we can always recognize with certainty if a word $\mathbf{x}$ is not in L, but the answer $\mathbf{x} \in L$ given by the machine may be erratic with probability at most $\frac{1}{2}$. However, repeating the algorithm k times, the probability of incorrect conclusion $\mathbf{x} \in L$ is reduced to be at most $\frac{1}{2^k}$. It is quite easy to conclude that $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$, but the definition of Monte Carlo Turing machine is unsymmetric with respect to acceptance. Therefore there is not any reason that **RP** should be closed under complementation.

As a natural analogy to the classical complexity classes we will define the corresponding quantum ones. However, due to the probabilistic nature of the quantum computation, the most meaningful complexity classes will be of semantic nature (see [20, p.255]).

Definition IV.1.2. Class **QRP** is defined as **RP**, but instead of a probabilistic Turing machine there will be a quantum Turing machine.

Definition IV.1.3. Class **co**$\mathcal{C}$, where $\mathcal{C}$ is any complexity class, consists of the complements of languages in $\mathcal{C}$.

Definition IV.1.4. Class **ZPP**, (Zero Probability of error Polynomial time), is given by $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$.

Suppose now, that $L \in \mathbf{ZPP}$. Then, by the definition of **RP** there exists a probabilistic Turing machine $\mathcal{M}_1$ that for any input $\mathbf{x}$ always detects if $\mathbf{x} \notin L$, and if $\mathbf{x} \in L$, then $\mathcal{M}_1$ answers "yes" with probability at least $\frac{1}{2}$. Moreover, by the definition of **coRP**, there exists a probabilistic Turing machine $\mathcal{M}_2$ that always detects the case $\mathbf{x} \in L$ and if $\mathbf{x} \notin L$, then $\mathcal{M}_2$ figures this out with probability not less than $\frac{1}{2}$. Running the machines $\mathcal{M}_1$ and $\mathcal{M}_2$ parallel and repeating the procedure until one of the machines gives answer "no" (which is known to be a correct answer) we will get answer to the query "$x \in L$"? with zero probability of error. It shall remain unknown how many repeats is required before the answer comes, but the expected number of the repeats can be calculated and is bounded by a polynomial.

Definition IV.1.5. If $\mathcal{C}$ is any classical complexity class determined by a class of probabilistic Turing machines, the class **Q**$\mathcal{C}$ shall denote the corresponding class determined by the quantum Turing machines. Class **QBPP** is usually denoted as **BQP** and **QP** as **EQP** to stand for "exact quantum polynomial time".

Ordinary Turing machines and nondeterministic Turing machines can also be seen as probabilistic ones. An ordinary Turing machine can be replaced with a probabilistic one with unambiguous transitions and acceptance mode "correct answer with probability 1 in all cases". On the other hand, a nondeterministic Turing machine can be replaced with a probabilistic one with acceptance mode as follows: For any "no"-instance reject with probability one and for any "yes"-instance accept with *nonzero probability*, no matter how low. Thus quantum analogies, for example, of **P** and **NP** can also be defined.

From the logical reversibility of the computation and form what is said above, it is clear that $\mathcal{C} \subseteq \mathbf{Q}\mathcal{C}$, whenever $\mathcal{C}$ is a classical complexity class including **P**. From this fact and the known inclusions of the classical complexity classes we are able to infer some relationships between the quantum complexity classes, but the most interesting issue in quantum complexity theory is how the quantum classes are related to classical ones. Presently very few facts are known about the quantum complexity classes at all, but some interesting results are collected in the section IV.3. Before this we will study a problem that was maybe the first example to suggest how powerful quantum computation might be.

IV.2. The problem of Deutsch and Jozsa

Here $\Sigma = \{0, 1\}$ will be the binary alphabet.

Definition IV.2.1. Let $X \subseteq \Sigma^*$ be a binary formal language. Language X is *balanced at length* n, if $|X \cap \Sigma^n| = 2^{n-1}$. Language X is *empty at length* n, if $X \cap \Sigma^n = \emptyset$. Moreover, we say that the *property* B *holds for* X if for each n the language X is either balanced or empty. If $f : \Sigma^* \to \Sigma$ is a function such that property B holds for language $f^{-1}(1)$, we also say that property B holds for f or that f satisfies property B.

Problem IV.2.2 (The problem of Deutsch and Jozsa). Let f be a recursive function that satisfies property B. Moreover, suppose that f is computable in time $\mathcal{O}(T(n))$. For given n, decide, whether $f^{-1}(1)$ is balanced or empty at length n. In other words, determine, if f gets value 1 for exactly half of the strings in Σ^n or if f is identically zero in Σ^n (one of these conditions will hold due to the assumption that f enjoys property B). The problem of Deutsch and Jozsa will be referred to as PDJ.

Altough unproven, it seems that solving the problem of Deutsch and Jozsa with certainty using the classical computation, will inherently take time $\Omega(2^{n-1}T(n))$. Quantum computation, however, offers a rapid solution.

Theorem IV.2.3. *We have* PDJ $\in$ $\mathbf{QTIME}(T(n) + n)$.

So theorem IV.2.3 asserts that PDJ can be solved in time $T(n) + n$ *with certainty* using a quantum computer.

Proof. The proof was introduced in [4], for example, and will be described by using a finite quantum automaton with memory of length $n+1$. In the very beginning, an equibalanced superposition of all strings in Σ^n is prepared, as in Shor's division algorithm, using time $\mathcal{O}(n)$:

$$\frac{1}{\sqrt{2^n}}\left(|\,0\rangle\otimes|\,0\rangle + |\,0\rangle\otimes|\,1\rangle + \ldots + |\,0\rangle\otimes|\,2^n - 1\rangle\right). \tag{4-1}$$

The elements of Σ^n will be interpreted as numbers in $\{0, 1, \ldots, 2^n - 1\}$ in the natural way. By the assumption, there exists a unitary matrix U_f that can be decomposed into a tensor product of $\mathcal{O}(T(n))$ unitary matrices acting only on two-dimensional subspaces and that the image will be $U_f(|\,0\rangle\otimes|\,x\rangle) = |\,f(x)\rangle\otimes|\,x\rangle$ (ignoring the workspace as usual). Thus we will get

$$\frac{1}{\sqrt{2^n}}\left(|\,f(0)\rangle\otimes|\,0\rangle + |\,f(1)\rangle\otimes|\,1\rangle + \ldots + |\,f(2^n-1)\rangle\otimes|\,2^n - 1\rangle\right). \tag{4-2}$$

Unitary transformation determined by matrix

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

works as $P \mid 0\rangle = \mid 0\rangle$, $P \mid 1\rangle = - \mid 1\rangle$, so P "encodes" the value of a qubit into the sign of the amplitude. Thus operating with P onto the first qubit of (4-2) will yield

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \mid f(x)\rangle \otimes \mid x\rangle. \tag{4-3}$$

We can then perform a "backwards" computation of U_f to erase all values $f(x)$ to obtain

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \mid 0\rangle \otimes \mid x\rangle$$

or ignoring the bit that represented $f(x)$,

$$C = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \mid x\rangle. \tag{4-4}$$

Application of $A \otimes \ldots \otimes A$ will now yield

$$C = \frac{1}{2^n} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} (-1)^{f(x)+x\cdot y} \mid y\rangle, \tag{4-5}$$

where $x \cdot y$ in the exponent stands for the inner product in $\mathbb{F}_2^n$ as in page 44. Suppose first that f is identically zero. Then we have

$$C' = \frac{1}{2^n} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} (-1)^{x\cdot y} \mid y\rangle = \frac{1}{2^n} 2^n \mid 0\rangle = \mid 0\rangle$$

by the orthogonality of characters (see section III.5). In this case, the observation of the registed will always yield $\mid 0\rangle$.

Assume then that there are sets $\mathcal{F}, \mathcal{T} \subset \{0, 1, \ldots, 2^n - 1\}$ of cardinality 2^{n-1} such that $f(x) = 1$ whenever $x \in \mathcal{T}$ and $f(x) = 0$ for each $x \in \mathcal{F}$. Then we get

$$\begin{aligned}
C' =& \frac{1}{2^n}\Big(\sum_{x\in\mathcal{F}} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y} \mid y\rangle - \sum_{x\in\mathcal{T}} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y} \mid y\rangle \Big) \\
=& \frac{1}{2^n}\Big(\sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y} \mid y\rangle - 2\sum_{x\in\mathcal{T}} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y} \mid y\rangle \Big) \\
=& \frac{1}{2^n}\Big(2^n \mid 0\rangle - 2\sum_{x\in\mathcal{T}} \mid 0\rangle - 2\sum_{x\in\mathcal{T}} \sum_{y=1}^{2^n-1} (-1)^{x\cdot y} \mid y\rangle \Big) \\
=& \frac{1}{2^n}\Big(2^n \mid 0\rangle - 2^n \mid 0\rangle - 2\sum_{x\in\mathcal{T}} \sum_{y=1}^{2^n-1} (-1)^{x\cdot y} \mid y\rangle \Big).
\end{aligned}$$

In this case, $\mid 0\rangle$ cannot be observed. □

IV.3. Oracle quantum computation

Definition IV.3.1. An *oracle Turing machine* $\mathcal{M}$ is a multitape Turing machine with a specific *query tape*, a special *query state* $q_?$ and two *answer states* q_{YES} and q_{NO}. An *oracle* A is a formal language. The action of an oracle Turing machine is described as the action of an ordinary machine, but whenever the machine goes into the query state, the next state will be either q_{YES} or q_{NO}, with all cursors unmoved, depending on whether the string on the query tape belongs to A or not. Thus the machine is supposed to get the answer to the query "$\mathbf{x} \in A$"? instantaneously. The corresponding oracle version for quantum Turing machines is defined analogously (the definition of a multitape Turing machine is a straightforward generalization of the two-tape machine).

A Turing machine $\mathcal{M}$ that is equipped with oracle A is denoted by $\mathcal{M}^A$. Moreover, for a complexity class $\mathcal{C}$ the notation $\mathcal{C}^A$ stands for the class of the languages that is decided by those machines deciding $\mathcal{C}$ equipped with oracle A.

Theorem IV.3.2. *There exists an oracle X for which there is a language L that can be recognized in linear time by a quantum Turing machine with zero error probability, but any deterministic Turing machine recognizing L consumes exponential time on all but finitely many inputs.*

Proof. Here $\Sigma = \{0,1\}$ will stand for the binary alphabet. Let function $\wp : \mathbb{N} \to \mathbb{N}$ be defined recursively by

$$\wp(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2^{\wp(n-1)} & \text{if } n > 1. \end{cases}$$

Let $\alpha : \mathbb{N} \to \mathbb{N}$. be an arbitrary recursive function such that for each i there are infinitely many integers n with $\alpha(n) = i$. For instance, a straightforward enumeration of the table

$$\begin{array}{lllll} 1 & & & & \\ 1 & 2 & & & \\ 1 & 2 & 3 & & \\ 1 & 2 & 3 & 4 & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

line by line gives one candidate which is easy to compute. Let also M_1, M_2, M_3, ... be a fixed enumeration of all deterministic oracle Turing machines. Oracle X on binary alphabet will be constructed in stages, in the beginning we set $X_1 = \emptyset$. At stage $n > 1$ machine $M_{\alpha(n)}$ will be simulated on input $1^{\wp(n)}$ for $2^{\wp(n)-1}$ steps, using X_n as oracle. If the machine $M_{\alpha(n)}$ has not stopped within $2^{\wp(n)-1}$ steps or stops in a rejecting state, then we define $R_n = \emptyset$. Otherwise, let Q_n be the set of queries asked during the simulation.

Then clearly $|Q_n| \leq 2^{\wp(n)-1}$, so Q_n contains at most half of the binary strings of length $\wp(n)$. Consequently, we can choose a set $R_n \subseteq \Sigma^{\wp(n)}$ such that $R_n \cap Q_n = \emptyset$ and $|R_n| = 2^{\wp(n)-1}$. In both cases, we define $X_{n+1} = X_n \cup R_n$. Oracle X is then defined as $X = \bigcup_{n \geq 1} X_n$. It is then clear that X satisfies property B.

Finally, we define

$$S_X = \{1^n \mid X \cap \Sigma^n = \emptyset\},$$

so S_X will contain strings of ones, and $1^n \in S_X$ if and only if there are no words of length n in X. Since X satisfies property B, language S_X can be recognized by a quantum Turing machine in linear time as follows: First a recursive function $f : \Sigma^* \to \Sigma$ is defined by an oracle machine such that $f(\mathbf{x}) = 1$ if and only if $\mathbf{x} \in X$. Computation of f thus takes at most linear time (it is a matter of taste wheter $\mathbf{x}$ is required to be written on the query tape). Moreover, $1^n \in S_X$ if and only if f is not balanced at length n. Thus, recognizing S_X turns into an instance of the problem of Deutsch and Jozsa which can be solved in linear time with a quantum Turing machine. Suppose then that S_X is decided by a deterministic oracle Turing machine M_i. Then, for all numbers n such that $\alpha(n) = i$, M_i will consume at least $2^{\wp(n)-1}$ steps at input $1^{\wp(n)}$. For details, see the proof of theorem IV.3.4. □

Corollary IV.3.3. *There exists an oracle X such that the inclusion $\mathbf{P}^X \subset \mathbf{QP}^X$ holds strictly.*

Proof. A polynomial cannot exceed an exponential function on more than finitely many positive integers. □

Theorem IV.3.4. *There exists an oracle Y for which there is a language that can be recognized in linear time by a quantum Turing machine with zero error probability, but any nondeterministic Turing machine that recognizes it takes exponentially many steps on infinitely many inputs.*

Proof. Let $N_1, N_2, N_3, \ldots$ be an enumeration of all nondeterministic oracle Turing machines. Function α will be as in theorem IV.3.2, and, as there, the oracle Y will be constructed in stages. Initially, $Y_1 = \emptyset$. At stage $n > 1$ simulate every path of nondeterministic machine $N_{\alpha(n)}$ for at most $2^{\wp(n)-1}$ steps on input $1^{\wp(n)}$ using Y_n as oracle.

If the machine accepts the input within $2^{\wp(n)-1}$ steps, choose the shortest accepting path and let Q_n be the set of queries on that path. Again, $|Q_n| \leq 2^{\wp(n)-1}$, and R_n is a subset of $\Sigma^{\wp(n)}$ of cardinality $2^{\wp(n)-1}$ such that $R_n \cap Q_n = \emptyset$. If the machine rejects the input or does not halt in the allowed number of steps, let $R_n = \emptyset$. In both cases, $Y_{n+1} = Y_n \cup R_n$ and finally $Y = \bigcup_{n \geq 1} Y_n$. Again, it is clear that Y enjoys the property B. We define, as before,

$$S_Y = \{1^n \mid Y \cap \Sigma^n = \emptyset\}.$$

As in the proof of theorem IV.3.2, there is a quantum Turing machine with oracle Y that recognizes S_Y in linear time.

Consider now a nondeterministic Turing machine N_i^Y that recognizes S_Y and any $n \in \mathbb{N}$ such that $\alpha(n) = i$. We claim that N_i^Y accepts input $1^{\wp(n)}$ and uses at least $2^{\wp(n)-1}$ steps to do so. Suppose, for the sake of contradiction, that N_i^Y rejects input $1^{\wp(n)}$. Then it follows that $1^{\wp(n)} \notin S_Y$, since N_i^Y was supposed to recognize S_Y. Therefore, by the definition of S_Y, $Y \cap \Sigma^{\wp(n)} \neq \emptyset$. But this means, by the definition of oracle Y, that N_i accepts $1^{\wp(n)}$ within $2^{\wp(n)-1}$ steps using oracle Y_n. Consider now the shortest accepting path used in that stage. By the definition, the set R_n does not contain any of the oracle queries asked along this path. Moreover, none of those query strings can be longer than $2^{\wp(n)} - 1$ because there would not have been time to write such a query. Sets Y and Y_n can differ on strings of length smaller that $2^{\wp(n)}$ only in the strings in R_n that belong to Y but do not belong to Y_n, so the same accepting path of N_i on $1^{\wp(n)}$ exists when oracle Y is used instead of Y_n. This is enough for N_i^Y to accept $1^{\wp(n)}$, a contradiction.

Suppose then, for contradiction, that $N_i{}^Y$ will accept $1^{\wp(n)}$ within $2^{\wp(n)-1}$ steps. Then $1^{\wp(n)} \in S_Y$, since N_i^Y was supposed to accept S_Y. But this means that $Y \cap \Sigma^{\wp(n)} = \emptyset$, so Y and Y_n are identical when restricted to strings of length $2^{\wp(n)}$ or less. Therefore it does not matter if we use oracle Y or Y_n for the first $2^{\wp(n)} - 1$ steps of N_i. So, $N_i^{Y_n}$ also accepts $1^{\wp(n)}$ and therefore $Y \cap \Sigma^{\wp(n)} \neq \emptyset$ by the construction of Y, a contradiction. □

Corollary IV.3.6. *There is an oracle Y such that*

$$\mathbf{QP}^Y \notin \mathbf{NP}^Y \cup \mathbf{coNP}^Y.$$

Proof. Choosing Y as in theorem IV.3.5 we see that $S_Y \in \mathbf{QP}^Y$ but $S_Y \notin \mathbf{NP}^Y$. Moreover, it is easy to see that the set

$$S_Y \cup \{0^n \mid 1^n \notin S_Y\}$$

belongs neither to $\mathbf{NP}^Y$ nor $\mathbf{coNP}^Y$ but is in $\mathbf{QP}^Y$. □

Theorem IV.3.7. *There exists an oracle Y for which there is a language that can be recognized in linear time by a quantum Turing machine but any* **ZPP** *or* **RP** *or* **coRP** *Turing machine that recognizes it must take exponential time on all but finitely many inputs.*

Proof. This follows from corollary IV.3.6, since $\mathbf{ZPP}^Y \subseteq \mathbf{RP}^Y \subseteq \mathbf{NP}^Y$ and also $\mathbf{ZPP}^Y \subseteq \mathbf{coRP}^Y \subseteq \mathbf{coNP}^Y$. □

Theorem IV.3.8. *There exists an oracle Z for which there is a language that can be recognized in exponential time with a quantum Turing machine but any deterministic Turing machine recognizing it takes doubly exponential time on all but finitely many inputs.*

Proof. Let $\sigma : \Sigma^* \to \mathbb{N}$ defined by

$$\sigma(x_n \ldots x_0) = 2^{n+1} + x_n \cdot 2^n + \ldots + x_2 \cdot 2^2 + x_1 \cdot 2^1 + x_0 \cdot 2^0.$$

It is clear that σ is a bijection, so it has the inverse, say τ. Oracle Z that satisfies property B will again be constructed in stages. The language in the theorem will not be

$$S_Z = \{1^n \mid Z \cap \Sigma^n = \emptyset\},$$

as earlier, but

$$V_Z = \{x \in \Sigma^* \mid X \cap \Sigma^{\sigma(x)} = \emptyset\}.$$

In this proof, a machine is called *cancelled*, if we have made sure that it will not correctly recognize V_Z using Z as oracle. A machine that has not been cancelled yet is called *live*.

Let M_1, M_2, M_3, ... be again an enumeration of all deterministic oracle Turing machines. Initially all machines are live and $Z_3 = \emptyset$. At stage $z \geq 3$ we simulate machines M_1, M_2, ... M_n on all the inputs $\{\tau(m) \mid 1 \leq m \leq n\}$ using Z_n as oracle up to $\frac{2^{n-1}}{n^2}$ steps. Let Q_n be the set of oracle queries asked on the simulations, so Q_n will contain at most $n \cdot n \cdot \frac{2^{n-1}}{n^2} = 2^{n-1}$ queries.

If no one of the live machines stops in $\frac{2^{n-1}}{n^2}$ steps on input $\tau(n)$, set $Z_{n+1} = Z_n$ and continue at the next stage. Otherwise, let i be the smallest integer such that M_i is live and stops within $\frac{2^{n-1}}{n^2}$ steps on input $\tau(n)$ using oracle Z_n. If $M_i^{Z_n}$ rejects $\tau(n)$, set $R_n = \emptyset$. Otherwise, let R_n be any subset of Σ^n of size 2^{n-1} such that $Q_n \cap R_n = \emptyset$. In both cases, cancel machine i, set $Z_{n+1} = Z_n \cup R_n$ and continue at the next stage. Finally set $Z = \bigcup_{n \geq 3} Z_n$. It is clear that oracle Z now satisfies property B.

As in theorems IV.3.2 and IV.3.4, language S_Z can be recognized in linear time by a quantum Turing machine using Z as an oracle. Therefore, a quantum Turing machine can recognize set V_Z in exponential time using Z as an oracle (V_Z being merely an exponentially more compact representation of S_Z).

Suppose now that a deterministic Turing machine M_i^Z recognizes V_Z. Then M_i cannot be cancelled. To see this, suppose, for contradiction, that M_i was cancelled at stage n. But $M_i^{Z_n}$ rejects input $\tau(n)$ if and only if Z is empty at length n, which happens if and only if $\tau(n) \in V_Z$. Thus $M_i^{Z_n}$ does not recognize V_Z. But all queries asked by M_i on input $\tau(n)$ under Z_n will be in Q_m for all $m \geq n$ and so Z will not differ from Z_n on any of those queries, which means that M_i^Z cannot recognize V_Z, a contradiction (starting at stage 3 makes sure that $\frac{2^{m-1}}{2^m} \geq \frac{2^{n-1}}{n^2}$ for all $m \geq n \geq 3$).

Let n_0 be large enough that all machines of index smaller than i that will be cancelled have been cancelled at stage n_0 and let n be greater than maximum of i and n_0. Then machines M_j cannot be cancelled at stage n, when $j \leq i$. But this means that M_i using oracle Z_n must consume more than $\frac{2^{n-1}}{n^2}$ steps on input $\tau(n)$, and, as before, the argument applies also when Z is used as an oracle instead of Z_n. The length of $\tau(n)$ is logarithmic in n and therefore $\frac{2^{n-1}}{n^2}$ is doubly exponential in the length of $\tau(n)$. Since τ is a bijection, M_i^Z that recognizes Z_V must take at least double exponential time on all inputs large enough. □

The proofs of theorems IV.3.2, IV.3.4 and IV.3.7 are introduced in [5]. These relative results suggest that quantum computation may be indeed exponentially more powerful than the classical computation, but also "inverse" relative results are known (see [2]):

Theorem IV.3.8. *Relative to a random oracle R, with probability 1, $\mathbf{NP}^R$ contains a language that is not in $\mathbf{BQP}^R$.*

Theorem IV.3.9. *Relative to a random permutation oracle π, with probability 1, $(\mathbf{NP} \cap \mathbf{coNP})^\pi$ contains a language that is not in $\mathbf{BQP}^\pi$.*

V. Conclusions

V.1. Comparing quantum and classical computation

In chapter III polynomial time quantum algorithms for factoring integers and for logarithm in prime fields were introduced. These algorithms were probabilistic ones, which is typical in quantum computation, due to the probabilistic nature of quantum physics. However, the working probability can be estimated and consequently one can give an estimation of the expected time of getting the result. Actually, no computer in the real world works certainly as we wanted, but there is always a small possibility of malfuction. Indeed, we can think that the randomization takes place everywhere, as mentioned in the following quotation from [20, p. 241]:

In some very real sense, computation is inherently randomized. It can be argued that the probability that a computer will be destroyed by a meteorite during any given microsecond of its operation is at least 2^{-100}.

As we have convinced ourselves that the probabilistic nature of computation is not an actual disadvantage, we will ask if quantum computation is strictly more powerful than the classical one. Let us therefore compare the classical and quantum algorithms for factorization and discrete logarithm.

In quantum factoring and discrete logarithm algorithm the running time is proportional to

$$|n|^3 \log |n|,$$

and the reliability goes exponentially to one as the constant coefficient grows (number of repeats is increased). The asymptotic running times of the fastest classical algorithms [24] are listed below:

The quadratic sieve:	$\mathcal{O}\Big(\exp\big((1+\mathrm{o}(1))\sqrt{\ln n \,\ln\, \ln n}\big)\Big)$
Elliptic curve method:	$\mathcal{O}\Big(\exp\big((1+\mathrm{o}(1))\sqrt{2\ln p \,\ln\, \ln p}\big)\Big)$
Number field sieve:	$\mathcal{O}\Big(\exp\big((1.92+\mathrm{o}(1))(\ln n)^{1/3}(\ln\, \ln n)^{2/3}\big)\Big)$

Here p stands for the smallest prime factor of n. In general, n can have large prime factors (as in the implementation of RSA) and these algorithms remain subexponential (remember that the size of the input, $|n|$, is here proportional to $\ln n$).

The fastest classical algorithms for discrete logarithm ([16] and [24]) are seen here:

Shanks' Algorithm:	$\mathcal{O}(\sqrt{p}\ln p)$
Index Calculus Method:	$\mathcal{O}\Big(\exp\big((1+\mathrm{o}(1))\sqrt{\ln p \,\ln\, \ln p}\big)\Big)$
Pohlig-Hellman:	Time depends on the factorization of $p-1$.

Also for the discrete logarithm the best known algorithm in the general case is subexponential.

The theoretical time complexity of known algorithms for both tasks are better in quantum computation. However one should keep in mind that the intractability of factoring and discrete logarithm is only a hypothesis (although well a justified hypothesis), so we cannot say with certainty that quantum automata are essentially faster than classical ones.

In chapter IV we tried to compare polynomial time exact quantum computation with nondeterministical polynomial time classical computation by showing that $\mathbf{QP}^Y$ is not included in $\mathbf{NP}^Y$ nor $\mathbf{coNP}^Y$ for an oracle Y. This implies that $\mathbf{QP}^Y$ is strictly greater than $\mathbf{P}^Y$, but the result is obtained with respect to oracle Y. The answer to the question $\mathbf{P}\subsetneq\mathbf{QP}$ is not known. Also, it is not known if $\mathbf{BPP}\subsetneq\mathbf{BQP}$. The hardness of this problem is evident: It is not difficult to show that $\mathbf{BPP} \subseteq \mathbf{PSPACE}$, so proving that $\mathbf{BPP} \neq \mathbf{BQP}$ would also solve the longstanding open question $\mathbf{BPP} \neq \mathbf{PSPACE}$.

As a conclusion we can say that even though no direct and absolute proof has been given to show that quantum computation is stricly stronger than classical one, we have many reasons to believe this.

V.2 On possible implementations of a quantum computer

So far we know, factoring is classically intractable, but with certainty we know that factoring is a tractable problem in the quantum computation model studied in this work. Therefore it is natural to ask if quantum computers could really be constructed. Again we have a question that presently can be answered only partially.

In fact, any two-state system small enough to allow a quantum superposition to appear could serve as a qubit. The crucial point is how to implement gate operations. Also, one problem is how we can isolate a quantum system apart from its environment well enough so that the surroundings would not disturb the superposition. The destructive effect of the environment onto the quantum superposition is here called *decoherence.*

In [8] a model based on cold ions confined in a linear trap as qubits was proposed. In this model the qubits would interact with laser beams. The decoherence would be neglible and the readout could be carried out with a high efficiency [8]. However, the weaknesses of this implementation were studied in [21]. In [19] weakly-coupled quantum systems that could be physically feasible and even realizable with the current technology were analyzed.

In [25] P. Törmä presented a realization of quantum computer using polarized photons as qubits. Also the effect of decoherence on performing the quantum Fourier transform was studied, and in the presence of decoherence, an approximate version of the quantum Fourier transform was shown to be more accurate than the exact one. Using this model we will represent in the next section an interesting extension of finite quantum automaton that would be able to solve **NP**-complete problems in polynomial time using exponential amoung of energy, but whether it will become possible to realize a large-scale quantum computer, remains to be seen.

V.3 On an optical quantum computer

In chapter III we described a quantum register as a collection of two-state quantum systems that are mathematically described as two-dimensional Hilbert spaces. In [25] P. Törmä has studied the possible realizations using polarized photons as qubits. This opens us an interesting scheme to solve **NP**-complete problems with a quantum computer in polynomial time and space, however using an exponential amount of energy (a **NP**-complete problem is a problem $\mathcal{P} \in \mathbf{NP}$ such that each problem in **NP** can be reduced to $\mathcal{P}$ in polynomial time). This idea was introduced in [7].

Let us consider a quantum automaton designed to solve SATISFIABILITY. In SATISFIABILITY a propositional formula on n Boolean variables, say $\psi(x_1, \ldots, x_n)$ is given as the input. We can assume that ψ is given in conjunctive normal form. Algorithm should decide whether ψ is satisfiable or not. It is well known that SATISFIABILITY is **NP**-complete [20, p. 171]. Our quantum automaton will be formed of memory long enough to sustain ψ encoded in some way plus $n+1$ additional cells. The initial configuration is given by

$$\underbrace{|0\rangle\otimes|0\rangle\otimes\ldots\otimes|0\rangle}_{n+1\text{ cells}}\otimes|\psi\rangle\otimes\underbrace{|0\rangle\otimes\ldots\otimes|0\rangle}_{\text{workspace}}. \tag{5-1}$$

The n first steps of the computation consist of applying A onto cells 2, 3, $\ldots$, $n-1$ thus yielding

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|0\rangle\otimes|x\rangle\otimes|0\rangle. \tag{5-2}$$

Each value of $x\in\{0,1,\ldots,2^n-1\}$ now stands for a possible valuation of ψ. Given any valuation, ψ can be evaluated in polynomial time, so there exists a unitary transformation U decomposable into polynomially many unitary mappings acting only on 2-dimensional spaces and evaluating ψ:

$$U\,|0\rangle\otimes|x\rangle\otimes|\psi\rangle\otimes|0\rangle=|\psi(x)\rangle\otimes|x\rangle\otimes|\psi\rangle\otimes|0\rangle. \tag{5-3}$$

Here we assumed that U already contains the inverse computation, so the workspace will be empty when the action of U is finished. Ignoring the workspace and ψ (5-2) then becomes

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|\psi(x)\rangle\otimes|x\rangle. \tag{5-4}$$

The problem is now this: If we have only exactly one satisfying valuation, the probability to observe 1 in the first bit is $\frac{1}{2^n}$ and the probability of observing 0 is $1-\frac{1}{2^n}$. This is not far away from the case that ψ is not satisfiable. Indeed, we should perform exponentially many computations and measurements to distinguish generally these two cases. In what follows, the idea is that in an optical realization of a quantum computer at least in principle, *all these computations can run at the same time.*

The idea is to make use of two photon polarization states $|0\rangle$ and $|1\rangle$. For each qubit, there would be a light source sending a photon polarized in a way that correspond to the initial configuration. The circuit will be built of physical components such as beam splitter (see [25] for more details). Finally, there should be a detector that informs us if state $|1\rangle$ is observed. Photons belong to integer-spin particles called bosons (these concepts will be explained in any basic textbook on quantum physics), and it is possible, at least in principle, to run *exponentially many coherent photons* via the machine simultaneously. This will require exponentially powerful laser light, but we can tell in polynomial time with bounded probability of error if there exists a satisfiable truth assignment or not.

Suppose that the input proposition is satisfiable. The probability of seeing this can be forced arbitrarily close to one provided the light source is exponentially powerful. For instance, an estimation $\ln(1-x)\leq -x$ for $x\in$

$(0,1)$ can be used to show that $(1-\frac{1}{2^n})^k \leq \mathrm{e}^{-\frac{k}{2^n}}$, and if the source is powerful enough to send $4 \cdot 2^n$ coherent photons at the same time, an easy calculation shows that the probability that an existing satisfiable truth assignment is not observed, is less than 0.02. If however the input proposition is unsatisfiable, an incorrect detection cannot occur.

However, this model cannot be suited into **BPP** and therefore is no help in showing that **NP** $\subseteq$ **BPP** (which is not certain at all). Indeed, in [7] it was conjectured that solving a **NP**-complete problem will inherently require exponentially some resource (e.g. time, machinery, energy).

Appendix

VI. QUANTUM PHYSICAL BACKGROUND

VI.1. Quantum physical phenomena

As soon as one becomes familiar with the model of classical Turing machine, there is no doubt that a Turing machine is a "good" model of computation. Evidently this is due to the finite and deterministic description of a Turing machine. Moreover, one immediately can say that a nondeterministic model is not realizable device but a mathematical tool to classify algorithmic problems, for instance. The main theme of this chapter is to give the reader some basic knowledge of present quantum physics to be convinced that the model of the quantum Turing machine could be some day a realistic model of computation.

In the classical non-relative physics the particles can be described by vectors in the three-dimensional vector space $\mathbb{R}^3$. Such a vector is interpreted to be the locations of single particles in three-dimensional space at the time t. The location is fully determined at a certain time, and the future behaviour of particles can be calculated in this model at any time, assuming that the current situation and the interaction of particles are known precisely. This deterministic wiew of the universe remained as the main flow in the sciences to the end of the 19th century.

However, in the beginning of 20th century phenomena that can not be explained in accordance to this model were observed. For example, the radiation of a black body was only partially explained by two contradictory classical theories. Other examples are the discoveries of particle-like behaviour of light and wave-like behaviour of particles as well. Further observations gave birth to an entirely new model, quantum physics. In this chapter a brief introduction to non-relativistic quantum mechanics will be given.

The discovery of non-classical phenomena began to tremble the old ideas of a fully deterministic universe acting as a "clockwork". It had turned out by the experiment, that one can not predict the behaviour of a single particle small enough in all circumstances, but the "distribution of behaviours" seemed to stabilize, if a large number of experiments were rendered in equal conditions. This gives rise, naturally enough, to a deterministically probabilistic model. Indeed, according to present quantum physics, one can determine the probability distribution of possible values of a dynamical variable, for instance the location and momentum of a single particle.

VI.2. Mathematical tools

The theory of linear spaces plays a great role in the mathematical formulation of quantum physics. Important mathematical tools in the field of quantum mechanics are linear operators in linear spaces. Some of those mathematics needed will now be introduced.

A complex *Hilbert space* $\mathcal{H}$ is a complete vector space over the field of complex numbers $\mathbb{C}$ equipped with an inner product $\langle \cdot \mid \cdot \rangle \to \mathbb{C}$. It is required that the inner product satisfies the following axioms:

1) $\langle \psi \mid \varphi \rangle = \overline{\langle \varphi \mid \psi \rangle}$

2) $\langle \psi \mid \psi \rangle \geq 0$ and $\langle \psi \mid \psi \rangle = 0$ if and only if $\psi = 0$.

3) $\langle \psi \mid c_1\varphi + c_2\eta \rangle = c_1\langle \psi \mid \varphi \rangle + c_2\langle \psi \mid \eta \rangle$

Usually it is required that the inner product should be linear with respect to the first argument, but in the literature of physics convention (3) is adopted more widely.

The inner product induces a norm in $\mathcal{H}$: $||\psi|| = \sqrt{\langle \psi \mid \psi \rangle}$. Furthermore, the norm induces a metric $d(\psi, \varphi) = ||\psi - \varphi||$, and the metric induces a topology. The completeness of the Hilbert space means that each Cauchy sequence (determined by the induced metric) in $\mathcal{H}$ converges. Any topological feature mentioned below is assumed to be determined by this topology unless stated otherwise. An important property of Hilbert spaces is the decomposition into orthogonal subspaces, more precisely:

Theorem VI.2.1. *For each closed subspace W of $\mathcal{H}$ there exists dual space $W^\perp$, such that, for any $\varphi \in W$ and $\psi \in W^\perp$, $\langle \varphi \mid \psi \rangle = 0$ and*

$$\mathcal{H} = W \oplus W^\perp.$$

Proof. See [18], for instance. □

Another important feature of Hilbert spaces is *self duality*, which is characterized in the following theorem. Linear functions from $\mathcal{H}$ to $\mathbb{C}$ will be called *functionals*.

Theorem VI.2.2. *(Fréchet-Riesz) For each continuous functional $f : \mathcal{H} \to \mathbb{C}$ there exists unique vector $\varphi_f \in \mathcal{H}$ that satisfies*

$$f(\psi) = \langle \varphi_f \mid \psi \rangle.$$

Proof. See [18]. $\square$

It must be emphasized that the functionals also form a linear space over $\mathbb{C}$, when the addition and scalar multiplication is defined in the most obvious way. This *dual space* will be called $\mathcal{H}^*$. The Fréchet-Riesz theorem establishes an *antilinear mapping* from $\mathcal{H}$ to the space of continous funtionals $\mathcal{H}^*$. The correspondence will be given by $\omega : \mathcal{H}^* \to \mathcal{H}$, $\omega(f) = \varphi_f$, where φ_f is the vector given by theorem VI.2.2. *Antilinearity* here means that

$$\omega(c_1\psi_1 + c_2\psi_2) = \overline{c}_1\omega(\psi_1) + \overline{c}_2\omega(\psi_2).$$

Elements of $\mathcal{H}$ and $\mathcal{H}^*$ are usually identified by the correspondence given by ω, and if $f \in \mathcal{H}^*$, $\psi \in \mathcal{H}$, then $\langle f \mid \psi \rangle$ will be defined as

$$\langle f \mid \psi \rangle = \langle \varphi_f \mid \psi \rangle,$$

where φ_f is the unique vector in $\mathcal{H}$ given by theorem VI.2.2.

The following notation is due to P.A.M. Dirac (1902-1984): Vector φ in the Hilbert space is denoted by $\mid \varphi \rangle$ and referred to as a *ket-vector* and the corresponding functional given by theorem VI.2.2 is denoted by $\langle \varphi \mid$ and referred as a *bra-vector*. The above bracket notation for inner product appears then very natural.

In addition to functionals, there is another important class of mappings, namely linear mappings from $\mathcal{H}$ to itself. These mappings are called linear *operators*.

VI.3. Linear operators

When studying the theory of finite-dimensional vector spaces one knows that all linear operators are continuous mappings and that they can be represented by $n \times n$-matrices, where n is the dimension of the vector space. This forms a solid background for the reader to understand the nature of linear mappings, but in quantum physics, also infinite-dimensional Hilbert spaces are very important. In infinite-dimensional spaces some non-familiar phenomena can occur, namely the non-continuity of a linear mapping. Moreover, it may be convenient to investigate linear mappings that are not defined in whole space $\mathcal{H}$, but only in a subspace of $\mathcal{H}$.

Definition VI.3.1. By an *operator* in $\mathcal{H}$ one understands a linear mapping (not necessarily continuous) $T : \mathrm{Dom}(T) \to \mathcal{H}$, whose domain $\mathrm{Dom}(T)$ is a subspace of $\mathcal{H}$. We write $\mathcal{L}(\mathcal{H})$ to denote all linear operators in $\mathcal{H}$.

Definition VI.3.2. Assume that T is an operator whose domain is dense in $\mathcal{H}$ (the closure of $\mathrm{Dom}(T)$ is $\mathcal{H}$). The *adjoint operator* of T is an operator T^*, whose domain $\mathrm{Dom}(T^*)$ consists exactly of those $\psi \in \mathcal{H}$, for which the linear mapping

$$\varphi \mapsto \langle \psi \mid T\varphi \rangle$$

is continuous in $\mathrm{Dom}(T)$, and for all $\psi \in \mathrm{Dom}(T^*)$ the value $T^*\psi$ is defined by

$$\langle \psi \mid T\varphi \rangle = \langle T^*\psi \mid \varphi \rangle$$

for all $\varphi \in \mathrm{Dom}(T)$. To ensure that the value $T^*\psi$ is well-defined, see [18].

Note that if T is continuous and defined in the whole space, then the adjoint operator is also continuous and defined everywhere.

Definition VI.3.3. Let us assume that T is an operator with dense domain. If $T = T^*$, then T is said to be *self-adjoint*.

Self-adjoint operators are of special importance in quantum physics. Let $\mathcal{L}_S(\mathcal{H})$ be the set of all self-adjoint operators in $\mathcal{H}$.

Definition VI.3.4. A linear operator T is *positive* if $\langle \varphi \mid T\varphi \rangle \in \mathbb{R}_+ \cup \{0\}$ for all vectors φ in $\mathcal{H}$. In this case we denote $T \geq 0$.

It can easily be shown (see [18]) that a positive operator is self-adjoint. We will denote the set of all positive operators by $\mathcal{L}_S(\mathcal{H})^+$. The relation $T \geq S$ defined in the natural way induces a partial order in $\mathcal{L}_S(\mathcal{H})^+$. Notations $T \leq S$, $T > S$ and $T < S$ are used in the natural way.

The decomposition into orthogonal subspaces offers us a way to define a special type of operators, namely the *projections*. Assume that $\mathcal{H} = W \oplus W^\perp$ is a decomposition of $\mathcal{H}$ into orthogonal subspaces. Then each $\varphi \in \mathcal{H}$ possesses a unique representation $\varphi = \varphi_W + \varphi_{W^\perp}$, where $\varphi_W \in W$ and $\varphi_{W^\perp} \in W^\perp$ are orthogonal. We can then define

$$P_W(\varphi) = \varphi_W.$$

It is an easy task to show that P_W becomes an operator. P_W is called the *projection onto subspace* W. Projections can be easily characterized:

Theorem VI.3.5. *An operator P is a projection onto some subspace W if and only if $P = P^2$ and $P = P^*$.*

Proof. See [18]. □

It is easily seen that if $||\varphi|| = 1$, mapping $|\varphi\rangle\langle\varphi|$ defined by

$$|\varphi\rangle\langle\varphi||\eta\rangle = \langle \varphi \mid \eta \rangle \, |\varphi\rangle$$

is a projection onto the one-dimensional linear subspace spanned by $|\varphi\rangle$. Note that here also Dirac's notation seems to be very natural. We also denote $|\varphi\rangle\langle\varphi| = P[\varphi]$.

Another important class of linear operators are defined by the means of *trace*. The trace of an operator A is defined to be

$$\mathrm{Tr}(A) = \sum_{\xi \in K} \langle \xi \mid A\xi \rangle,$$

where K is an orthonormal basis of $\mathcal{H}$. It can be shown (see [18]) that the trace is independent of the choice of K.

Definition VI.3.6. The *trace class* $\mathcal{T}(\mathcal{H})$ consists of linear operators with finite trace.

Note that in the finite-dimensional case the trace of a linear mapping A equals the sum of diagonal elements of a matrix representing A. Moreover, each element in the trace class can be represented (see [6] or [18]) in form

$$A = \sum_n c_n \mid \varphi_n \rangle \langle \varphi_n \mid,$$

where (φ_n) is an orthogonal sequence of projections in the operator space and $c_n \in [0, \infty)$. This is an example of *spectral representation.* Spectral theory is far off from the range of this thesis, but a basic outline will briefly be represented. For more accurate representation, see [18] for instance.

VI.4. Spectral theory briefly

Definition VI.4.1. Let Ω be an arbitrary set. A collection $\mathcal{A}$ of subsets of Ω is called a *σ-algebra* if $\emptyset \in \mathcal{A}$ and $\mathcal{A}$ is closed under complementation and countable union. If (Ω, τ) is a topological space, the least σ-algebra containing τ is said to be the *Borel* σ-algebra of (Ω, τ) and denoted by $\mathcal{B}(\Omega, \tau)$, or by $\mathcal{B}(\Omega)$ if there is no danger of confusion.

Definition VI.4.2. Let $\mathcal{A} \subseteq 2^\Omega$ be a σ-algebra. A mapping $\mu : \mathcal{A} \to \mathbb{C}$ is a *complex measure*, if $\mu(\emptyset) = 0$ and

$$\mu(\bigcup_{n=1}^{\infty} A_n) = \sum_{n=1}^{\infty} \mu(A_n)$$

whenever (A_n) is a sequence of disjoint sets in $\mathcal{A}$.

Definition VI.4.3. Let $\mathcal{A}$ be as above and let $\mathcal{P}(\mathcal{H})$ be the set of projections of the Hilbert space $\mathcal{H}$. Mapping $E : \mathcal{A} \to \mathcal{P}(\mathcal{H})$ is a *spectral measure*, if $E(\Omega) = I$ is the identity mapping and the induced mapping $E_{\varphi,\psi} : \mathcal{A} \to \mathbb{C}$ defined by

$$E_{\varphi,\psi}(X) = \langle \psi \mid E(X)\varphi \rangle.$$

is a complex measure for all vectors $\varphi, \psi \in \mathcal{H}$.

Definition VI.4.4. A function $F : \mathbb{R} \to \mathcal{P}(\mathcal{H})$ is called *a spectral function*, if $F(s) \leq F(t) \leq I$ for all $s \leq t$, F is continuous from left and F satisfies both

$$\lim_{t \to -\infty} F(t) = 0 \text{ and } \lim_{t \to \infty} F(t) = I.$$

Note that there is a one-to-one correspondence (see [18]) between spectral functions and spectral measures on $\mathcal{B}(\mathbb{R})$ given by $F(t) = E((-\infty, t])$ and $E((a, b]) = F(b) - F(a)$. Here $\mathbb{R}$ is equipped with the natural topology.

Let A_j be a collection of elements of $\mathcal{A}$ and χ_{A_j} the characteristic function of A_j ($\chi_{A_j}(\omega) = 1$ if $\omega \in A_j$ and 0 otherwise). A function $f : \mathcal{A} \to \mathbb{C}$ of form

$$f = \sum_{j=1}^{n} a_j \chi_{A_j}, \quad a_j \in \mathbb{C}$$

is said to be *simple*. If $E : \mathcal{A} \to \mathcal{P}(\mathcal{H})$ is a spectral measure, then the *integral of a simple function with respect to spectral measure* E is defined to be

$$\int f \, dE = \sum_{j=1}^{n} a_j E(A_j).$$

It is easy to show that the class of simple functions is dense in the set of all bounded $\mathcal{A}$-measurable functions, $\mathcal{F}_{\mathcal{A}}$. Thus the integral with respect to spectral measure can be uniquely defined in whole $\mathcal{F}_{\mathcal{A}}$. Integral of a general $\mathcal{A}$-measurable function can naturally be defined as the limit of a sequence of integrals of simple functions. It can be shown that the result is independent of the choice of the sequence chosen. For details, see [18].

One of the most important results in the spectral theory is

Theorem VI.4.5 (The Spectral Theorem). *Let A be a self-adjoint operator in $\mathcal{H}$. There exists unique spectral measure $E : \mathcal{B}(\mathbb{R}) \to \mathcal{P}(\mathcal{H})$ satisfying*

$$A = \int_{\mathbb{R}} \lambda \, dE$$

The spectral measure E in the theorem will in continuation be referred to as the *spectral measure associated to* A and denoted by E^A. It can be show, that instead of $\mathbb{R}$, there is the least subset of $\mathbb{R}$, $\sigma(A)$, such that integration over $\sigma(A)$ with respect to E^A yields the same result. The set $\sigma(A)$ is called *spectrum* of operator A. Let $a \in \sigma(A)$. If, for all positive ε, the neighbourhood $(a - \varepsilon, a + \varepsilon)$ is *not* included in $\sigma(a)$, we say that a is in the *discrete spectrum* of A. The discrete spcetrum of A is denoted by $\sigma_d(A)$ and the complement $\sigma_c(A) = \sigma(A) \setminus \sigma_d(A)$ is called the *continuous spectrum*. It is easy to see ([6] or [18]) that the discrete spectrum contains all eigenvalues of A.

Let us now suppose that $\mathcal{H}$ is of finite dimension n and A is a self-adjoint operator defined in the whole space. Then A is bounded and its spectrum consists exactly of the eigenvalues, say λ_1, λ_2, ..., λ_n (not necessarily distinct). Let φ_i be an eigenvector belonging to λ_i. It is well known that the set $\{\varphi_1, \varphi_2, \dots, \varphi_n\}$ can be chosen to be an orthonormal set. Operator A can then be represented as

$$A = \lambda_1 \mid \varphi_1\rangle\langle\varphi_1 \mid + \lambda_2 \mid \varphi_2\rangle\langle\varphi_2 \mid + \ldots + \lambda_n \mid \varphi_n\rangle\langle\varphi_n \mid .$$

Note that the projections $\mid \varphi_i\rangle\langle\varphi_i \mid$ are unique (this follows from the the spectral theorem, see [18]), but the *vectors* φ_i are not. The spectral measure associated to A is now determined by the spectral function

$$F(t) = \sum_{i=1}^{n} \eta(t - \lambda_i) \mid \varphi_i\rangle\langle\varphi_i \mid$$

by setting $E^A((a,b]) = F(b) - F(a)$. Here η is Heaviside's unit function defined by $\eta(t) = 0$ for $t < 0$ and $\eta(t) = 1$ for $t \geq 0$.

VI.5. States

Intuitively, by a *state* one understands a situation in a physical world. One can *prepare* a physical object, for example a particle to be in a certain state by using specific instrumentation. Mathematically these states are represented by normed elements in the trace class.

Definition VI.5.1. The *State set* associated to a physical system S is defined to be

$$\mathcal{T}_S(\mathcal{H})_1^+ = \{A \in \mathcal{T}(\mathcal{H}) \mid A \geq 0, \mathrm{Tr}(A) = 1\}.$$

The elements of $\mathcal{T}_S(\mathcal{H})_1^+$ are called *states*. Set $\mathcal{T}_S(\mathcal{H})_1^+$ is easily seen to be *convex*, i.e. $tT_1 + (1-t)T_2 \in \mathcal{T}_S(\mathcal{H})_1^+$ for all $T_1, T_2 \in \mathcal{T}_S(\mathcal{H})_1^+$ and $t \in [0,1]$. A state is said to be *pure*, if it is extremal in $\mathcal{T}_S(\mathcal{H})_1^+$, i.e. if $T = tT_1 + (1-t)T_2$ then $t = 0$ or $t = 1$. A state is *mixed* if it is not pure. It is easy to see [18] that a state T is pure if and only if $T = \mid \varphi\rangle\langle\varphi \mid$ for some vector φ of unit length. A pure state $\mid\varphi\rangle\langle\varphi\mid$ is often identified with its *radius* $\{\mathrm{e}^{i\theta}\varphi \mid \theta \in \mathbb{R}\}$ or even with the so-called *vector state* $\varphi \in \mathcal{H}$.

Let $T_1 = \mid \varphi_1\rangle\langle\varphi_1 \mid$ and $T_2 = \mid \varphi_2\rangle\langle\varphi_2 \mid$ be pure states. Choosing α_1 and α_2 in such a way that vector $\varphi = \alpha_1\varphi_1 + \alpha_2\varphi_2$ is of unit length φ also determines a pure state. This state is called the *superposition* of T_1 and T_2. One can verify that

$$\mid\varphi\rangle\langle\varphi\mid = |\alpha_1|^2 \mid \varphi_1\rangle\langle\varphi_1 \mid + |\alpha_2|^2 \mid \varphi_2\rangle\langle\varphi_2 \mid + \alpha_1\overline{\alpha_2} \mid \varphi_1\rangle\langle\varphi_2 \mid + \alpha_2\overline{\alpha_1} \mid \varphi_2\rangle\langle\varphi_1 \mid .$$

The latter two terms in the sum are called *interference terms*. The Hilbert space formalism of quantum physics contains two main methods to obtain new states from given ones: mixing and superposition. In quantum physics, there are however so-called superselection rules that forbid some types of superpositions. Some of these rules are obtained theoretically, and some of them are experimental. Mixing states occur often if there is some *ignorance* when preparing the state, and interference phenomena can often be explained by means of superposition.

VI.6. Observables and the probability axiom

Examples of physical *observables* can be given: Position, velocity, energy, momentum. etc. In classical physics the main interest is concentrated to *canonical observables*, position and momentum. The importance of these observables is that other classical observables can be represented in terms of these two.

In the mathematical formalization of quantum physics, any physical system S is associated to a Hilbert space $\mathcal{H}_S$, and an observable corresponds to a linear operator $A : \mathcal{H}_S \to \mathcal{H}_S$, where $\mathcal{H}$ is the vector space of the state vectors.

Definition VI.6.1. An *observable* is a self-adjoint operator

$$A : \mathrm{Dom}(A) \to \mathcal{H}.$$

Any pair of an observable A and a state T defines a measure

$$E_T^A : \mathcal{B}(\mathbb{R}) \to [0,1],\ E_T^A(X) = \mathrm{Tr}(E^A(X)T),$$

where $E^A(X)$ is the spectral projection of operator A at set X.

Axiom 1. *Measure E_T^A gives the probability distribution of values of the observable A in state T.*

Axiom 1 means that the *set of all possible values of an observable A is exactly* $\sigma(A)$, the spectrum of A.

Let us now give an example of a finite-dimensional case. Consider a Hilbert space $\mathcal{H}$ of dimension n. Let A be an operator with disjoint eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ (not necessarily distinct). Let also $\varphi_1, \varphi_2, \ldots, \varphi_n$ be an orthonormal set of eigenvectors of A. Then A has unique spectral representation

$$A = \lambda_1 \mid \varphi_1\rangle\langle\varphi_1 \mid + \ldots + \lambda_n \mid \varphi_n\rangle\langle\varphi_n \mid .$$

For all $X \subseteq \mathbb{R}$ it is easy to see that

$$E_T^A(X) = E_T^A\left(X \cap \sigma_d(A))\right) = \sum_{a \in X \cap \sigma_d(A)} E_T^A(\{a\}).$$

The spectral measure associated to A gives now

$$E^A(\{\lambda_i\}) = \mid \varphi_{i_1}\rangle\langle\varphi_{i_1} \mid + \ldots + \mid \varphi_{i_k}\rangle\langle\varphi_{i_k} \mid,$$

where $\varphi_{i_1}, \dots \varphi_{i_k}$ are orthonormal eigenvectors of λ_i. For each pure state $T = |\eta\rangle\langle\eta|$ we have

$$\begin{aligned}
E_T^A(\{\lambda_i\}) &= \mathrm{Tr}(E^A(\{\lambda_i\})T) \\
&= \sum_{l=1}^{n} \langle \varphi_l \mid E^A(\{\lambda_k\})(|\eta\rangle\langle\eta|)\varphi_l\rangle \\
&= \sum_{j=1}^{k}\sum_{l=1}^{n} \langle \eta \mid \varphi_l\rangle\langle\varphi_l \mid (|\varphi_{i_j}\rangle\langle\varphi_{i_j}|)\eta\rangle \\
&= \sum_{j=1}^{k}\sum_{l=1}^{n} \langle \eta \mid \varphi_l\rangle\langle\varphi_{i_j} \mid \eta\rangle\langle\varphi_l \mid \varphi_{i_j}\rangle \\
&= \sum_{j=1}^{k} \langle \eta \mid \varphi_{i_j}\rangle\langle\varphi_{i_j} \mid \eta\rangle = \sum_{j=1}^{k} \left|\langle\varphi_{i_j} \mid \eta\rangle\right|^2
\end{aligned}$$

In general, if we represent the state vector in basis $\{\varphi_1, \dots, \varphi_n\}$, say

$$\eta = c_1\varphi_1 + \ldots + c_n\varphi_n,$$

the requirement of unit norm gives

$$\begin{aligned}
1 = \mathrm{Tr}(|\eta\rangle\langle\eta|) &= \sum_{i=1}^{n} \langle\varphi_i \mid \langle\eta||\eta\rangle\varphi_i\rangle \\
&= \sum_{i=1}^{n} \langle\eta \mid \varphi_i\rangle\langle\varphi_i \mid \eta\rangle = \sum_{i=1}^{n} |\langle\varphi_i \mid \eta\rangle|^2 .
\end{aligned}$$

Furthermore, the coefficients are given by $c_i = \langle\varphi_i \mid \eta\rangle$. So

$$E_T^A(\{\lambda_i\}) = \sum_{j=1}^{k} \left|\langle\varphi_{i_j} \mid \eta\rangle\right|^2 = \sum_{j=1}^{k} \left|c_{i_j}\right|^2 . \tag{6-1}$$

According to axiom 1 the sum (6-1) can be interpreted as the probability that observation of A yields result λ_i if system is in state $T = |\langle\eta \mid \eta\rangle|$. Careful interpretation of the nature of measure E_T^A is much more complicated, but a closer investigation is not the purpose of this thesis. Therefore we adopt this simple interpretation.

VI.7. The dynamical evolution of a quantum system

A linear mapping U in $\mathcal{H}$ is *unitary*, if $U^*U = UU^* = I$. Unitary mappings are known to be length and angle -preserving.

The dynamical evolution of a quantum system could be derived by analyzing some general properties of quantum physics (see [18]), but for convenience we state it here as an axiom.

Axiom 2. *The evolution of a pure quantum state* $|\phi_0\rangle\langle\phi_0|$ *is given by*

$$\varphi(t) = U(t)\varphi_0,$$

where each $U(t)$ *is unitary and moreover* $U : (\mathbb{R}, +) \to \mathcal{L}(\mathcal{H})$ *is continuous homomorphism.*

Axiom 2 includes the fact that the time evolution never turns a pure state into a mixed one. It is required that $U(t_1+t_2) = U(t_1)U(t_2)$ and that

$$\lim_{t \to t_0} U(t)\varphi = U(t_0)\varphi \text{ for all } \varphi \in \mathcal{H}.$$

A mapping satisfying these conditions is called a *strongly continuous one-parametric* group in $\mathcal{L}(\mathcal{H})$. There is a theorem by M.H. Stone [18] concerning strongly continuous one-parametric groups:

Theorem VI.7.1. *Let* U *be a strongly continuous one-parametric group in* $\mathcal{L}(\mathcal{H})$*. There exists unique self-adjoint operator* H *such that*

$$U(t) = e^{-itH}$$

for all $t \in \mathbb{R}$*.*

The notion e^{-itH} is defined by

$$\mathrm{e}^{-itH} = \int_{\mathbb{R}} \mathrm{e}^{-it\lambda}\, dE^H(\lambda),$$

where E^H is the spectral measure associated to H. The unitary evolution thus leads to the *Schrödinger equation*

$$i\frac{\partial}{\partial t}\varphi(t) = H\varphi(t).$$

Operator H also defines an observable; this is the *total energy* of the quantum system. Operator H is called the *Hamiltonian* of the system.

VI.8. Compound systems

Axiom 3. *Consider physical systems* S_1 *and* S_2 *and the Hilbert spaces* $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *associated to these systems. We assume that one can always distinguish between the systems* S_1 *and* S_2*.*

1) *The Hilbert space associated to the compound system, say* $S_1 + S_2$ *is the tensor product* $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$*.*

2) *The evolution of the compound system is determined by unitary operators in* $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$*.*

3) *The observables of* $S_1 + S_2$ *are self-adjoint operators in* $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$*.*

4) The states of the compound system are the trace class operators in $\mathcal{H}_\infty \otimes \mathcal{H}_2$.

5) The state T of the compound system determines uniquely the states T_1 and T_2 of the subsystems by mappings $\Pi_i : \mathcal{T}(\mathcal{H}_1 \otimes \mathcal{H}_2) \to \mathcal{T}(\mathcal{H}_i)$ which are determined by requirement

$$Tr(\Pi_1(T)A) = Tr(T(A \otimes \mathrm{id})) \text{ and } Tr(\Pi_2(T)A) = Tr(T(\mathrm{id} \otimes A))$$

for any self-adjoint A.

If $\{\varphi_i \mid i \in \mathbb{N}\}$ and $\{\psi_i \mid i \in \mathbb{N}\}$ are orthonormal bases of $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, then it can be shown [6] that

$$\Pi_1(T) = \sum_{i,j,k} \langle \varphi_i \otimes \psi_k \mid T(\varphi_j \otimes \psi_k)\rangle \, |\varphi_i\rangle\langle\varphi_j| \, .$$

Analogious representation can be found for Π_2.

Inner product in $\mathcal{H}_1 \otimes \mathcal{H}_2$ is defined by $\langle \varphi_1 \otimes \varphi_2 \mid \psi_1 \otimes \psi_1\rangle = \langle \varphi_1 \mid \varphi_2\rangle\langle\psi_1 \mid \psi_2\rangle$ and extending this in the only possible way.

The last statement should be reconsidered in the case that the subsystems are *unidentifiable*, for instance a pair of two bosons. See [6] or [18] for discussion.

If systems S_1 and S_2 do not interact, then the unitary evolution of $\mathcal{H}$, say $U(t)$ can be written as $U(t) = U_1(t) \otimes U_2(t)$, where U_1 and U_2 are unitary evolution operators of systems S_1 and S_2 respectively. However, all unitary mappings in $\mathcal{H}_1 \otimes \mathcal{H}_2$ cannot be written as a tensor product of unitary mappings in $\mathcal{H}_1$ and $\mathcal{H}_2$. In this case we say that the system S_1 and S_2 are under interaction.

It is known that if $\{\zeta_i \mid i \in \mathbb{N}\}$ and $\{\xi_i \mid i \in \mathbb{N}\}$ are orthonormal bases of $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, then $\{\zeta_i \otimes \xi_j \mid i, j \in \mathbb{N}\}$ is an orthonormal basis of the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$. We assume now that the compound system $S_1 + S_2$ is in a pure state $|\psi\rangle\langle\psi|$ and identify the state as usually with vector ψ.

Definition VI.8.1. If the state vector of S can be written in for $\psi = \psi_1 \otimes \psi_2$ where $\psi_1 \in \mathcal{H}_1$ and $\psi_2 \in \mathcal{H}_2$, we say that S is in a *decomposable* state. If the system S is not in a decomposable state, we say that subsystems S_1 and S_2 are *entangled* or that system S is entangled.

Example VI.8.2. Consider a two-dimensional Hilbert space $\mathcal{H}$ with orthonormal basis $\{\varphi_1, \varphi_2\}$. We suppose then that we have two distinguishable two-state systems S_1 and S_2 with associated Hilbert space $\mathcal{H}$. The Hilbert space associated to the compound system $S_1 + S_2$ is then $\mathcal{H} \otimes \mathcal{H}$ with orthonormal basis

$$\{\varphi_1 \otimes \varphi_1, \varphi_1 \otimes \varphi_2, \varphi_2 \otimes \varphi_1, \varphi_2 \otimes \varphi_2\}$$

for instance. We assume first that systems are both in pure states $|\varphi_1\rangle\langle\varphi_1|$. So the state vector of the compound system is $\varphi_1 \otimes \varphi_1$. We suppose that S_1 and S_2 are in the interaction and the unitary evolution of $S_1 + S_2$ during some period is given by matrix

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

which is clearly unitary. After this transformation the state vector of the compound system is

$$\frac{1}{\sqrt{2}}(\varphi_1 \otimes \varphi_1 + \varphi_2 \otimes \varphi_2),$$

and we see easily that this vector cannot be represented as a tensor product of vectors in $\mathcal{H}$. Thus the compound systems S_1 and S_2 are entangled. Another interesting thing in compound systems is that even though the state in $\mathcal{H} \otimes \mathcal{H}$ is pure, the states of the subsystems are not. Using item 5) in axiom 3 we can compute that the states of S_1 and S_2 are both

$$\frac{1}{2}\,|\varphi_1\rangle\langle\varphi_2| + \frac{1}{2}\,|\varphi_1\rangle\langle\varphi_2|\,.$$

VI.9. Measurement

Let S be a physical system, Hilbert space $\mathcal{H}_S$ associated to that system and A an observable. The measurement is an event performed in order to receive information about the value of an observable A. Measurement is rendered via a specific *measuring apparatus*. This is another physical system associated to Hilbert space $\mathcal{H}_A$. System $\mathcal{H}_A$ has to have a counterpart for observable A, say B. The systems are initially supposed to be disjoint and non-interactive. While performing the measurement, the apparatus and S interact. During the interaction the development of the compound system $\mathcal{H}_S \otimes \mathcal{H}_I$ runs through unitary evolution and usually *causes changes in the state of the compound system* and so in its subsystems, too. It is supposed that each possible value of A has its counterpart among the possible values of B, and by observing the post-measurement value of B we get some information of the value of A in the initial state of S. In this thesis, the purpose of the measurement is mainly to determine the current state of the system. We will omit the careful investigation of the measurement process and later on even ignore the measurement apparatus and just think that there exists some physical procedure which gives us information about the state and also changes the state.

In general, there are great difficulties: The values of B should be somehow "easily" reachable, or we have to perform the measurement again on B

and so on. Moreover, by experiment, a definite value of the measurement is always given; macroscopic systems do not occur in any superposition of states. So what is left from the original superposition of the quantum system when the process stops? What is the state of the original system when the observer finally observes a specific value of A? The problem of objectification of the quantum measurement is a very difficult one and there has been many attempts to interpret the quantum measurement process, see [6], for instance. However, the investigation needed there is not in the range of this thesis and we will adopt here so-called *projection postulate* proposed by von Neumann (see [18] and the references within). According to Neumann's interpretation observing a specific value of an observable makes the state vector to "collapse" and renormalize. This is formalized below. Here we will consider only the case where the Hilbert space of state vectors is of finite dimension.

Let $\mathcal{H}$ be a Hilbert space of dimension n with orthonormal basis $\{\varphi_1, \dots, \varphi_n\}$. Suppose that a quantum physical system S is in a state determined by state vector $\eta = c_1\varphi_1 + \ldots + c_n\varphi_n$ and consider an observable

$$A = 1 \cdot |\varphi_1\rangle\langle\varphi_1| + 2 \cdot |\varphi_2\rangle\langle\varphi_2| + \ldots + n \cdot |\varphi_n\rangle\langle\varphi_n| \,.$$

Now A is non-degenerate, i.e. all eigenvalues are disjoint. Indeed, all eigenvalues of A are $1, 2, \dots, n$ (The choice of the coefficients does not matter very much as they are disjoint). By Axiom 1 they are all possible values of A, and the probability that the measurement yields result i in state $|\eta\rangle\langle\eta|$ is exactly $|c_i|^2$.

Axiom 4. *Suppose that the measurement of A in state $T = |\eta\rangle\langle\eta|$ yields result i. Then the post-measurement state of system S is*

$$E^A(i)TE^A(i) = |\varphi_i\rangle\langle\varphi_i| \,.$$

Axiom 4 states that the observation of specific value of A really makes the state vector "collapse" according to which value was observed. Note that this interpretation is very close to the epistemic ignorance interpretation; when we say that the state of the system is a superposition

$$c_1T_1 + c_2T_2 \ldots + c_mT_m, \; c_i \in [0,1], \; \sum_{i=1}^{m} c_i = 1$$

we know that the system is in state T_i *with weight* c_i.

We will finally consider the measurement of a subsystem of a compound system in the example VI.8.2. Suppose that this system is in pure state determined by vector

$$\boldsymbol{\psi} = \frac{1}{\sqrt{2}}(\varphi_1 \otimes \varphi_1 + \varphi_2 \otimes \varphi_2),$$

so the subsystems are entangled. The measurement of observable

$$1 \cdot \mathrm{P}[\varphi_1 \otimes \varphi_1] + 2 \cdot \mathrm{P}[\varphi_1 \otimes \varphi_2] + 3 \cdot \mathrm{P}[\varphi_2 \otimes \varphi_1] + 4 \cdot \mathrm{P}[\varphi_2 \otimes \varphi_2]$$

would essentially measure the state of the compound system. In state $\mathrm{P}[\boldsymbol{\psi}]$ the possible values would be 1 and 4, each chosen with probability $\frac{1}{2}$, and the post-measure state of the compound system is determined by vector $|\,\varphi_1 \otimes \varphi_1\rangle$ or $|\,\varphi_2 \otimes \varphi_2\rangle$ depending on whether 1 or 4 was observed.

Suppose now that we would like to perform measurement only on system S_1. This is done via observable

$$A = 1 \cdot \mathrm{P}[\varphi_1 \otimes \varphi_1] + 1 \cdot \mathrm{P}[\varphi_1 \otimes \varphi_2] + 2 \cdot \mathrm{P}[\varphi_2 \otimes \varphi_1] + 2 \cdot \mathrm{P}[\varphi_2 \otimes \varphi_2],$$

whose spectrum is $\{1, 2\}$. We have

$$E^A_{\mathrm{P}[\boldsymbol{\psi}]}(\{1\}) = |\langle \varphi_1 \otimes \varphi_1 \mid \boldsymbol{\psi}\rangle|^2 + |\langle \varphi_1 \otimes \varphi_2 \mid \boldsymbol{\psi}\rangle|^2 = \frac{1}{2}$$

and similary $E^A_{\mathrm{P}[\boldsymbol{\psi}]}(\{2\}) = \frac{1}{2}$. So possible values 1 and 2 will be obtained equiprobably. Suppose for instance, that 1 was observed. The post-measure state of the compound system is

$$E^A(\{1\}) P[\boldsymbol{\psi}] E^A(1) = \mathrm{P}[\phi_1 \otimes \phi_1].$$

In general, measuring observing A on subsystem S_1 of a compound system yields the same result than observing $A \times \mathrm{id}$ on entire system. The measurement affects the state of the entire system.

INDEX

accept, 6, 9, 65
adjoint operator, 80
algorithm, 14
algorithmic problem, 14
antilinear mapping, 79
amplitude, 21
bra-vector, 79
Bennett, 5, 35
BPP, 16, 65
BQP, 66
Chomsky hierarchy, 17
Church-Turing thesis, 10, 16
co$\mathcal{C}$, 66
compound systems, 86
computation, 7, 10, 12, 20, 42
computation tree, 10, 12, 20
computational basis, 2, 41
configuration, 7, 42
configuration space, 13
complexity
  class, 18
  function, 14
  space -, 16
  time -, 15
decide, 9
decomposable, 23, 42, 87
decompos. of unitary matrix, 32
deterministic automaton, 6
Deutsch, D., 4, 19, 41, 67
discrete
  Fourier transform, 46
  logarithm, 58
  spectrum, 82
Dirac, P.A.M., 79
dual space, 78
entangled, 23, 42, 87
EQP, 66
evolution, 20, 24, 85
exponential time, 16
factoring integers, 51
Feynman, R., 1, 18
finite automata, 6
  deteministic, 6
  nondeterministic, 10
  probabilistic, 12
  quantum, 41
finite model, 30
Fréchet-Riesz, 79
formal language, 17
Fourier transform, 46
Fourier-Hadamard transform, 48
  Sampling Problem, 48
functional, 78
graph, 6
halt, 8, 11, 13
halting vertex, 11
Hilbert space, 78
inner product, 78
input word, 7
interference, 23, 83
intractable, 16
ket-vector, 79
level, 11
measure
  complex, 81
  spectral, 81
measurement, 88
memory cell, 7
mixed state, 83
Monte Carlo, 65
NP, 16
nondeterministic automaton, 10
observable, 84
observation, 21, 22
operator, 79
oracle, 69
P, 16
polynomial time, 16
positive operator, 80
probabilistic automaton, 12
problem of Deutsch and Josza, 67
probability
  distribution, 12
  of vertex, 13
  of reaching configuration, 14, 21
projection, 80

projection postulate, 89
$\mathbf{Q}\mathcal{C}$, 66
quantum
  automata, 41
  bit, 40
  computer, 71, 72
  gate, 40
  network, 41
  parallelism, 45, 46
  register, 40
  Turing machine, 19
  state, 83
radius, 83
recognize, 9
recursive
  function, 9
  language, 9
recursively enumerable, 9
reversible computation, 33
RP, 65
RSA, 5, 52
σ-algebra, 81
Shcrödinger
  equation, 86
  picture, 3
Shor, P., 5, 18, 41, 60
Simon, D., 52
simple function, 82
simulation, 28, 34
self-adjoint, 80
smooth number, 59
spectral
  function, 82
  projection, 2
  representation, 81
  theorem, 82
spectrum, 48, 72
state
  final state, 6
  initial state, 6
  mixed, 83
  pure, 83
  quantum states, 83
  states of automaton, 6
  vector, 2, 83
Stone, M.H., 86
superposition
  of configurations, 13, 20, 42
  decomposable, 23
  entangled, 23
  quantum -, 83
tape, 7
tensor product, 86
TIME, 17
time
  evolution, 23, 85
  complexity, 14
trace, 81
trace class, 81
tractable, 14
transition
  amplitude, 20
  amplitude function, 19
  function, 6
  relation, 10
  probability function, 12
Turing machine
  deterministic, 6
  nondeterministic, 10
  probabilistic, 12
  quantum, 19
unitary
  operation, 23
  operator, 85
ZPP, 66

References

[1] C.H. Bennett: *Logical Reversibility of Computation*, IBM Journal of Res. Development 17, p. 525-532, nov 1973.

[2] C.H. Bennett, Ethan Bernstein, G. Brassard and Umesh V. Vazirani: *Strengths and Weaknesses of Quantum Computation*, Los Alamos preprint, Dec 1, 1994.

[3] Ethan Bernstein and Umesh Vazirani: *Quantum Complexity Theory*, Proceedings of the 25th annual ACM symposium on the theory of computing, San Diego, California, 1993.

[4] André Berthiaume: *Quantum Computation*, a manuscript.

[5] André Berthiaume and Gilles Brassard: *Oracle Quantum Computing*, Proceedings of the Workshop on Physics and Computation - PhysComp '92, October 1992, IEEE Press, pp.195-199.

[6] P. Busch, P.J. Lahti and P. Mittelstaedt: *The Quantum Theory of Measurement*, Springer-Verlag, 1991.

[7] Vladimír Černý: *Quantum Computers and Intractable (NP-complete) Computing Problems*, Physical Review A vol 48, n. 1, pp. 116-119, July 1993.

[8] J.I. Cirac and P. Zoller: *Quantum Computations with Cold Trapped Ions*, Physical Review letters, Vol. 74, n.20, pp. 4091-4094, 1995.

[9] David Deutsch: *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*, Proceedings of the Royal Society of London, A400, pp. 97-117, 1985.

[10] David Deutsch: *Quantum Computational Networks*, Proceedings of the Royal Society of London, A 425, 73, pp. 73-90, 1989.

[11] David Deutsch, Adriano Barenco and Artur Ekert: *Universality in Quantum Computation*, Proceedings of the Royal Society of London A, Vol 449.

[12] Artur Ekert: *Quantum Computation*, Atomic Physics 14, AIP conference Proceedings 323 XIV International Conference on Atomic Physics, Boulder, CO, USA pp. 450-466 (AIP Press, New York 1995).

[13] Artur Ekert and Richard Josza: *Shor's Quantum Algorithm for Factorizing Numbers*, Clarendon Laboratory, University of Oxford, School of Mathematics and Statistics, University of Plymouth, a manuscript, to appear in the Rev. Mod. Phys.

[14] Richard P. Feynman: *Simulating Physics with Computers*, International Journal of Theoretical Physics, Vol 21, Nos 6/7, 1982.

[15] G.H. Hardy and E.M. Wright: *An Introduction to the Theory of Numbers*, 4th ed with corrections, Clarendon press, Oxford, 1971.

[16] Matti Kajala: *Diskreetin Logaritmin Laskentamenetelmistä*, Licentiate's thesis, University of Turku, 1991.

[17] Hua Loo Keng: *Introduction to Number Theory*, Springer-Verlag, 1982.

[18] Pekka J. Lahti and Kari Ylinen: *Johdatus Kvanttimekaniikkaan*, Gummerus kirjapaino OY, Jyväskylä 1989.

[19] Seth Lloyd: *A Potentially Realizable Quantum Computer*, Science, 261, pp. 1569-1571, September 17, 1993.

[20] Christos H. Papadimitriou: *Computational Complexity*, Addison-Wesley Publishing Company, Inc., 1994.

[21] M.B. Plenio and P.L. Knight: *Realistic Lower Bounds for the Factorization Time of Large Numbers on a Quantum Computer*, Physical Review A Vol 53, n. 5, pp. 2986-2990, may 1996.

[22] Peter W. Shor: *Algorithms for Quantum Computation: Discete Log and Factoring*, Proceedings of the 35th Annual Symposium on the Foundations of Computer Science, pp. 20-22, 1994.

[23] Daniel R. Simon: *On the Power of Quantum Computation*, Proceedings of the 35th Annual Symposium on the Foundations of Computer Science, 1994, pp. 116-123.

[24] Douglas R. Stinson: *Cryptography Theory and Practice*, CRC Press, Inc., Boca Raton, 1995.

[25] Päivi Törmä: *Manipulation of Quantum Information in Optical Networks*, Ph.D. dissertation, University of Helsinki, Department of physics, 1996.

[26] Andrew Chi-Chih Yao: *Quantum Circuit Complexity*, FOCS (pp. 352-361) 1993.

[27] Henri Cohen: *A Course in Computational Algebraic Number Theory*, Springer-Verlag, 1993.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**