The Complexity of Quantum Computation

By Takashi Mihara

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Doctor of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Susumu Kunifuji

January 1997

Abstract

Nowadays, there are many problems that are intractable to be solved on ordinary computers. Therefore, in order to invent more powerful computers, we need a computing model that is based on quantum physics, i.e., a *quantum computer*, because the computational principles of ordinary computers are based on classical physics, whereas the principles of the real world can be explained by quantum physics. In 1985, for the first time, Deutsch proposed a computing model involving a superposition of physical states, which is one of inherent properties of quantum physics, as a quantum Turing machine (QTM) and in 1993, Bernstein and Vazirani mathematically formalized the QTM. After that, some results have indicated that the QTM may be more powerful than ordinary computers. In 1994, in fact, Shor showed that the QTM can find discrete logarithms and factor integers in polynomial time with bounded error probability. We do not know whether ordinary computers can efficiently solve these problems or not. In this thesis, we present an overview on quantum computers and quantum complexity classes and then, we show some results on the complexity and algorithms based on the QTM.

Some techniques for solving problems efficiently on QTMs have been proposed. Especially, the most well-known techniques, a quantum Fourier transform and a quantum iterating method, have been used. A *quantum Fourier transform* is a quantum version of discrete Fourier transforms and can efficiently obtain some properties of functions. By this technique, we show that the periods in some kinds of periodic functions, $f(x) = f(x + r)$ and $x = f^r(x)$ for a period r, can be found in polynomial time with bounded error probability on QTMs. Some of the functions proposed as pseudo-random generators are also included in these functions.

A *quantum iterating method* is a method to increase the probability of accepting states by using an algorithm repeatedly. By this technique, we show that for an unsorted table T of n items and a query item q, there is a quantum search algorithm that finds a pair of indices (j, k) corresponding to two successive items, $T[j]$ and $T[k]$, which satisfy that $T[j] \leq q \leq T[k]$, in expected time $O(n^{1/2})$ with bounded error probability. As a special case, this algorithm can find the minimum or the maximum value of T in expected time $O(n^{1/2})$ with bounded error probability. Moreover, we also show that QTMs can solve some problems in computational geometry more efficiently than ordinary computers.

Finally, we investigate the relationships between the computational complexity and quantum physics. Although NP-complete problems appear in many situations, nobody knows whether ordinary computers can efficiently solve these problems or not. On the other hand, the problem of measurement is one of the most interesting problems in physics and nobody can explain sufficiently what happens when we measure yet. Here, we propose the following two assumptions on measurement: (i) Assumption Π_1 : a superposition of physical states is preserved after measurements and all of the states in the superposition can be measured in time proportional to the number of the states in the superposition, and (ii) Assumption Π_2 : we can measure the existence of a specific physical state C in a given superposition with certainty in polynomial time if the state C exists in the superposition. Then, we show that there is a QTM that solves the satisfiability problem (SAT) in $O(2^{n/4})$ time under Assumption Π_1 and there is a QTM that solves SAT in $n^{O(1)}$ time under Assumption Π_2, where n is the length of an instance of SAT. SAT is a typical NP-complete problem.

Acknowledgments

In the course of conducting this research, the author was fortunate to receive guidances and encouragements from a large number of people. Without them, this research would have never been accomplished. To begin with, the first and most important person to whom the author would like to express his sincere gratitude is his principal advisor Professor Susumu Kunifuji of Japan Advanced Institute of Science and Technology (JAIST). He gave the author his constant encouragements and kind guidances during this research.

The author would like to thank his advisor Associate Professor Satoshi Tojo of JAIST for his helpful discussions and suggestions. The author also would like to thank to Professor Hiroakira Ono and Professor Eiji Okamoto of JAIST for their kind comments and suggestions.

The author is grateful to all who have affected or suggested his areas of research. Associate Professor Tetsuro Nishino of the University of Electro-Communications inspired the author through his constant activities on quantum computer, and gave the author valuable comments and continuous encouragements.

The author is grateful to all who have read the manuscript and offered useful comments: Dr. Thanaruk Theeramunkong, Dr. Masahiro Mambo, Dr. Ratana Rujiravanit, Keisuke Tanaka, Shao-Chin Sung, and Hiroyuki Shirasu.

The author devotes his sincere thanks and appreciation to all of them, and his colleagues.

Contents

1 Introduction **1**

2 Preliminaries **5**
- 2.1 Notations 5
- 2.2 Turing Machines 5
- 2.3 Reversible Computation 7

3 Quantum Computers **12**
- 3.1 Quantum Computers before Deutsch's 12
  - 3.1.1 Simulating TMs by Quantum Physics 12
  - 3.1.2 Feynman's Universal Quantum Simulator 13
- 3.2 Quantum Turing Machine 13
  - 3.2.1 Definition of Quantum Turing Machine 13
  - 3.2.2 Physical Representation of QTM 15
- 3.3 Other Quantum Computing Models 17
  - 3.3.1 Quantum Circuits 17
  - 3.3.2 Quantum Cellular Automata 18
- 3.4 Realizabilities of Quantum Computers 20

4 Quantum Complexity Classes **22**
- 4.1 Ordinary Complexity Classes 22
  - 4.1.1 Complexity Classes Based on TMs 22
  - 4.1.2 Complexity Classes Based on PTMs 22
  - 4.1.3 Reducibility and Completeness 24
- 4.2 Quantum Complexity Classes and Relationships Between Classes 24
  - 4.2.1 BQP 25
  - 4.2.2 Other Quantum Complexity Classes 28
  - 4.2.3 Relationships Between Complexity Classes 29

5 Periodic Functions **32**
- 5.1 Introduction 32
- 5.2 Quantum Fourier Transforms 32
  - 5.2.1 Shor's Quantum Algorithm for Factoring Integers 32
  - 5.2.2 Various Transforms 34

5.3 Finding the Periods of Periodic Functions 39
5.3.1 Type $f(x) = f(x+r)$ 39
5.3.2 Type $x = f^r(x)$ 41

6 Quantum Search Algorithms 44
6.1 Introduction 44
6.2 Search Algorithms on QTMs 44
6.2.1 Grover's Quantum Search Algorithm 44
6.2.2 Extended Quantum Search Algorithm 46
6.3 Applications 49

7 The Complexity of NP-complete Problems 52
7.1 Introduction 52
7.2 Solving SAT under the First Assumption 53
7.3 Solving SAT under the Second Assumption 58

8 Conclusions 63

A Quantum Theory 65
A.1 Hilbert Spaces 65
A.2 Tensor Products 67
A.3 Unitary Operators 68
A.4 Quantum Physics 70

Bibliography 71

Publications 80

List of Figures

2.1 A Turing machine. 6

3.1 A Toffoli gate. 17

4.1 Relationships between complexity classes. 31

7.1 The QTM U. 54
7.2 The program that the QTM U executes under Assumption Π_1. 55
7.3 The computation executed by the QTM U. 57
7.4 The program that the QTM U executes under Assumption Π_2. 58
7.5 The changes of the superposition of configurations of the QTM U. 62

List of Tables

2.1 A reversible computing process. 11

Chapter 1

Introduction

A standard theoretical model of computing devices is a *Turing machine* that was proposed by Turing in 1930's. For some specific problems, the abilities of ordinary computers are more powerful than those of human beings. On the other hand, many problems that are intractable to be solved on ordinary computers are already known. Furthermore, the computational principles of ordinary computers are based on classical physics, whereas it is thought that the principles of the real world can be explained by quantum physics. Namely, it is well-known that classical physics is sufficient to explain macroscopic phenomena but is not sufficient to explain microscopic phenomena like the interference of electrons. In these days, the speed-up and down-sizing of computing devices have been carried out by using quantum physical effects, however, the computational principles of these devices are based on classical physics.

In this situation, from 1980's, physicists have seen the limitations of ordinary computers when they simulated physical phenomena on those computers. They thought that, since the current computational principles are based on classical physics, they might be able to make a more powerful computer if they could use the computational principles based on quantum physics, which can explain the real world more precise than classical physics. Thus, they proposed a computer that involves inherent properties of quantum physics as the computational principles and called it a *quantum computer*.

In early models of quantum computers, researchers placed a great importance on finding precise representations of Turing machines based on quantum physics (e.g., see [9, 11, 14, 93]). Thus, inherent properties of quantum physics were not involved in those models. In other words, their purpose was to design good simulators of Turing machines by using quantum physics.

On the other hand, Feynman and Deutsch proposed the models of quantum computers in which inherent properties of quantum physics are involved. In 1982, Feynman asked whether ordinary computers can *efficiently* simulate physical phenomena and pointed out that they may not be able to efficiently simulate physical phenomena [54]. Moreover, he suggested that a new computer based on quantum physics, a quantum computer, may be more powerful than ordinary computers. However, he did not formalize his quantum computer.

In 1985, for the first time, a quantum computer in which inherent properties of quantum physics are involved was proposed as a *quantum Turing machine* (*QTM*) by Deutsch [39]. However, Deutsch's quantum computer was not sufficiently formalized. So, he did not precisely evaluate the computing power of his quantum computer and did not mention anything about the computational complexity of his quantum computation.

In 1993, Bernstein and Vazirani mathematically formalized the QTM [23] and some results have indicated that the QTM seems to be more powerful than ordinary computers (e.g., see [22, 24, 25, 41, 68, 113]). In 1994, in fact, Shor showed that discrete logarithms and factoring integers are included in BQP, which is a language that can be accepted in polynomial time with bounded error probability on QTMs [108, 110]. These problems are generally considered hard on ordinary computers and have been used as the bases of several proposed cryptosystems.

Many physicists are also studying physical realizabilities of the QTM, and Lloyd reported that the QTM is potentially realizable [77, 78]. On the other hand, from computer scientific points of view, it is almost impossible to construct a new simulator for a QTM whenever its finite control is changed. Thus, it is important to answer the question whether a universal QTM exists. Deutsch showed that there is a universal QTM [39]. However, if we use his construction, the simulating overhead can be exponential in the running time of the simulated QTM in the worst case. After that, Bernstein and Vazirani showed that there is a universal QTM whose simulating overhead is polynomially bounded [23].

The remainder of this thesis is divided into six major chapters. In Chapter 2, we describe elementary definitions and notations. We define Turing machines that are standard theoretical models of computing devices. There are a lot of computing models that are known as Turing machines. Here, we define only two typical Turing machines, a deterministic Turing machine and a nondeterministic Turing machine. A QTM is defined based on these models. Since the execution of the QTM is evolved by unitary matrices, it is also a model of reversible computation. Therefore, in this chapter, we also describe the definition of a reversible Turing machine that was proposed by Bennett [15].

In Chapter 3, we review several models of quantum computers. A quantum computer was firstly proposed as a QTM by Deutsch [39]. His quantum computer, for the first time, involves inherent properties of quantum physics as the computational principles. Nowadays, many computer scientists study the complexity and algorithms based on his model. However, since there are other quantum computing models such as quantum circuits and quantum cellular automata, we also describe these models briefly. Finally, we summarize the realizabilities of quantum computers.

In Chapter 4, we define typical complexity classes based on ordinary Turing machines and complexity classes based on QTMs. QTMs can be regard as a kind of probabilistic Turing machines, because in quantum physics we can obtain results only probabilistically. Therefore, we define complexity classes based on QTMs as classes corresponding to the probabilistic complexity classes and call the complexity classes based on QTMs *quantum complexity classes* [23, 83]. Moreover, we also show the relationships between complexity classes.

In Chapter 5, we discuss methods to find the periods of functions efficiently. Shor showed that a QTM can find discrete logarithms and factor integers efficiently [108, 110]. No polynomial time algorithm to find these problems on ordinary computers is known. In order to solve these problems efficiently, he reduced them to the problems of finding the periods of some functions and used the most famous technique for solving problems efficiently on QTMs, a *quantum Fourier transform*.

We show that the periods in some kinds of periodic functions, $f(x) = f(x + r)$ and $x = f^r(x)$ for a period r, can be found in polynomial time with bounded error probability on QTMs by using the quantum Fourier transform [84]. For instance, we can efficiently solve a cycle problem on a QTM. Given a finite domain D, let us consider a function $f : D \to D$ such that it has $s + r$ distinct values $x_0 = f^0(x_0), f^1(x_0), \dots, f^{s+r-1}(x_0)$, but $f^s(x_0) = f^{s+r}(x_0)$. This means that $f^j(x_0) = f^{j+r}(x_0)$ for all $j \geq s$. A *cycle problem* for f and x_0 is to find the pair (s, r). Sedgewick et al. showed that ordinary computers need $O(n)$ time in order to solve the problem, where $n = s + r$ [106]. We show that a QTM can solve it in $(\log_2 n)^{O(1)}$ time. Some of the functions proposed as pseudo-random generators are also included in these functions.

In Chapter 6, we discuss quantum search algorithms for a table search (e.g., a database search). In order to efficiently search items in a table on a QTM, we use another well-known technique, a quantum iterating method. A *quantum iterating method* was proposed by Grover and is a method to increase the probability of accepting states by using an algorithm repeatedly. He showed that for an unsorted table T of n distinct items, $T[0], T[1], \dots, T[n-1]$, there is a quantum search algorithm that finds the index m of a query item $q(= T[m])$ in expected time $O(n^{1/2})$ with bounded error probability [60]. Ordinary computers need $O(n)$ time to find the index.

We show that for an unsorted table T of n items and a query item q (where q may not necessarily exist in T), there is a quantum search algorithm that finds a pair of indices (j, k) corresponding to two successive items, $T[j]$ and $T[k]$, which satisfy that $T[j] \leq q \leq T[k]$, in expected time $O(n^{1/2})$ with bounded error probability. Our algorithm uses Grover's search algorithm as a subroutine, and as a special case, we can find the minimum or the maximum value of T by using this algorithm in expected time $O(n^{1/2})$ with bounded error probability. Moreover, we also show that QTMs can solve some problems in computational geometry more efficiently than ordinary computers.

In Chapter 7, we discuss methods to solve NP-complete problems on QTMs. Although NP-complete problems appear in many situations, nobody knows whether ordinary computers can efficiently solve these problems or not. It is one of the most important issues in theoretical computer science to find a method to solve NP-complete problems efficiently.

Even if a QTM can compute all the values of a function by using quantum parallel computation, we cannot, in general, obtain all the values of the function simultaneously. Moreover, according to current quantum physics, it is not certain whether we can efficiently read each value in the obtained superposition. These are because the following measurement problem has not been completely solved.

Measurement Problem in Quantum Physics :

> What will happen when we measure a quantum physical object ?
> Explain it in terms of quantum physics.

Therefore, we study the relationships between the assumptions on measurement and the efficiency of quantum computation. Especially, we study the satisfiability problem (SAT), because SAT is a typical NP-complete problem [56]. Here, we propose the following two assumptions on measurement: (i) Assumption Π_1 : a superposition of physical states is preserved after measurements and all of the states in the superposition can be measured in time proportional to the number of the states in the superposition, and (ii) Assumption Π_2 : we can measure the existence of a specific physical state C in a given superposition with certainty in polynomial time if the state C exists in the superposition. Consequently, we show that a QTM can solve SAT in $O(2^{n/4})$ time under Assumption Π_1 and a QTM can solve SAT in $n^{O(1)}$ time under Assumption Π_2, where n is the length of an instance of SAT [81, 82].

The assumptions above are not widely supported in current quantum physics, however, nobody knows whether these assumptions are valid or not. This is because *interpretations of measurement* have not been fixed yet among physicists. The measurement problem is one of the central issues in quantum physics and several interpretations of measurement exist. In this situation, it is important to find various relationships between the restrictions on measurement and the efficiency of quantum computation.

Chapter 2

Preliminaries

2.1 Notations

In this section, we describe some notations that are used in this thesis.

1. The Sets
   - $\mathbf{C}$: the set of complex numbers.
   - $\mathbf{R}$: the set of real numbers.
   - $\mathbf{Z}$: the set of integers.
   - $\mathbf{Z}^+$: the set of positive integers.
   - $\mathbf{Z}_n$: $\{0, 1, \ldots, n-1\}$.
   - $\mathbf{N}$: the set of natural numbers
2. O-notation
   - $O(f)$ is the set of functions g such that for some constant $c > 0$ and for all but finitely many n, $g(n) < cf(n)$.
   - $\Omega(f)$ is the set of functions g such that for some constant $c > 0$ and for infinitely many n, $g(n) > cf(n)$.
   - $\Theta(f)$ is the set of functions g such that for some constants $c_1, c_2 > 0$ and for all but finitely many n, $c_1 f(n) < g(n) < c_2 f(n)$.

2.2 Turing Machines

A Turing machine is a kind of mathematical computing machine model. The definition of a Turing machine is described in this section. For a more complete overview, the reader is referred to [3, 61, 91]. There are a lot of computing models that are known as Turing machines. Here, we denote only two typical Turing machines, a deterministic Turing machine and a nondeterministic Turing machine.

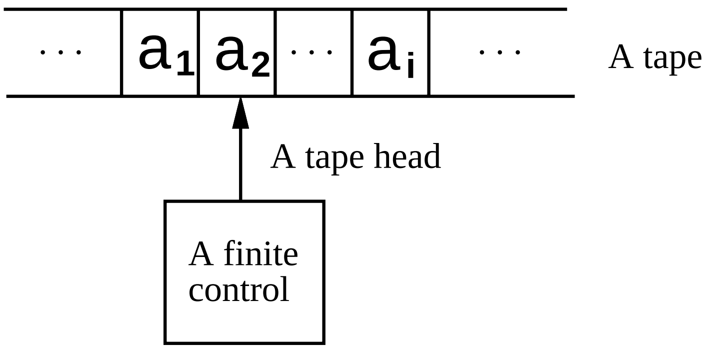

Figure 2.1: A Turing machine.

As shown in Fig. 2.1, a *deterministic Turing machine* consists of a finite control, an infinite tape, and a tape head. In one move, the Turing machine, depending on the symbol scanned by the tape head and the state of the finite control,

1. changes state,
2. writes a symbol on the tape cell scanned, replacing what was written there, and
3. moves its head left one cell, right one cell, or keep it stationary.

Definition 2.1. *A deterministic Turing machine(DTM) is a 7-tuple* $M = (Q, \Sigma, \Gamma, \delta, q_0, b, F)$*, where*

> Q *is the finite set of states,*
> Γ *is the finite set of tape symbols,*
> $b \in \Gamma$ *is a blank symbol,*
> $\Sigma \subseteq \Gamma$ *is the set of input symbols,*
> δ *is the state transition function that is a mapping from* $Q \times \Gamma$ *to* $Q \times \Gamma \times \{-, 0, +\}$*,*
> $q_0 \in Q$ *is the initial state,*
> $F \subseteq Q$ *is the set of final states.*

Here, $\{-, 0, +\}$ *in the state transition function* δ *decides the movement of its head, i.e.,* $-$*(resp.* 0*,* $+$*) means the movement of its head left one cell (resp. stationary, right one cell).*

A *nondeterministic Turing Machine* (*NDTM*) consists of a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where all components have the same definition as the DTM, except that the state transition function δ is a mapping from $Q \times \Gamma$ to subsets of $Q \times \Gamma \times \{-, 0, +\}$. Here, we defined only one-tape Turing machines, however, we can also define many-tape Turing machines. Furthermore, a *universal Turing machine* is defined as a Turing machine that can simulate any Turing machine with any input.

Nowadays, a Turing machine has become a typical computing model. Nobody can prove that a Turing machine is equivalent to our intuitive notion of computation, however, there are many arguments for this equivalence, which has become known as Church-Turing thesis.

Church-Turing Thesis:

> The intuitive notion of *computable function* can be identified with the class of computation with a Turing machine.

Therefore, a Turing machine is equivalent to all of the most general mathematical notions of computation. Furthermore, Deutsch reinterpreted the Church-Turing thesis physically from a viewpoint of constructing a real computer and called it *Church-Turing principle.* That is, a quantum computer is a computing model proposed under this interpretation (cf. Chapter 3).

2.3 Reversible Computation

For a long time, a computation was thought as an irreversible process. However, Bennett constructed a logical reversible Turing machine and showed that the reversible Turing machine can simulate an irreversible computation with constant slowdown [15, 16, 17, 18, 76].

As shown in Section 2.2, the state transition function δ of a one-tape Turing machine can be represented by the following quintuple.

$$At \rightarrow A't'\sigma, \tag{2.1}$$

where

A and A' are the states before and after the transition, respectively,

t is the tape symbol that is read at the head position,

t' is the tape symbol that will be written at the head position, and

σ is the movement of the head (i.e., $\sigma \in \{-, 0, +\}$).

A Turing machine is *deterministic* if and only if its quintuples have non-overlapping domains, and is *reversible* if and only if they have non-overlapping ranges. It was thought that a computation can be executed only irreversibly because the ranges of quintuples overlap. Therefore, a usual Turing machine is not reversible. However, Bennett showed that a reversible computation can be executed with splitting the state transition function into a read-write operation and a head movement operation, and a reversible Turing machine can simulate an irreversible computation with constant slowdown. Here, we denote Bennett's construction of a reversible Turing machine (reversible TM) [15].

First, we define the following quadruple for a reversible TM that corresponds to the usual state transition function.

Definition 2.2. *A quadruple for an n-tape Turing machine having one head per tape is defined by*

$$A[t_1, t_2, \ldots, t_n] \rightarrow A'[t'_1, t'_2, \ldots, t'_n], \tag{2.2}$$

where

> *A and A' are the states before and after the transition, respectively,*
>
> *t_k is the tape symbol that is read on the kth tape or / (indicating that the kth tape is not read during the transition), and*
>
> *t'_k is the tape symbol that will be written on the kth tape or the movement of the kth head (i.e., $\sigma_k \in \{-, 0, +\}$).*

Note that a usual Turing machine executes the read-write operation and the head movement operation at the same time, on the other hand, the reversible TM writes on the tape if and only if it has just read it, and moves the tape head if and only if it has not just read it. Thus, the following quadruples define the mappings of the whole-machine states that are one-to-one. Any read-write-movement quintuple can be split into a read-write operation and a head movement operation. Therefore, the quintuple of Eq. (2.1) is equivalent to a pair of quadruples.

$$\begin{array}{lcl} A[t_1, t_2, \ldots, t_n] & \rightarrow & A"[t'_1, t'_2, \ldots, t'_n], \quad \text{and} \\ A"[/, /, \ldots, /] & \rightarrow & A'[\sigma_1, \sigma_2, \ldots, \sigma_n], \end{array}$$

where $A"$ is a new state different from A and A'. When several quintuples are so split, a different connecting state $A"$ must be used for each to avoid introducing indeterminacy.

Now, let us consider the following two n-tape quadruples of Eq. (2.2).

$$A[t_1, \cdots, t_n] \rightarrow A'[t'_1, \cdots, t'_n], \tag{2.3}$$

and

$$B[u_1, \cdots, u_n] \rightarrow B'[u'_1, \cdots, u'_n]. \tag{2.4}$$

They have the following additional important properties:

1. Eqs. (2.3) and (2.4) are mutually inverse if and only if

$$\begin{array}{l} (A = B') \wedge (B = A') \\ \quad \wedge(\forall k((t_k = u_k = /) \wedge (t'_k = -u'_k) \vee (t_k \neq /) \wedge (t'_k = u_k) \wedge (t_k = u'_k))). \end{array}$$

2. The domains of Eqs. (2.3) and (2.4) overlap if and only if

$$(A = B) \wedge (\forall k((t_k = /) \vee (u_k = /) \vee (t_k = u_k))).$$

3. The range of Eqs. (2.3) and (2.4) overlap if and only if

$$(A' = B') \land (\forall k((t_k = /) \lor (u_k = /) \lor (t'_k = u'_k))).$$

Thus, an *n-tape reversible deterministic Turing machine* is defined as a finite set of n-tape quadruples of Eq. (2.2), no two of which overlap either in domain or in range.

Next, we define a *standard* Turing machine.

Definition 2.3. *An input or output is said to be standard when it is on otherwise blank tape and contains no embedded blanks, when the tape head scans the blank cell immediately to the left of it, and when it includes only symbols belonging to the tape alphabet of the machine scanning it.*

Definition 2.4. *A standard one-tape Turing machine is a finite set of one-tape quintuples*

$$At \rightarrow A't'\sigma,$$

which satisfies the following properties:

1. *Determinism*

   No two quintuples agrees in both A and t.

2. *Format*

   If the Turing machine starts in control state A_1 on any standard input, it will halt in control state A_f, leaving its output in standard format.

3. *Special quintuples*

   The machine includes the following two special quintuples: the initial quintuple

   $$A_1 b \rightarrow A_2 b +,$$

   and the final quintuple

   $$A_{f-1} b \rightarrow A_f b 0,$$

   and the states A_1 and A_f appear in no other quintuple.

Moreover, we denote that a one-tape Turing machine M is given a standard input string I and computes a standard output string O by

$$M : I \rightarrow O.$$

For an n-tape machine, we denote that M is given standard input strings, $I_1, I_2, \cdots, I_n$, and computes standard output strings $O_1, O_2, \cdots, O_n$ by

$$M : (I_1; I_2; \cdots; I_n) \rightarrow (O_1; O_2; \cdots; O_n).$$

Then, Bennett showed the following relationships between a usual Turing machine and a reversible Turing machine.

Theorem 2.1 [15].

1. *For every standard one-tape deterministic Turing machine M, there is a three-tape reversible deterministic Turing machine R such that if I and O are strings on the alphabet of M, then, M halts on I if and only if R halts on $(I; b; b)$, and $M : I \to O$ if and only if $R : (I; b; b) \to (I; b; O)$.*
2. *If M has f control states, N quintuples, and a tape alphabet of z symbols, including the blank, R will have $2f + 2N + 4$ states, $4N + 2z + 3$ quadruples, and a tape alphabet of z, $N + 1$, and z symbols, respectively.*
3. *In a particular computation, if M requires t steps and uses s tape cells, producing an output of length l, then, R will require $4t + 4l + 5$ steps and use s, $t + 1$, and $l + 2$ cells on its three tapes, respectively.* $\square$

We do not provide the proof of this theorem. In Table 2.1, we only denote how the reversible TM R can simulate the usual Turing machine. Thus, the reversible TM R is constructed in such a way that R will record the *history* of its state transitions on the second tape. That is, each state transition rule (i.e., quadruple) of R has its own number and when R changes its state by using the kth rule, R will write the number k on the second tape. Nobody knows whether there is a reversible TM that does not make histories (i.e., working data) or not.

Table 2.1: A reversible computing process.

Stage	Quadruples	Work Tape	History Tape	Output Tape
		_INPUT	_	_
	1) $A_1[b,/,b] \to A'_1[b,+,b]$			
	$A'_1[/,b,/] \to A_2[+,1,0]$			
	$\vdots$			
Compute	m) $A_j[t,/,b] \to A'_m[t',+,b]$			
	$A'_m[/,b,/] \to A_k[\sigma,m,0]$			
	$\vdots$			
	N) $A_{f-1}[b,/,b] \to A'_N[b,+,b]$			
	$A'_N[/,b,/] \to A_f[0,N,0]$			
		_OUTPUT	HISTOR$\underline{Y}$	_
	$A_f[b,N,b] \to B'_1[b,N,b]$			
	$B'_1[/,/,/] \to B_1[+,0,+]$			
	x $\neq$ b:{ $B_1[x,N,b] \to B'_1[x,N,x]$ }			
Copy Output[1]	$B_1[b,N,b] \to B'_2[b,N,b]$			
	$B'_2[/,/,/] \to B_2[-,0,-]$			
	x $\neq$ b:{ $B_2[x,N,x] \to B'_2[x,N,x]$ }			
	$B_2[b,N,b] \to C_f[b,N,b]$			
		_OUTPUT	HISTOR$\underline{Y}$	_OUTPUT
	N) $C_f[/,N,/] \to C'_N[0,b,0]$			
	$C'_N[b,/,b] \to C_{f-1}[b,-,b]$			
	$\vdots$			
Retrace	m) $C_k[/,m,/] \to C'_m[-\sigma,b,0]$			
	$C'_m[t',/,b] \to C_j[t,-,b]$			
	$\vdots$			
	1) $C_2[/,1,/] \to C'_1[-,b,0]$			
	$C'_1[b,/,b] \to C_1[b,-,b]$			
		_INPUT	_	_OUTPUT

[1]In the second stage, the small braces indicate sets of quadruples with one quadruple for each non-blank tape symbol x.

Chapter 3

Quantum Computers

3.1 Quantum Computers before Deutsch's

A quantum computer was proposed as a quantum Turing machine by Deutsch [39]. His computing model, for the first time, involves *inherent properties* of quantum physics as the computational principles. Nowadays, many computer scientists study the complexity and algorithms based on his model. Many researches are also done in order to realize his computing model. However, there were also some quantum computers proposed before Deutsch's.

3.1.1 Simulating TMs by Quantum Physics

In early models of quantum computers such as those of Benioff, researchers placed a great importance on finding precise representations of Turing machines based on quantum physics. Therefore, inherent properties of quantum physics were not involved in those models. In other words, their purpose was to design good simulators of Turing machines by using quantum physics. Thus, they did not proposed a new type of computing model based on quantum physics.

In early 1980's, Benioff had studied whether a Turing machine can be simulated by computing models based on quantum physics or not. His research was to construct a physical system that simulates a Turing machine, i.e., it was to construct a Hamiltonian operator in Schrödinger's wave equation (see Eq. (A.1)). Therefore, his computing model was called a *quantum mechanical Hamiltonian model.* His early quantum computers were models to simulate irreversible Turing machines [9, 10]. After that, he also proposed quantum computers simulating Bennett's reversible Turing machine because closed physical systems act reversibly [11, 12, 13, 14].

Furthermore, Peres and Zurek had proposed Benioff type of quantum computers recovering errors of hardware [93, 128, 129, 130]. However, since none of these quantum computers used inherent properties of quantum physics (e.g., quantum superposition, interference, uncertainty), the powers of these computers were equivalent to those of ordinary Turing machines.

3.1.2 Feynman's Universal Quantum Simulator

Feynman took a different approach from Benioff, i.e., he asked,

> "What type of computers can *efficiently* simulate physical phenomena?".

Consequently, he concluded that a new type of computer based on quantum physics must be proposed to *efficiently* simulate physical phenomena, because the computational principles of ordinary computers are based on classical physics, whereas it is thought that the principles of the real world can be explained based on quantum physics. He called it a *universal quantum simulator* [54]. In order to realize computers that simulate physical phenomena efficiently, we naturally require a computer based on quantum physics, i.e., a quantum computer. Thus, for the first time, he suggested that the quantum computer will become more powerful than ordinary computers [54]. Furthermore, some researchers have mentioned that we should study the relationships between computation and physics (e.g., see [19, 20, 71, 72, 73, 74]).

3.2 Quantum Turing Machine

3.2.1 Definition of Quantum Turing Machine

We define a quantum computer as a quantum Turing machine. A quantum Turing machine was proposed by Deutsch [39] and was mathematically formalized by Bernstein and Vazirani [23]. In [39], Deutsch proposed a physical version of Church-Turing thesis as Church-Turing principle,

> *Church-Turing Principle*:
>
> > Every finitely realizable physical system can be perfectly simulated
> > by a universal model computing machine operating by finite means,

and he defined a quantum Turing machine under this principle. Like an ordinary Turing machine, a quantum Turing machine M also consists of a finite control, an infinite tape, and a tape head.

Definition 3.1. *A quantum Turing machine (QTM) is a 7-tuple*
$M = (Q, \Sigma, \Gamma, \delta, q_0, b, F)$, *where*

> Q *is the finite set of states,*
> Γ *is the finite set of tape symbols,*
> $b \in \Gamma$ *is a blank symbol,*
> $\Sigma \subseteq \Gamma$ *is the set of input symbols,*
> δ *is the state transition function that is a mapping from* $Q \times \Gamma \times \Gamma \times Q \times \{-, +\}$ *to* $\mathbf{C}$,
> $q_0 \in Q$ *is the initial state,*
> $F \subseteq Q$ *is the set of final states.*

A *configuration* of the QTM M is defined by a triple (p, h, a), where $p \in Q$ denotes a current state, $h \in \mathbf{Z}$ denotes an index of the tape cell over which the tape head is currently located, and $a \in \Gamma$ denotes a tape symbol in the tape cell over which the tape head is currently located. A *state transition function* $\delta(p, a, b, q, d) = A_m$ (we call A_m an *(probability) amplitude*) denotes that, if M reads a symbol a in a state p (let C_1 be this configuration of M), M writes a symbol b on the cell under the tape head, changes the state into q, and moves the head one cell in the direction of $d \in \{-, +\}$(let C_2 be this configuration of M). Then, we define the probability that M changes its configuration from C_1 to C_2 by $|A_m|^2$.

Furthermore, this state transition function δ corresponds to a *time evolution matrix* U_δ of M as follows: each row and column of U_δ corresponds to a configuration of M. Let C_1 and C_2 be two configurations of M, then the element corresponding to C_2 row and C_1 column of U_δ is the value of δ evaluated at the tuple that transforms C_1 into C_2 in one step. If no such tuple exists, the corresponding element is zero. Moreover, from physical restrictions, the time evolution matrix U_δ must be a *unitary matrix.* Namely, relations $U_\delta^\dagger U_\delta = U_\delta U_\delta^\dagger = I$ must be satisfied by U_δ, where $U_\delta^\dagger$ is the transposed conjugate of U_δ and I is the unit matrix.

Since any time evolution matrix U_δ is regular, the QTM is a kind of reversible computing machine. Namely, the QTM directly simulates a reversible deterministic Turing machine (reversible DTM) in the operations of ordinary Turing machines. The existence of a reversible DTM was shown by Bennett [15] (see also Sec. 2.3). Thus, the QTM can execute all operations of the reversible DTM and eight types of unitary transforms for two-state space [39].

$$\left.\begin{array}{l}
V_0 = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}, \quad V_1 = \begin{pmatrix} \cos\alpha & i\sin\alpha \\ i\sin\alpha & \cos\alpha \end{pmatrix}, \\[2ex]
V_2 = \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & 1 \end{pmatrix}, \quad V_3 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}, \\[2ex]
V_4 = V_0^{-1}, \quad V_5 = V_1^{-1}, \quad V_6 = V_2^{-1}, \quad \text{and} \quad V_7 = V_3^{-1},
\end{array}\right\} \tag{3.1}$$

where α is an arbitrary irrational multiple of π. Here, we take the following convention:

> The QTM executes one step of the reversible DTM in one step and also executes each one of the eight types of transforms above in one step.

When the QTM simulates an execution of the reversible DTM, it will move as follows. Let q be a state of the DTM written on the (work) tape and a be a symbol scanned by the head of the DTM. Then, the QTM will scan the set P_M of the state transition rules of the DTM written on the tape from the leftmost square of P_M to right and will find only one state transition rule of the form $\delta(q, a) = \cdots$ (we assume that P_M always has one such a rule). When the QTM finds such a rule, it memorizes the rule by its finite control and moves the tape head to the right until the rightmost square of P_M is reached. If the tape head reaches to the rightmost square of P_M, the QTM moves the tape to the leftmost square of P_M. After that, the QTM will change the configuration of the DTM written on the tape according to the found state transition rule. From this, in all configurations in a superposition, the QTM can simulate a single step of the DTM in the same number of steps.

In general, since a QTM simulates the movement of a reversible DTM, the QTM must also leave histories. However, it is known that we can compute the value of a function without histories with constant slowdown as long as we keep the input[17]. Furthermore, Bernstein and Vazirani showed that there is a universal QTM whose simulating overhead for any QTM is polynomially bounded [23].

3.2.2 Physical Representation of QTM

Next, let us consider a physical representation of QTM. We can construct one-bit, a minimum unit of information, by using a two-state physical system (e.g., a spin-$\frac{1}{2}$ system, etc.) and call it a *qubit* [104]. A qubit has a chosen *computational basis* $\{|0\rangle, |1\rangle\}$ corresponding to ordinary bit values 0 and 1, and we define by

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

For n qubits ($n \geq 2$), we can construct as a composed system $|x_1, x_2, \dots, x_n\rangle$ of simultaneous observable two-state physical systems. Namely, it is represented as the tensor products of two-state physical systems as follows:

$$|x_1, x_2, \dots, x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle,$$

where $x_i \in \{0, 1\}$ for $i = 1, 2, \dots, n$. A collection of n qubits $|x_1, x_2, \dots, x_n\rangle$ is called a *register* of size n.

Since the QTM consists of a finite control, an infinite tape, and a tape head, it will be constructed as a composed system of physical systems corresponding to these three components. Let $|C\rangle$ be a physical system corresponding to the finite control, $|T\rangle$ be a physical system to the tape, and $|H\rangle$ be a physical system to the tape head. Each of these physical systems is also constructed as a composed system of two-state physical systems (e.g., $|C\rangle = |c'_1\rangle \otimes |c'_2\rangle \otimes \ldots \otimes |c'_u\rangle$, where $c'_i \in \{0, 1\}$ for $i = 1, 2, \dots, u$). Then, a physical system $|M\rangle$ corresponding to the QTM is represented as a composed system of these physical systems as follows:

$$|M\rangle = |C\rangle \otimes |H\rangle \otimes |T\rangle.$$

In general, a physical state of the QTM corresponds to a superposition of configurations of the QTM. Namely, when $|C\rangle = \sum_{i=1}^{l} |c_i\rangle$, $|H\rangle = \sum_{j=1}^{m} |h_j\rangle$, and $|T\rangle = \sum_{k=1}^{n} |t_k\rangle$, then,

$$|M\rangle = \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{n} |c_i\rangle \otimes |h_j\rangle \otimes |t_k\rangle.$$

If a physical state of the QTM is not a superposed one, the state of the QTM is equal to a configuration of the QTM.

Furthermore, when the QTM has r tapes $T_1, \ldots, T_r$, the physical state of the QTM will be represented as follows:

$$|M\rangle = |C\rangle \otimes |H_1\rangle \otimes \cdots \otimes |H_r\rangle \otimes |T_1\rangle \otimes \cdots \otimes |T_r\rangle,$$

where $H_1, \ldots, H_r$ are heads on $T_1, \ldots, T_r$, respectively.

Finally, we denote the movements of the QTM. A *computation* on the QTM is an evolving process of the physical system defined by the unitary matrix U_δ. Let $|\psi(0)\rangle$ be an initial state (a state at time zero) of the QTM that is represented as follows:

$$|\psi(0)\rangle = |q_0\rangle \otimes |1\rangle \otimes |T_0\rangle,$$

where T_0 denotes a tape content before the execution. When we denote the state at time s by $|\psi(s)\rangle$,

$$|\psi(\tau t)\rangle = U_\delta^t |\psi(0)\rangle,$$

where τ is the time required by the QTM to execute one step. The tape content will be obtained with physical *measurements* (i.e., with observations of the physical systems constructing the QTM) as follows:

When a superposition of configurations $|\psi\rangle = \sum_i \alpha_i |c_i\rangle$ is written on the tape, we can measure φ component of ψ with probability $|\langle\varphi|\psi\rangle|^2$. Especially, we can measure c_i component of ψ with probability $|\alpha_i|^2$.

For instance, let us consider an n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$. First, for the initial configuration $|\underbrace{0, 0, \ldots, 0}_{n}, 0\rangle$, a QTM makes all the input assignments of f. This is performed by applying

$$V_4 = \frac{1}{2^{1/2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

to each qubit corresponding to n variables in order, where let $\alpha = \pi/4$.

$$\begin{aligned} &|0, 0, \ldots, 0, 0\rangle \\ &\quad \to \frac{1}{2^{n/2}} \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} |x_1, x_2, \ldots, x_n, 0\rangle. \end{aligned}$$

Next, the QTM computes the values of f.

$$\begin{aligned} &\frac{1}{2^{n/2}} \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} |x_1, x_2, \ldots, x_n, 0\rangle \\ &\quad \to \frac{1}{2^{n/2}} \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} |x_1, x_2, \ldots, x_n, f(x_1, x_2, \ldots, x_n)\rangle. \end{aligned}$$

All the computations of f are executed in parallel and this type of computation is called a *quantum parallel computation* [39, 66].

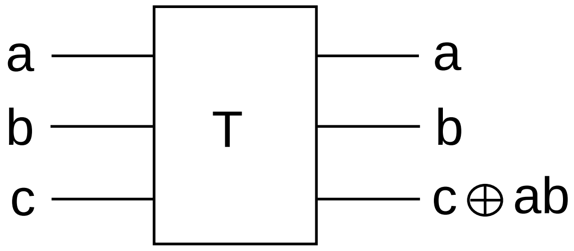

Figure 3.1: A Toffoli gate.

3.3 Other Quantum Computing Models

The most popular computing model of quantum computers is the QTM. However, there are also other quantum computing models. In this section, we briefly review quantum circuits and quantum cellular automata.

3.3.1 Quantum Circuits

Quantum circuits were first proposed by Feynman [55] and Deutsch showed that there is a universal quantum gate [40]. A gate is *universal* if we can construct any gate by using the gate. Although quantum gates and quantum circuits are important for constructing QTMs, here, we only denote some results on the complexity of quantum circuits. For the realizabilities of QTMs, see Sec. 3.4.

A quantum gate is a kind of reversible gates because QTMs are reversible computers, and quantum circuits consist of quantum gates, which are a generalization of ordinary (or classical) logic gates. For ordinary gates, Toffoli showed that a three-bit Toffoli gate (one of reversible gates) is universal [119] (see Fig. 3.1), whereas it is not known whether there are universal two-bit ordinary reversible gates.

For quantum gates, for the first time, Deutsch showed that the following three-bit quantum gate is universal [40].

$$U_\lambda^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cos\lambda & i\sin\lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & i\sin\lambda & \cos\lambda \end{pmatrix}. \tag{3.2}$$

Since we know that a physical evolution can be represented by a unitary matrix (operator) from Eq. (A.2), we can also represent quantum gates by the same way. In this case, the computational basis is $\{|0,0,0\rangle, |0,0,1\rangle, |0,1,0\rangle, |0,1,1\rangle|1,0,0\rangle, |1,0,1\rangle, |1,1,0\rangle, |1,1,1\rangle\}$ and λ is an irrational multiple of π. He also showed the following results.

Theorem 3.1 [39]. *Let U be any d-dimensional unitary matrix. Then, U can be written as a product of $2d^2-d$ unitary matrices, each of which acts only within a two-dimensional subspace spanned by a pair of computational basis states (i.e., Eq. (3.1)).* □

Theorem 3.2 [40]. *Let $n \geq 1$. Any unitary transform in $\mathbf{C}^{2^n}$ (as induced by n Boolean variables) can be computed by a quantum Boolean circuit by using $2^{O(n)}$ elementary gates from Φ_3 and with $O(n)$ auxiliary wires, where Φ_3 is the set of all three-bit quantum gates.* □

After that, DiVincenzo showed that there is a universal two-bit quantum gate [44, 45, 115]:

$$U_\lambda^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\lambda & i\sin\lambda \\ 0 & 0 & i\sin\lambda & \cos\lambda \end{pmatrix}, \tag{3.3}$$

where the computational basis is $\{|0,0\rangle, |0,1\rangle, |1,0\rangle, |1,1\rangle\}$ and λ is an irrational multiple of π. This will be a feature on quantum gates because nobody knows whether there is a universal two-bit ordinary reversible gate or not. Moreover, it is also shown that almost any two-bit quantum gate is universal [6, 42, 79].

Finally, Yao studied the *quantum circuit complexity* and showed the relationships between QTMs and quantum circuits [127]. For any language $L \subseteq \{0,1\}^n$, let $C_Q(L)$ be the minimum circuit size for any quantum circuit computing L. A quantum Boolean circuit K with n input variables is said to (n,t)*-simulate* a QTM M, if the family of probability distributions $p_{\tilde{x}}$, $\tilde{x} \in \{0,1\}^n$ generated by K is identical to the distribution of the configuration of M after t steps with $\tilde{x}$ as input.

Theorem 3.3 [127]. *Let M be a QTM and n, t be positive integers. Then, there is a quantum Boolean circuit K of size poly(n,t) that (n,t)-simulates M.* □

Corollary 3.1 [127]. *If $L \in P$, then, $C_Q(L_n) = O(n^k)$ for some fixed k, where L_n is the set of strings in L of length n.* □

3.3.2 Quantum Cellular Automata

Watrous defined one-dimensional quantum cellular automata and studied the relationships between QTMs and quantum cellular automata [125]. Moreover, Dürr et al. also defined them independently and called them linear quantum cellular automata [48]. Here, we denote linear quantum cellular automata in [48].

The cells of an automaton are organized in a line and are indexed by $\mathbf{Z}$.

Definition 3.2 [48]. *A linear quantum cellular automaton(LQCA) is a quadruple $M = (\Sigma, q, N, \delta)$, where*

Σ is the finite set of (cell-)states,
N is the neighborhood,
δ is the local transition function that is a mapping from $\Sigma^r \times \Sigma$ to $\mathbf{C}$, which satisfies that for every $(p_1, \ldots, p_r) \in \Sigma^r$, there is $p \in \Sigma$ such that $\delta(p_1, \ldots, p_r, p) \neq 0$, where $r = |N|$, and
$q \in \Sigma$ is the distinguished quiescent state, which satisfies

$$\delta(q, \ldots, q, p) = \begin{cases} 1 & \text{if } p = q, \\ 0 & \text{otherwise.} \end{cases}$$

The states of the cells are changing simultaneously at every step according to the local transition function δ. If at some step the neighbors of a cell are in $p_1, \ldots, p_r$, then, at the next step the cell will change into state p with amplitude $\delta(p_1, \ldots, p_r, p)$. The neighborhood $N = (a_1, \ldots, a_r)$ is a strictly increasing sequence of integers for some $r \geq 1$, giving the addresses of the neighbors relative to each cell. This means that the neighbors of cell i are indexed by $i + a_1, \ldots, i + a_r$. The set of *configurations* is defined by $\Sigma^{\mathbf{Z}}$, where for every configuration c and every integer i, the state of the cell indexed by i is $c(i)$. Moreover, the local transition function δ of the LQCA M induces the *time evolution operator*

$$U_\delta : \mathcal{C}_M \times \mathcal{C}_M \to \mathbf{C},$$

where $\mathcal{C}_M$ is the set of finite configurations and $U_M(d, c)$ is the transition amplitude of changing configuration c to configuration d in one step. It is defined by

$$U_M(d, c) = \prod_{i \in \mathbf{Z}} \delta(c(i + a_1), \ldots, c(i + a_r), d(i)).$$

Thus, the LQCA differs from an ordinary one in the sense that the automaton evolves on a superposition of configurations like QTMs.

Furthermore, a *linear partitioned quantum cellular automaton* (*LPQCA*) is a LQCA in which each cell is partitioned into three *subcells*: a left subcell, a middle subcell, and a right subcell (and the set Σ of states is decomposed accordingly). The LPQCA, which is a generalization of (deterministic) partitioned cellular automata discussed by Morita and Harao [89], is a restricted class of the LQCA.

Watrous showed that any QTM can be efficiently simulated by a LPQCA with constant slowdown and any LPQCA can be simulated by a QTM with linear slowdown.

Theorem 3.4 [125]. *Given any QTM M_{qtm}, there is an LPQCA M_{pqca} that simulates M_{qtm} with constant slowdown.* $\Box$

Theorem 3.5 [125]. *Given any LPQCA M_{pqca}, there is a QTM M_{qtm} that simulates M_{pqca} with linear slowdown.* $\Box$

It is an open problem whether there is a QTM that simulates any LQCA efficiently or not.

A LQCA M is *well-formed* if the time evolution operator U_δ preserves the norm, and it is *unitary* if U_δ is a unitary transform. Moreover, we call an *interval* a finite subset of consecutive integers $\{j, j+1, \ldots, k\}$ of $\mathbf{Z}$ for any j and k (if $j > k$, this defines the empty interval $\emptyset$), and call *simple* if the integers in the neighborhood N form an interval. In [48, 49], it is shown that there is an efficient algorithm to decide whether a LQCA is well-formed and unitary. Here, we define the size of the automaton by $n = |\Sigma|^{r+1}$. Further, let $N = (a_1, \ldots, a_r)$, $s = a_r - a_1 + 1$, and $e = (s+1)/(r+1)$.

Theorem 3.6 [48]. *There is an algorithm that takes a simple LQCA as input and decides in $O(n^2)$ time whether it is well-formed.* □

Corollary 3.2 [48]. *There is an algorithm that takes a LQCA as input and decides in $O(n^{2e})$ time whether it is well-formed.* □

Theorem 3.7 [49]. *There is an algorithm that takes a simple LQCA as input and decides in $O(n^4)$ time whether it is unitary.* □

Corollary 3.3 [49]. *There is an algorithm that takes a simple LQCA as input and decides in $O(n^{4e})$ time whether it is unitary.* □

3.4 Realizabilities of Quantum Computers

Quantum computers do not exist yet and nobody knows whether we can realize quantum computers or not. In this section, we denote some situations on the realizabilities of quantum computers. In order to realize quantum computers, we will have to realize quantum gates and quantum networks (or circuits). Therefore, many researchers have been studying methods and techniques to realize quantum gates.

As mentioned in Sec. 3.3, first, quantum gates and quantum networks were proposed by Feynman [55] and Deutsch [40]. Their gates extended a classical reversible logic gate (i.e., Toffoli gate [119]) to quantum gates. Deutsch also showed the existence of a three-bit universal quantum gate [40] and DiVincenzo showed the existence of a two-bit universal quantum gate [44, 45, 115] (see Eqs. (3.2) and (3.3)). It is not known whether there are universal two-bit ordinary reversible gates or not. Moreover, Barenco showed that the following two-bit quantum gate $A(\phi, \alpha, \theta)$ is universal [6].

$$A(\phi,\alpha,\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\alpha}\cos\theta & -ie^{i(\alpha+\phi)}\sin\theta \\ 0 & 0 & -ie^{i(\alpha+\phi)}\sin\theta & e^{i\alpha}\cos\theta \end{pmatrix},$$

where the computational basis is $\{|0,0\rangle, |0,1\rangle, |1,0\rangle, |1,1\rangle\}$, and ϕ, α, and θ are fixed irrational multiples of π. Deutsch et al. [42] and Lloyd [79] independently showed that almost any two-bit quantum gate is universal and Barenco et al. systematically studied elementary gates for quantum computers [4].

Moreover, some networks that realize algorithms for solving problems (e.g., Shor's factoring algorithm) are also proposed. In [33, 34], Chuang et al. propose networks to solve Deutsch & Jozsa's problem in [41]. For Shor's factoring algorithm and the quantum Fourier transform used in the algorithm, some networks are proposed (e.g., [7, 8, 86]). Networks for more primitive operators such as arithmetic operators are also proposed [123].

Physical systems to realize the quantum gates (and quantum networks) above are also proposed. Mainly, physicists have proposed methods of how to physically realize a quantum (two-bit) controlled-NOT gate. A quantum controlled-NOT gate is

$$|\epsilon_1, \epsilon_2\rangle \rightarrow e^{i\theta}|\epsilon_1 \oplus \epsilon_1, \epsilon_2\rangle,$$

where $\epsilon_1, \epsilon_2 \in \{0, 1\}$ and θ is an irrational multiple of π. This gate is one of two-bit quantum universal gates.

Barenco et al. proposed two physical systems to realize the quantum controlled-NOT gate [5]. One is based on ramsey atomic interferometry and the other is on the selective driving of optical resonances of two subsystems undergoing a dipole-dipole interaction. Sleator and Weinfurter also proposed the quantum controlled-NOT gate that is based on cavity QED (quantum electrodynamics) [114].

The two proposals above are only methods of how to construct one quantum gate. Cirac and Zoller proposed that a set of n cold ions interacting with laser light and moving in a linear trap provides a realistic physical system to implement a quantum computer [36]. The distinctive features of this system are that (i) the system allows the implementation on n-bit quantum gates between any set of (not necessarily neighboring) ions, (ii) decoherences can be negligible during the whole computation, and (iii) the final readout can be performed with unit efficiency. Pellizzari et al. also proposed a physical system for a set of n atoms based on cavity QED [92]. Furthermore, for example, Monroe et al. and Turchette et al. demonstrated the quantum gate in the laboratory [88, 120].

Finally, we describe about a decoherence problem. A *decoherence problem* is as follows: the central obstacle to realize quantum computers is the fragility of macroscopic atomic superpositions with respect to decoherence by coupling to an environment. According to this problem, some physicists insist that we will not be able to realize the quantum computers if we can not control coherent states [74, 75, 121]. Therefore, some researchers estimate the influences of decoherences for solving problems on the quantum computers [26, 32, 57, 62, 90, 95, 97]. Furthermore, many methods to compensate for and suppress quantum noises in realistic systems are also developed (e.g., see [21, 31, 35, 46, 52, 59, 70, 96, 105, 109, 111, 112, 116, 117, 122]).

Chapter 4

Quantum Complexity Classes

4.1 Ordinary Complexity Classes

4.1.1 Complexity Classes Based on TMs

We define typical complexity classes based on ordinary Turing machines. For a more complete overview, the reader is referred to [3, 61, 65, 91].

First, we define a complexity class based on DTMs. We say that the problems in this class can be *efficiently* solved because on ordinary computers we can solve them in *polynomial time* of the input size.

Definition 4.1. *Let P be the set of all languages that can be recognized by DTMs in polynomial time of the input size.*

The following classes are classes of the problems that are believed not to be efficiently solved on ordinary computers.

Definition 4.2. *Let NP be the set of all languages that can be recognized by NDTMs in polynomial time of the input size and PSPACE be the set of all languages that can be recognized by DTMs in polynomial space of the input size.*

It is well-known that the set of all languages that can be recognized by NDTMs in polynomial space of the input size is also PSPACE.

4.1.2 Complexity Classes Based on PTMs

Next, we define a probabilistic Turing machine and complexity classes based on probabilistic Turing machines [3, 58].

Definition 4.3. *A probabilistic Turing machine(PTM) is an NDTM whose ratio of accepting computations is greater than 1/2.*

Definition 4.4. *Let PP be the set of all languages that can be recognized by PTMs in polynomial time of the input size.*

Moreover, we define complexity classes with bounded error probability.

Definition 4.5. *The error probability is the ratio of computations giving the wrong answers to the total number of computations on input x and is defined by the real-valued function $e(x)$.*

The error probability is the ratio of accepting computations on probabilistically rejected inputs, and the ratio of rejecting computations on probabilistically accepted inputs.

Definition 4.6. *Let BPP be the set of all languages that can be recognized by PTMs in polynomial time of the input size with error probability $e(x) < 1/3$. Let R be the set of all languages that can be recognized by PTMs in polynomial time of the input size with error probability $e(x) < 1/3$ for inputs in the language and with error probability $e(x) = 0$ for inputs not in the language.*

Furthermore, we define a 3-output probabilistic Turing machine and a complexity class based on 3-output probabilistic Turing machines.

Definition 4.7. *A 3-output probabilistic Turing machine (3-output PTM) is a PTM that has three final states, ACCEPT, REJECT, and UNKNOWN. For the 3-output PTM, the acceptance is defined by the same as that for the PTM, i.e., more than half of the computations halt in the ACCEPT final state. The error probability $e_3(x)$ of the 3-output PTM is the probability of halting in the REJECT final state on an accepted input, and the probability of halting in the ACCEPT final state on a rejected input.*

Definition 4.8. *Let ZPP be the set of all languages that can be recognized by 3-output PTMs in polynomial time of the input size with error probability $e_3(x) = 0$.*

Given a class $\mathcal{C}$, the complexity class co-$\mathcal{C}$ is defined by the class of sets $L \subseteq \Sigma^*$ whose complements $\bar{L} = \Sigma^* - L$ are in $\mathcal{C}$. For the classes mentioned above, every class $\mathcal{C}$ satisfies that $\mathcal{C} =$ co-$\mathcal{C}$ except for R and NP, e.g., P = co-P and BPP = co-BPP, however, it is not known whether NP = co-NP or not. Then, the following relationships between complexity classes are shown, however, nobody knows whether P is a proper subset of PSPACE or not.

Theorem 4.1 [58].

1. *P $\subseteq$ ZPP $\subseteq$ R $\subseteq$ NP $\subseteq$ PP $\subseteq$ PSPACE.*
2. *R $\subseteq$ BPP $\subseteq$ PP.*
3. *ZPP = R $\cap$ co-R.* □

4.1.3 Reducibility and Completeness

Finally, we introduce the concepts of reducibility and completeness. These provide a way of evaluating the difficulty of solving two different problems and we can formalize the concept of the most difficult elements of a given class.

Definition 4.9. *Given two sets L_1 and L_2, L_1 is polynomial time many-one reducible to L_2 if and only if there is a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ such that $x \in L_1$ if and only if $f(x) \in L_2$ holds for all $x \in \Sigma^*$.*

We denote that L_1 is reducible to L_2 by $L_1 \leq_m L_2$ and call this reduction *m-reducible.*

Definition 4.10. *Given a class $\mathcal{C}$, a set L_1 is $\mathcal{C}$-hard(or m-hard for $\mathcal{C}$) if and only if for any set L_2 in $\mathcal{C}$, $L_2 \leq_m L_1$, and a set L_1 is $\mathcal{C}$-complete(or m-complete for $\mathcal{C}$) if and only if L_1 is $\mathcal{C}$-hard and $L_1 \in \mathcal{C}$.*

Then, the following theorems are immediately obtained from the definitions (e.g., see [3]).

Theorem 4.2.

1. *$L_1 \leq_m L_2$ iff $\bar{L}_1 \leq_m \bar{L}_2$.*

2. *L is NP-hard (or NP-complete) iff $\bar{L}$ is co-NP hard (or co-NP-complete).*

3. *if L is NP-complete, $L \in$ co-NP iff NP=co-NP.* □

It is well-known that the satisfiability problem of propositional logics (SAT) is NP-complete. SAT is to determine whether a given Boolean formula is satisfiable. Here, a *Boolean formula* is a formula composed of variables, parentheses, and operators $\wedge$ (AND), $\vee$ (OR) and $\neg$ (NOT). A Boolean formula is said to be *satisfiable* if there is an assignment of 0's and 1's to the variables that gives the formula the value 1. Moreover, a lot of problems are known as NP-complete problems [56].

Theorem 4.3. *If $L_1 \leq_m L_2$ and L_2 has a polynomial time algorithm, then so does L_1.* □

By this theorem, if one of NP-complete problems has a polynomial time algorithm, then P=NP, however, nobody knows whether NP-complete problems are included in P. Many computer scientists believe that NP $\neq$ P. The P=NP? problem is one of the most important problems in theoretical computer science.

4.2 Quantum Complexity Classes and Relationships Between Classes

We can regard QTMs as a kind of PTMs because in quantum physics we can obtain results only probabilistically. Therefore, we can also define complexity classes based on QTMs as classes corresponding to the probabilistic complexity classes PP, BPP, R, and ZPP. We call the complexity classes based on QTMs *quantum complexity classes* [23].

4.2.1 BQP

First, we redefine the most familiar quantum complexity class BQP introduced in [23], which corresponds to the class BPP based on PTMs, as a more specific form including the number of measurements during computation. In order to solve some problems efficiently, measurements (i.e., observations of the physical systems constructing QTMs) were effectively used on QTMs. Thus, the measurements seem to be effective in several cases on quantum computation.

Definition 4.11. *Let BQP(k) (Bounded-error Quantum Polynomial time) be the set of all languages that can be recognized by QTMs, whose number of interruptions for measurements during computation is at most k times (k is a non-negative integer), in polynomial time of the size of input x with error probability $e(x) < 1/3$.*

Here, we make the following counting rules for the number k of interruptions in measurements:

1. The last measurements executed to obtain results are not added in k because we must obtain the last results on ordinary Turing machines also.

2. The number of measurements must be added in time complexity (i.e., if the number of measurements is exponential, the problem is not in BQP).

In quantum physics it is widely supported that when the state measured during computation is a *superposition* of several basic states(configurations), we cannot preserve the superposition because of uncertainty principle. Namely, after we executed measurements during computation, we cannot control the computation by using the measured values. On the other hand, measurements on quantum computation have been effectively used to solve problems [41]. Moreover, we may use measurements during computation to decrease the number of configurations superposed and it seems to make faster algorithms by this procedure. In the following theorem, however, we show that the number of measurements during computation do not extend the complexity class [83].

Theorem 4.4 [83]. *BQP(k+1) = BQP(k), where k is a non-negative integer.*

Proof. Obviously, BQP(k) $\subseteq$ BQP(k+1). Then, we prove that BQP(k+1) $\subseteq$ BQP(k). Let the set $\{|a,b,c\rangle\}$ for registers $|a\rangle$,$|b\rangle$, and $|c\rangle$ be a normalized computational basis, i.e., $\langle a_1, b_1, c_1 | a_2, b_2, c_2\rangle = \delta_{a_1,a_2}\delta_{b_1,b_2}\delta_{c_1,c_2}$, where $\delta_{i,j} = 1$ if $i = j$, otherwise $\delta_{i,j} = 0$. Moreover, we call $|a\rangle$(resp. $|b\rangle$, $|c\rangle$) Reg1(resp. Reg2, Reg3).

First, without loss of generality, let us consider the following superposition during computation.

$$\psi_1 = \sum_{k=1}^{m}\sum_{i=1}^{l_k} a_{ik}^{(0)} |c_{ik}, d_k, 0\rangle,$$

where $\sum_{k=1}^{m}\sum_{i=1}^{l_k}\left|a_{ik}^{(0)}\right|^2 = 1$. Now, we measure Reg2. When a value d_K for Reg2 is measured, ψ_1 becomes as follows (even if other one is measured, we can take the same procedure).

$$\psi_1 \to \psi_2 = \sum_{i=1}^{l_K} a_{iK}^{(1)}|c_{iK}, d_K, 0\rangle,$$

where $a_{iK}^{(0)}$ is renormalized to

$$a_{iK}^{(1)} = \frac{a_{iK}^{(0)}}{\left(\sum_{i=1}^{l_K}\left|a_{iK}^{(0)}\right|^2\right)^{1/2}}$$

and $\sum_{i=1}^{l_K}\left|a_{iK}^{(1)}\right|^2 = 1$. The probability to measure the value d_K, $\mathrm{Pr}_2^{(1)}(\mathrm{Reg2}{=}d_K)$, is

$$\mathrm{Pr}_2^{(1)}(\mathrm{Reg2}{=}d_K) = \sum_{i=1}^{l_K}\left|a_{iK}^{(0)}\right|^2.$$

After the next computation, in general, ψ_2 will become ψ_3.

$$\psi_2 \to \psi_3 = \sum_{i=1}^{l'_K}\sum_{j=1}^{n} a_{ijK}^{(2)}|c'_{iK}, d'_{jK}, 0\rangle,$$

where

$$\sum_{i=1}^{l'_K}\sum_{j=1}^{n}\left|a_{ijK}^{(2)}\right|^2 = \sum_{i=1}^{l_K}\left|a_{iK}^{(1)}\right|^2 = 1$$

because of unitary transforms. Here, let us consider measuring Reg1 and Reg2 to obtain the results. When we measure the values c'_{IK} and d'_{JK} for Reg1 and Reg2, respectively, the probability to obtain them, $\mathrm{Pr}_{1,2}^{(1)}(\mathrm{Reg1}{=}c'_{IK},\mathrm{Reg2}{=}d'_{JK})$, is

$$\mathrm{Pr}_{1,2}^{(1)}(\mathrm{Reg1}{=}c'_{IK},\mathrm{Reg2}{=}d'_{JK}) = \left|a_{IJK}^{(2)}\right|^2$$

and ψ_3 becomes ψ_4.

$$\psi_3 \to \psi_4 = |c'_{IK}, d'_{JK}, 0\rangle.$$

Instead of the first measurement, let us consider copying Reg2 to Reg3. Here, we must copy the qubits needed to measure Reg2 of the first measurement and to avoid interferences among configurations occurring because of no execution of the measurements. Obviously, this copy can be executed in polynomial time because of executing the measurements in polynomial time. Then, ψ_1 becomes ψ'_2.

$$\psi_1 \to \psi'_2 = \sum_{k=1}^{m} \sum_{i=1}^{l_k} a_{ik}^{(0)} |c_{ik}, d_k, d_k^*\rangle.$$

The value d_k^* has more information than d_k. After that, in order to obtain the same results above, this computation will become as follows, where the computation must be executed to preserve Reg3.

$$\psi'_2 \to \psi'_3 = \sum_{k=1}^{m} \sum_{i=1}^{l'_k} \sum_{j=1}^{n} a_{ijk}^{(2')} |c'_{ik}, d'_{jk}, d_k^*\rangle.$$

Note that

$$\sum_{i=1}^{l_k} \left| a_{ik}^{(0)} \right|^2 = \sum_{i=1}^{l'_k} \sum_{j=1}^{n} \left| a_{ijk}^{(2')} \right|^2$$

and

$$a_{ijk}^{(2)} = \frac{a_{ijk}^{(2')}}{\left(\sum_{i=1}^{l'_k} \sum_{j=1}^{n} \left| a_{ijk}^{(2')} \right|^2 \right)^{1/2}}$$

for $k = 1, 2, \ldots, m$. When we need to obtain the values of Reg1 and Reg2, we must measure Reg3 first. The probability to measure the value d_K^* for Reg3 (the value d_K^* corresponds to the value d_K of the first case), $\mathrm{Pr}_3^{(2)}(\mathrm{Reg3}{=}d_K^*)$, is

$$\mathrm{Pr}_3^{(2)}(\mathrm{Reg3}{=}d_K^*) = \sum_{i=1}^{l'_K} \sum_{j=1}^{n} \left| a_{ijK}^{(2')} \right|^2 = \sum_{i=1}^{l_K} \left| a_{iK}^{(0)} \right|^2 .$$

This is the same probability as d_K of the first case, i.e., $\mathrm{Pr}_3^{(2)}(\mathrm{Reg3}{=}d_K^*) = \mathrm{Pr}_2^{(1)}(\mathrm{Reg2}{=}d_K)$. Moreover, when we measure the values c'_{IK} and d'_{JK} for Reg1 and Reg2, respectively, in order to obtain the results, the probability to obtain them after measuring Reg3, $\mathrm{Pr}_{1,2}^{(2)}(\mathrm{Reg1}{=}c'_{IK},\mathrm{Reg2}{=}d'_{JK})$, is

$$\mathrm{Pr}_{1,2}^{(2)}(\mathrm{Reg1}{=}c'_{IK},\mathrm{Reg2}{=}d'_{JK}) = \frac{\left| a_{IJK}^{(2')} \right|^2}{\sum_{i=1}^{l'_K} \sum_{j=1}^{n} \left| a_{IJK}^{(2')} \right|^2} = \left| a_{IJK}^{(2)} \right|^2$$

and ψ'_3 will become ψ'_4 with the same probability as ψ_4 of the first case above, i.e., $\mathrm{Pr}_{1,2}^{(2)}(\mathrm{Reg1}{=}c'_{IK},\mathrm{Reg2}{=}d'_{JK})$ $=\mathrm{Pr}_{1,2}^{(1)}(\mathrm{Reg1}{=}c'_{IK},\mathrm{Reg2}{=}d'_{JK})$,

$$\psi'_3 \to \psi'_4 = |c'_{IK}, d'_{JK}, d_K^*\rangle.$$

Further, from ψ_4', we can obtain the same results with the same probability as ψ_4 except for the value d_K^* of Reg3. The executing time of the second case (i.e., the time obtaining ψ_4') is the sum of the executing time of the first case (i.e., the time obtaining ψ_4) and the time copying Reg2 to Reg3, and the copying time can be executed in polynomial time. Thus, we can reduce one interruption for measurements during computation. $\square$

In [23], Bernstein and Vazirani defined a quantum complexity class BQP with bounded error probability.

Definition 4.12. *Let BQP be the set of all languages that can be recognized by QTMs in polynomial time of the size of input x with error probability $e(x) < 1/3$.*

From Theorem 4.4, the following result is immediately obtained.

Corollary 4.1 [83]. *BQP = BQP(0).* $\square$

4.2.2 Other Quantum Complexity Classes

Furthermore, we can also define other quantum complexity classes PQP, QR, ZQP, and EQP corresponding to PP, R, ZPP, and P, respectively.

Definition 4.13. *Let PQP (Probabilistic Quantum Polynomial time) be the set of all languages that can be recognized by QTMs in polynomial time of the input size with error probability $e(x) < 1/2$.*

The following quantum complexity classes are classes with bounded error probability.

Definition 4.14. *Let QR (Quantum Random polynomial time) be the set of all languages that can be recognized by QTMs in polynomial time of the input size with error probability $e(x) < 1/3$ for inputs in the language and with error probability $e(x) = 0$ for inputs not in the language.*

Definition 4.15. *Let EQP (Exact or Error-free Quantum Polynomial time) be the set of all languages that can be recognized by QTMs in polynomial time of the input size with error probability $e(x) = 0$.*

Finally, we define a 3-output quantum Turing machine and a complexity class based on 3-output quantum Turing machines.

Definition 4.16. *A 3-output quantum Turing machine(3-output QTM) is a QTM that has three final states, ACCEPT, REJECT, and UNKNOWN. For the 3-output QTM, the acceptance is that more than half the computations halt in the ACCEPT final state. The error probability $e_3(x)$ of the 3-output QTM is the probability of halting in the REJECT final state on an accepted input, and the probability of halting in the ACCEPT final state on a rejected input.*

Definition 4.17. *Let ZQP (Zero-error Quantum Polynomial time) be the set of all languages that can be recognized by 3-output QTMs in polynomial time of the input size with error probability $e_3(x) = 0$.*

4.2.3 Relationships Between Complexity Classes

Bernstein and Vazirani showed that BQP $\subseteq$ PSPACE [23]. Since we can show that PQP $\subseteq$ PSPACE by the same way, there are the following relationships between complexity classes.

Theorem 4.5.

1. *P $\subseteq$ EQP $\subseteq$ ZQP $\subseteq$ QR $\subseteq$ BQP $\subseteq$ PQP $\subseteq$ PSPACE.*
2. *PP $\subseteq$ PQP.*
3. *BPP $\subseteq$ BQP.*
4. *R $\subseteq$ QR.*
5. *ZPP $\subseteq$ ZQP.*

Proof. We prove only that PQP $\subseteq$ PSPACE, because other relations are trivial from the definitions.

Let us consider any problem $x \in$ PQP, where the size of x is n. First, we show that the number of configurations needed to solve x on a QTM is at most $2^{n^{O(1)}}$. In one step on the QTM, any configuration can be evolved into a superposition of at most two configurations by Eq. (3.1). Therefore, the total number of configurations needed to solve x is at most $2^{n^{O(1)}}$ even if all the configurations differ during the computation (i.e., even if there is no interference between configurations during the computation), because x can be solved in polynomial time on the QTM.

Next, let us consider an influence that occurs for neglecting interferences on the QTM. Without loss of generality, let

$$|\psi\rangle = \sum_i \alpha_i |a_i\rangle$$

be a superposition of configurations during the computation, where $|a_i\rangle \neq |a_j\rangle$ if and only if $i \neq j$. In general, each configuration $|a_i\rangle$ evolves as follows:

$$|a_i\rangle \to \sum_j \beta_{ij} |b_j\rangle. \tag{4.1}$$

Therefore,

$$|\psi\rangle \to \sum_j (\sum_i \alpha_i \beta_{ij}) |b_j\rangle.$$

Then, the probability $\Pr_1(b_j)$ to obtain a configuration $|b_j\rangle$ is

$$\Pr_1(b_j) = \left| \sum_i \alpha_i \beta_{ij} \right|^2 .$$

Instead of Eq. (4.1), let us consider an evolution in which all configurations differ in every step (and this corresponds to the neglect of interferences), i.e.,

$$|a_i\rangle \to \sum_j \beta_{ij}|b_j^{(i)}\rangle,$$

where $|b_j^{(i)}\rangle \neq |b_l^{(k)}\rangle$ if and only if $i \neq k$ or $j \neq l$. Then,

$$|\psi\rangle \to \sum_j \sum_i \alpha_i\beta_{ij}|b_j^{(i)}\rangle.$$

The probability $\Pr(b_j^{(i)})$ to obtain a configuration $|b_j^{(i)}\rangle$ is

$$\Pr(b_j^{(i)}) = |\alpha_i\beta_{ij}|^2$$

and the probability $\Pr_2(b_j)$ to obtain either of configurations $|b_j^{(i)}\rangle$ for all i is

$$\Pr{}_2(b_j) = \sum_i \Pr(b_j^{(i)}) = \sum_i |\alpha_i\beta_{ij}|^2.$$

In general, $\Pr_1(b_j) \neq \Pr_2(b_j)$. However, since the sum $\mathrm{Amp}(b_j)$ of the amplitudes corresponding to the configurations $|b_j^{(i)}\rangle$ for all i is $\mathrm{Amp}(b_j) = \sum_i \alpha_i\beta_{ij}$,

$$|\mathrm{Amp}(b_j)|^2 = \left|\sum_i \alpha_i\beta_{ij}\right|^2 = \Pr{}_1(b_j).$$

Finally, we show that in order to solve the $x \in$ PQP, a DTM can simulate the superposition of the configurations made by the QTM in polynomial space. First, let a triple (α, a, i) correspond to a configuration $\alpha|a\rangle$, where let i be an identity corresponding to the configuration a(and an integer). In order to solve x, the DTM constructs two sets, V and E, in the following way. Initially, $V = \emptyset$, $E = \emptyset$, and $i = 0$. When a configuration $\alpha|a\rangle$ is made during the computation on the QTM and i is the maximum number of the identity corresponding to a in V,

$$V \leftarrow V \cup \{(\alpha, a, i+1)\}.$$

Moreover, when a configuration $\alpha|a\rangle$ evolves into $\beta_1|b_1\rangle + \beta_2|b_2\rangle$ in one step on the QTM,

$$E \leftarrow E \cup \{((\alpha, a, i), (\beta_1, b_1, j)), ((\alpha, a, i), (\beta_2, b_2, k))\}.$$

For V and E that are finally constructed when the computation halts, let us consider a rooted direct tree $G = (V, E)$. The width of G is $2^{n^{O(1)}}$ and the depth of G is $n^{O(1)}$. Namely, the total number of paths from the root needed to solve x is $2^{n^{O(1)}}$. Therefore, the DTM needs exponential time and exponential space to construct the tree G, however, it merely needs to compute the sum of the amplitudes corresponding to accepting(or rejecting) configurations in order to solve x. This can be done in polynomial space on the DTM. Thus, the DTM can compute the accepting(or the rejecting) probability of x in polynomial space. Then, $x \in$ PSPACE. $\square$

$$\begin{tikzcd}
 & & & \mathrm{NP} \arrow[dr] & & & \\
 & \mathrm{ZPP} \arrow[r] \arrow[dr] & \mathrm{R} \arrow[ur] \arrow[r] \arrow[dr] & \mathrm{BPP} \arrow[r] \arrow[dr] & \mathrm{PP} \arrow[dr] & & \\
\mathrm{P} \arrow[ur] \arrow[r] & \mathrm{EQP} \arrow[r] & \mathrm{ZQP} \arrow[r] & \mathrm{QR} \arrow[r] & \mathrm{BQP} \arrow[r] & \mathrm{PQP} \arrow[r] & \mathrm{PSPACE}
\end{tikzcd}$$

Figure 4.1: Relationships between complexity classes.

In Fig. 4.1, we show the relationships between complexity classes (e.g., BPP $\rightarrow$ BQP means that BPP $\subseteq$ BQP). None of the inclusions in Theorem 4.5 is known to be proper.

Chapter 5

Periodic Functions

5.1 Introduction

Many public key cryptosystems such as RSA public key cryptosystem have been proposed and some of them are based on the difficulties of finding discrete logarithms and factoring integers. No polynomial time algorithm to find them on ordinary computers is known. Moreover, some of provably secure pseudo-random bit sequence generators introduced by Blum and Micali [27] are also based on them. In 1994, however, Shor showed that a QTM can find discrete logarithms and factor integers in polynomial time with bounded error probability [108, 110](also see [53]). It means that if we can realize the QTM, today's many public key cryptosystems will not be secure.

In this chapter, we show that the periods in some kinds of periodic functions (e.g., a cycle problem below) can be found in polynomial time with bounded error probability on a QTM. Some of the functions proposed as pseudo-random generators are also included in these functions.

A cycle problem is as follows: let us consider a function $f : D \to D$ such that it has $s + r(= n)$ distinct values $x_0 = f^0(x_0), f^1(x_0), \ldots, f^{s+r-1}(x_0)$, but $f^s(x_0) = f^{s+r}(x_0)$ (throughout this chapter, let D be a finite domain). This means that $f^j(x_0) = f^{j+r}(x_0)$ for all $j \geq s$. A *cycle problem* for f and x_0 is to find the pair (s, r). In [106], Sedgewick et al. showed that $n\left(1 + \Theta\left(1/M^{1/2}\right)\right)$ time is both necessary and sufficient to solve the problem on ordinary computers if M memory cells are available to store the values of the function. It takes $O(n)$ time, however, we show that a QTM can solve it in $(\log_2 n)^{O(1)}$ time.

5.2 Quantum Fourier Transforms

5.2.1 Shor's Quantum Algorithm for Factoring Integers

First, we briefly summarize Shor's quantum algorithm for factoring integers [108, 110]. Let us consider the factorization of an integer N. In essential, his algorithm is to find the least integer r in polynomial time of the input size (i.e., in $(\log_2 N)^{O(1)}$ time) with bounded error probability such that $x^r \equiv 1 (\mathrm{mod}\ N)$, choosing a random x. When r is even and $x^{r/2} \not\equiv \pm 1 (\mathrm{mod}\ N)$, we will succeed in factoring N if $1 < \gcd(x^{r/2} \pm 1, N) < N$, where $\gcd(m,n)$ is the greatest common divisor of m and n. In the following, we show the algorithm to find r.

We find q, the power of 2 with $N^2 \le q < 2N^2$ (in general, q may be a smooth number instead of the power of 2, where q is called *smooth* if all prime factors of q are bounded by $(\log_2 q)^{O(1)}$). First, a QTM makes the following configuration.

$$|0,0\rangle \to \frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, 0\rangle.$$

We call s_1(resp. s_2) for a configuration $|s_1, s_2\rangle$ the first (resp. second) register. Next, given a random $x (0 < x \le N-1)$, the QTM computes $x^a (\mathrm{mod}\ N)$ for $a = 0, \ldots, q-1$ in quantum parallel computation and puts the value in the second register.

$$\frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, x^a (\mathrm{mod}\ N)\rangle.$$

Here, the QTM performs a quantum Fourier transform A_q,

$$|a\rangle \overset{A_q}{\to} \frac{1}{q^{1/2}} \sum_{c=0}^{q-1} e^{2\pi i ac/q} |c\rangle.$$

When the QTM applies this transform on the first register,

$$\frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, x^a (\mathrm{mod}\ N)\rangle \overset{A_q}{\to} \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} e^{2\pi i ac/q} |c, x^a (\mathrm{mod}\ N)\rangle.$$

Finally, we measure the first and second registers. The probability $\Pr(c)$ obtaining a configuration $|c, x^k (\mathrm{mod}\ N)\rangle$ is

$$\Pr(c) = \left| \frac{1}{q} \sum_{b=0}^{\lfloor \frac{q-k-1}{r} \rfloor} e^{2\pi i (br+k)c/q} \right|^2,$$

because the sum is over all a satisfying $a \equiv k (\mathrm{mod}\ r)$, where r is the order of x.

$$\begin{aligned} \Pr(c) &= \left| \frac{1}{q} \sum_{b=0}^{\lfloor \frac{q-k-1}{r} \rfloor} e^{2\pi i (br+k)c/q} \right|^2 \\ &= \frac{1}{q^2} \left| \sum_{b=0}^{\lfloor \frac{q-k-1}{r} \rfloor} e^{2\pi i b \{rc\}_q / q} \right|^2, \end{aligned}$$

where $\{rc\}_q$ is the residue that is congruent to $rc (\mathrm{mod}\ q)$ and is in the range $-q/2 < \{rc\}_q \le q/2$. Shor showed that for $-r/2 \le \{rc\}_q \le r/2$,

$$\Pr(c) = \Omega\left(\frac{1}{r^2}\right).$$

When $-r/2 \le \{rc\}_q \le r/2$, there is a d such that $-r/2 \le rc - dq \le r/2$, i.e.,

$$\left|\frac{c}{q} - \frac{d}{r}\right| \le \frac{1}{2q}.$$

Since we know c and q, we can efficiently find r that satisfies the inequality above if d is relatively prime to r. The number of configurations $|c, x^k (\mathrm{mod}\ N)\rangle$ that enable us to compute r in this way is $r\phi(r)$, where $\phi(r)$ is Euler's totient function. Then, we obtain r with probability $\Omega(\phi(r)/r) = \Omega(1/\log_2 \log_2 r)$, because each of these configurations occurs with probability $\Omega(1/r^2)$. Therefore, we will succeed in factoring N if r is even and $1 < \gcd(x^{r/2} \pm 1, N) < N$.

5.2.2 Various Transforms

Shor's quantum algorithm mentioned above efficiently uses the quantum Fourier transform, i.e., $e^{2\pi iab/q}$ is used as the amplitude of configurations. Instead of $e^{2\pi iab/q}$, we show that functions satisfying some properties may be used as the amplitude. First, we consider a function $C(a, b, q) : \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}^+ \to \mathbf{C}$ that satisfies the following conditions:

$$\left.\begin{array}{l} C(a,b,q) = C(b,a,q), \\ C(a \circ c, b, q) = C(a,b,q)C(c,b,q), \\ C(0,b,q) = 1, \end{array}\right\} \tag{5.1}$$

and

$$\sum_{b=0}^{q-1} C^*(a,b,q)C(b,c,q) = \left\{\begin{array}{ll} q & \text{if } a = c, \\ 0 & \text{otherwise,} \end{array}\right. \tag{5.2}$$

where $\circ$ is an operator and $C^*(a,b,q)$ is the complex conjugate of $C(a,b,q)$. Furthermore, we define a transform C_q by

$$|a\rangle \overset{C_q}{\to} \frac{1}{q^{1/2}} \sum_{b=0}^{q-1} C(a,b,q)|b\rangle. \tag{5.3}$$

QTMs can execute this transform because the matrix U_{C_q} constructed with $\frac{1}{q^{1/2}}C(a,b,q)$, the ath row and bth column element, is a unitary matrix from Eq. (5.2) combining with the first condition of Eq. (5.1), where the rows and columns will be indexed by beginning with 0 unless otherwise specified. Now, we suppose that this transform can be executed in $(\log_2 q)^{O(1)}$ time.

In the following, we consider two operators, $\oplus$ and $+$, as the operator $\circ$.

1. Let the operator $\circ$ be $\oplus$(bitwise exclusive-or). Then, the second condition of Eq. (5.1) is

$$C(a \oplus c, b, q) = C(a, b, q)C(c, b, q).$$

Now, let us consider Simon's problem in [113]: suppose that we are given a function $f : \{0,1\}^n \to \{0,1\}^m$ with $m \geq n$, and that we are promised that either f is one-to-one or there is a non-trivial s such that $\forall x \neq x'(f(x) = f(x') \Leftrightarrow x' = x \oplus s)$. We wish to determine which of these conditions holds for f.

In order to solve this problem, first, a QTM executes the following procedure (where $q = 2^n$).

$$\begin{aligned} |0,0\rangle &\to \frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a,0\rangle \\ &\to \frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, f(a)\rangle. \end{aligned}$$

Here, the QTM performs the transform C_q on the first register.

$$\frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, f(a)\rangle \overset{C_q}{\to} \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} C(a,c,q)|c, f(a)\rangle.$$

If f is not one-to-one, configurations $|c, f(a)\rangle$ and $|c, f(a \oplus s)\rangle$ are identical for each c and a, and the amplitude $\alpha(a,c)$ is

$$\alpha(a,c) = \frac{1}{q}\left(C(a,c,q) + C(a \oplus s, c, q)\right) = \frac{1}{q}C(a,c,q)(1 + C(s,c,q)).$$

For instance, let $C(a,b,q) = (-1)^{ab}$ and q be the power of 2. Obviously, $(-1)^{ab}$ satisfies Eqs. (5.1) and (5.2). Then, the algorithm above becomes Simon's algorithm in [113], and $\alpha(a,c) = \pm\frac{2}{q}$ if $sc \equiv 0 (\text{mod } 2)$, otherwise $\alpha(a,c) = 0$.

Moreover, let $C(a,b,q)$ be a discrete Walsh function (e.g., the Walsh-Paley function) [102]. Then, we can obtain the same result as using Simon's algorithm.

First, we define a function $r(x) : \mathbf{R} \to \{-1, 1\}$ by

$$r(x) = \begin{cases} 1 & \text{if } 0 \leq x < 1/2, \\ -1 & \text{if } 1/2 \leq x < 1, \end{cases}$$

and

$$r(x+1) = r(x).$$

Moreover, let

$$r_l(x) = r(2^l x),$$

where $x \in \mathbf{R}$ and $l \in \mathbf{N}$. Given $l \in \mathbf{N}$, we denote l as

$$l = \sum_{k=0}^{\infty} l_k 2^k,$$

where $l_k = \{0, 1\}$. The *Walsh-Paley function* $W_l(x)$ is defined by

$$W_l(x) = \prod_{k=0}^{\infty} r_k(x)^{l_k}.$$

Note that this product is always finite because $l_k = 0$ for k that is sufficiently large and the values of the Walsh-Paley function are only 1 and -1. Furthermore, the *discrete Walsh-Paley function* is defined by $W_a\left(\frac{b}{q}\right)$ for $a \in \mathbf{N}$, $b \in \mathbf{N}$ and $q \in \mathbf{Z}^+$. Now, let q be the power of 2, i.e., $q = 2^n$ for $n \in \mathbf{N}$. Then, the following properties are known (therefore, this function satisfies Eqs. (5.1) and (5.2)).

$$\begin{aligned} W_a\left(\frac{b}{q}\right) &= W_b\left(\frac{a}{q}\right), \\ W_{a\oplus c}\left(\frac{b}{q}\right) &= W_a\left(\frac{b}{q}\right) W_c\left(\frac{b}{q}\right), \\ W_0\left(\frac{b}{q}\right) &= 1, \end{aligned}$$

and

$$\sum_{b=0}^{q-1} W_a\left(\frac{b}{q}\right) W_c\left(\frac{b}{q}\right) = \begin{cases} q & \text{if } a = c, \\ 0 & \text{otherwise.} \end{cases}$$

Namely, the matrix U_{W_q} constructed with $\frac{1}{q^{1/2}} W_a\left(\frac{b}{q}\right)$, the ath row and bth column element, is a unitary matrix. Therefore, we only need to show that the matrix U_{W_q} can be computed in $n^{O(1)}$ time to solve Simon's problem.

We show that the following transform can be computed in $O(n)$ time on the QTM.

$$|a\rangle \overset{U_{W_q}}{\rightarrow} \frac{1}{q^{1/2}} \sum_{c=0}^{q-1} W_a\left(\frac{c}{q}\right) |c\rangle.$$

Let H_q be the q-dimensional Hadamard matrix, i.e., let

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

then,

$$H_q = H_2 \otimes H_{q/2} = \begin{pmatrix} H_{q/2} & H_{q/2} \\ H_{q/2} & -H_{q/2} \end{pmatrix}.$$

The matrix H_q can be computed in $O(n)$ time. Next, let H_{kj} be the kth row and jth column element of H_q, then,

$$H_{kj} = W_{b(k)}\left(\frac{j}{q}\right),$$

where $b(k) = \sum_{i=0}^{n-1} k_{n-i-1}2^i$ when $k = \sum_{i=0}^{n-1} k_i 2^i$ for $k_i \in \{0,1\}$. The map $k \to b(k)$ is called a *bit-reversal map.* We define B_q as a q-dimensional bit-reversal matrix such that

$$|k_{n-1}, k_{n-2}, \ldots, k_1, k_0\rangle \xrightarrow{B_q} |k_0, k_1, \ldots, k_{n-2}, k_{n-1}\rangle.$$

Obviously B_q is a unitary matrix and can be computed in $O(n)$ time. Then,

$$U_{W_q} = \frac{1}{q^{1/2}} H_q B_q.$$

Thus, U_{W_q} can be computed in $O(n)$ time on the QTM.

2. Let the operator $\circ$ be $+$. Then, the second condition of Eq. (5.1) is

$$C(a+c, b, q) = C(a, b, q)C(c, b, q).$$

Moreover, let

$$\left.\begin{array}{l} C(a, b+q, q) = C(a, b, q), \\ C(ac, b, q) = C(a, cb, q). \end{array}\right\} \qquad (5.4)$$

When $C(a, b, q) = e^{2\pi i ab/q}$, it satisfies Eqs. (5.1), (5.2), and (5.4), and the transform of Eq. (5.3) is the quantum Fourier transform.

Now, let us consider factoring a number N by using $C(a, b, q)$. We find q, the power of 2 with $N^2 \le q < 2N^2$. First, a QTM makes the following configuration.

$$|0, 0\rangle \to \frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, 0\rangle.$$

Next, given a random $x (0 < x \leq N-1)$, the QTM computes $x^a (\bmod N)$ for $a = 0, \ldots, q-1$ in quantum parallel computation and puts the value in the second register.

$$\frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, x^a (\bmod N)\rangle.$$

Here, the QTM performs the transform C_q on the first register.

$$\frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a, x^a (\bmod N)\rangle \xrightarrow{C_q} \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} C(a, c, q) |c, x^a (\bmod N)\rangle.$$

Finally, we measure the first and second registers. The probability $\Pr(c)$ obtaining a configuration $|c, x^k (\bmod N)\rangle$ is

$$\Pr(c) = \left| \frac{1}{q} \sum_{b=0}^{\left\lfloor \frac{q-k-1}{r} \right\rfloor} C(br+k, c, q) \right|^2,$$

because the sum is over all a satisfying $a \equiv k (\bmod r)$.

$$\begin{aligned} \Pr(c) &= \left| \frac{1}{q} \sum_{b=0}^{\left\lfloor \frac{q-k-1}{r} \right\rfloor} C(br+k, c, q) \right|^2 \\ &= \left| \frac{1}{q} \sum_{b=0}^{\left\lfloor \frac{q-k-1}{r} \right\rfloor} C(br, c, q) C(k, c, q) \right|^2 \\ &= \left| \frac{1}{q} C(k, c, q) \right|^2 \left| \sum_{b=0}^{\left\lfloor \frac{q-k-1}{r} \right\rfloor} C(br, c, q) \right|^2 \\ &= \left| \frac{1}{q} C(k, c, q) \right|^2 \left| \sum_{b=0}^{\left\lfloor \frac{q-k-1}{r} \right\rfloor} C(b, rc, q) \right|^2 \\ &= \left| \frac{1}{q} C(k, c, q) \right|^2 \left| \sum_{b=0}^{\left\lfloor \frac{q-k-1}{r} \right\rfloor} C(b, \{rc\}_q, q) \right|^2, \end{aligned}$$

where $\{rc\}_q$ is in the range $-q/2 < \{rc\}_q \leq q/2$. If

$$\Pr(c) = \Omega\left(\frac{1}{r^2}\right) \qquad \text{for } -d_1 \leq \{rc\}_q \leq d_2, \tag{5.5}$$

where $0 \leq d_1, d_2 \leq r/2$ and $d_2 + d_1 = \Omega(r)$. Then, we can obtain the order r in $(\log_2 q)^{O(1)}$ time with bounded error probability on the QTM.

5.3 Finding the Periods of Periodic Functions

5.3.1 Type $f(x) = f(x + r)$

As shown in the previous section, Shor's algorithm is to find the least r such that $x^r \equiv 1 (\text{mod } N)$. Let $f(a) = x^a (\text{mod } N)$. Since this function has a property $f(k) = f(k + r)$, his algorithm means finding the period r of a function f such that $f(k) = f(k+r)$. This observation immediately leads to the following theorem [84].

Theorem 5.1 [84]. *Let $f : \mathbf{N} \to D$ be a polynomial computable function such that $f(k) = f(k + r)$, where the function f is one-to-one per one period. Then, a QTM can find r in polynomial time of the input size with bounded error probability.*

Proof. We modify $x^a (\text{mod } N)$ in Shor's algorithm into the given f and execute the algorithm. First, a QTM executes Shor's algorithm with $q := 2$ for the function f, where q is the number of configuration of the superposition in Shor's algorithm. If the QTM can find r, halt. Otherwise, let $q := 2q$, and the QTM executes Shor's algorithm again. This procedure is iterated until we find r. Since the number of iterations is at most $O(\log_2 r)$ times and f can be executed in polynomial time, the QTM can find r in polynomial time. $\square$

Furthermore, the function f in Theorem 5.1 may be a combined one with some operators (e.g., a composite of functions).

Corollary 5.1 [84]. *Let f_i for $i = 1, \ldots, n$ be computable functions and F be a polynomial computable function such that*

$$F(f_1(k), f_2(k), \ldots, f_n(k)) = F(f_1(k + r), f_2(k + r), \ldots, f_n(k + r)).$$

Then, a QTM can find r in polynomial time of the input size with bounded error probability. $\square$

Note that each f_i may not be executed in polynomial time and n may not be a polynomial size of the input.

For instance, given a and N, let

$$f_k(x) = \frac{(\log_e a)^k}{k!} x^k (\text{mod } N)$$

for $k = 0, 1, 2, \ldots, \infty$. A QTM cannot compute all values of $f_k(x)$ for every k in polynomial time. However, let us consider a function

$$F(x) = \sum_{k=0}^{\infty} f_k(x).$$

Then, the QTM can compute the function $F(x)$ in polynomial time, because

$$\begin{aligned}
F(x) &= \sum_{k=0}^{\infty} f_k(x) \\
&= \sum_{k=0}^{\infty} \frac{(\log_e a)^k}{k!} x^k (\text{mod } N) \\
&= a^x (\text{mod } N).
\end{aligned}$$

Since the function $F(x)$ is a periodic function, the QTM can find the period of $F(x)$ in polynomial time. Moreover, suppose that data that have a period are given and the number of the data is n. Then, a QTM can find the period in $(\log_2 n)^{O(1)}$ time.

Boneh and Lipton showed that the periods of periodic functions, which are not one-to-one per one period, can be found in polynomial time on a QTM [29]. Here, we define that a function f has *order* at most m if f does not map more than m elements of $\mathbf{Z}_r$ to one, i.e., all $z \in D$ satisfy $|f^{-1}(z)(\text{mod } r)| \leq m$.

Theorem 5.2 [29]. *Suppose that $f : \mathbf{N} \to D$ is a periodic function. Let r be the least period of f and the order of f be at most m. We impose two conditions:*

1. *Let $n = \log_2 r$, then m is at most $n^{O(1)}$.*

2. *Let p be the least prime divisor of q, then $m < p$.*

Then, a QTM can find the period r of the function f in polynomial time with bounded error probability. □

It is not known whether a QTM can find the periods of any periodic function in polynomial time or not, even if the QTM can compute any one value of the function in polynomial time. If a QTM can efficiently find the period of a function, we can show that NP $\subseteq$ BQP.

Theorem 5.3. *Let $f : \mathbf{N} \to D$ be a periodic function that can be computed in polynomial time. Then, the problem that determines whether the least period of the function f is one or not is NP-hard. Therefore, the problem that finds the least period of f is also NP-hard.*

Proof. If there is a polynomial time reduction from an NP-complete problem to this problem, we can prove this. Here, we show that there is a polynomial time reduction from SAT to this problem. First, let $F(x_1, x_2, \ldots, x_n)$ be an n-variable Boolean formula, where $x_i \in \{0, 1\}$ for $i = 1, 2, \ldots, n$, and a binary digit $x_1 x_2 \cdots x_n$ corresponds to each input of F, $x_1, x_2, \ldots, x_n$. For instance, if an input of 4-variable formula F is $0, 1, 0, 1$, the corresponding value is $(0101)_{\text{binary}} = 5$. Moreover, let $g : \{0, 1, \ldots, 2^n - 1\} \to \{0, 1\}$ and $f : \mathbf{N} \to \{0, 1\}$ be functions such that

$$g(x_1 x_2 \cdots x_n) = F(x_1, x_2, \ldots, x_n),$$

and

$$f(x) = g(x(\text{mod } 2^n)).$$

Obviously, the function f is a periodic function of period 2^n and can be computed in polynomial time for each input. Moreover, the least period of f is one if and only if the formula F is tautology or unsatisfiable. When the least period is one, we can verify in polynomial time whether F is either tautology or unsatisfiable. Therefore, if we can find the least period of f, we can verify whether F is satisfiable. □

From this theorem, we obtain the following result immediately.

Corollary 5.2. *Let f be a periodic function that can be computed in polynomial time. If the problem that determines whether the least period of the function f is one or not is included in BQP, NP $\subseteq$ BQP.* □

Moreover, we can show that the following restricted problem of Theorem 5.3 is NP-complete.

Corollary 5.3. *Let f be a function that can be computed in polynomial time. Given a positive integer r, the problem verified that r is not the least period of the function f is NP-complete.*

Proof. First, we show that the problem is in NP, i.e., we find an instance that can be verified that r is not the least period of f in polynomial time. If r is not the least period of f, there is a k such that $k \in \{k_0, k_0 + 1, \ldots, k_0 + r - 1\}$ for some k_0 and $f(k) \neq f(k+r)$. Since we compute f only twice, this can be verified in polynomial time. Next, we show that the problem is NP-hard. From Theorem 5.3, there is a polynomial time reduction from SAT to this problem. Let $r = 1$, and a given Boolean formula of SAT is not tautology (if a formula is tautology, we can verify that it is satisfiable in polynomial time). Then, the formula is satisfiable if and only if $r(= 1)$ is not the least period of f. □

Namely, the conditions in Theorem 5.2 mean that we can verify whether the obtained r is the least period in polynomial time. Then, from this corollary, getting rid of the conditions will be as hard as to solve NP-complete problems.

5.3.2 Type $x = f^r(x)$

Next, given a finite domain D, a function $f : D \to D$, and a seed $x_0 \in D$, let us consider a generation of an infinite sequence $x_0 = f^0(x_0), f^1(x_0), f^2(x_0), \ldots$. Obviously, this sequence becomes cyclic because D is finite. Therefore, for some s and r, we have $s + r$ distinct values $f^0(x_0), f^1(x_0), \ldots, f^{s+r-1}(x_0)$, but $f^s(x_0) = f^{s+r}(x_0)$. This means that $f^j(x_0) = f^{j+r}(x_0)$ for all $j \geq s$. Then, a *cycle problem* for f and x_0 is to find the pair (s, r). Note that the pair (s, r) depends on x_0. From this pair, we can efficiently determine the value of $f^m(x_0)$ for each $m \geq 0$. Let $n = s + r$ and the input size of this problem be $|n|$, i.e., $O(\log_2 n)$. Obviously, the size of D is greater than or equal to n, however, it need not be a polynomial size of n (i.e., it may be an exponential size of n). In [106], Sedgewick et al. showed that $n\left(1+\Theta\left(1/M^{1/2}\right)\right)$ time is both necessary and sufficient to solve the cycle problem on ordinary computers if M memory cells are available to store the values of the function. Their algorithm takes exponential time of the input size. On the other hand, we show that for some kinds of functions, a QTM can solve it in polynomial time with bounded error probability [84].

Given x_0 and N, let $f(x) = x_0 \cdot x (\text{mod } N)$. Shor's algorithm is also to find the least r such that $x_0 = f^r(x_0) \equiv x_0^{r+1} (\text{mod } N)$. Then, from this observation, we obtain the following lemma.

Lemma 5.4 [84]. *Given x_0, let f be a function such that $f^m(x_0)$ can be computed in polynomial time, where m is at most $n^{O(1)}$. Then, a QTM can find r of the cycle problem for f and x_0 in polynomial time with bounded error probability.*

Proof. First, a QTM executes Shor's algorithm with $q := 2$ for the function f, where q is the number of configurations of the superposition in Shor's algorithm. If the QTM can find r, halt. Otherwise, let $q := 2q$, and the QTM executes Shor's algorithm again. This procedure is iterated until the QTM finds r. Since the number of iterations is at most $O(\log_2 n)$ times, the QTM can find r in polynomial time. $\square$

If we know that the size of D is a polynomial size of n, we can find r more efficiently.

Next, given r, we show that a DTM can find s in polynomial time.

Lemma 5.5 [84]. *Given x_0, let f be a function such that $f^m(x_0)$ can be computed in polynomial time, where m is at most $n^{O(1)}$ and the period r is given. Then, a DTM can find s of the cycle problem for f and x_0 in polynomial time.*

Proof. First, let $s' := 0$. If $f^{s'}(x_0) = f^{s'+r}(x_0)$, halt. Otherwise, let $s' := 2s'$, and the DTM checks again whether $f^{s'}(x_0) = f^{s'+r}(x_0)$. This procedure is iterated until the DTM obtains s' such that $f^{s'}(x_0) = f^{s'+r}(x_0)$. The number of iterations is at most $O(\log_2 n)$ times. Since $0 \le s \le s'$ and f has the following conditions:

$$f^{s-1}(x_0) \neq f^{s+r-1}(x_0),$$

and

$$f^{s}(x_0) = f^{s+r}(x_0),$$

The DTM can find s by using binary search in polynomial time. $\square$

Finally, from Lemma 5.4 and Lemma 5.5, a QTM can obtain the following theorem to solve the cycle problem in polynomial time.

Theorem 5.6 [84]. *Given x_0, let f be a function such that $f^m(x_0)$ can be computed in polynomial time, where m is at most $n^{O(1)}$. Then, we can find (s, r) of the cycle problem for f and x_0 in polynomial time with bounded error probability on a QTM.* $\square$

A method for designing provably secure pseudo-random bit sequence generators based on the use of one-way predicates is presented by Blum and Micali [27, 99] as follows. Let $f : D \rightarrow D$, and $B : D \rightarrow \{0, 1\}$ such that

1. given $x = f^{-1}(y)$, it is easy to compute $B(y)$, and
2. given only y, it is difficult to compute $B(y)$.

Given a seed $x_0 \in D$, we can create a sequence $x_0, x_1, \ldots, x_n$ by using $x_{i+1} = f(x_i)$. Moreover, we define $b_i = B(x_{n-i})$ to produce a binary sequence $b_0, b_1, \ldots, b_{n-1}$ of length n. Note that, in general, we must compute $x_0, x_1, \ldots, x_n$ first, and then compute b_0 from x_{n-1}, b_1 from x_{n-2}, and so on.

The generators based on this have been proposed by many researchers (e.g., [2, 27, 28]). However, since we know that a QTM can find discrete logarithms and factor integers in polynomial time, they are not appropriate for pseudo-random bit sequence generators. Moreover, a QTM can also efficiently find the period of some kinds of functions shown in Theorem 5.6. For instance, given a and N, let $f(x) = x^a (\text{mod } N)$. A QTM can compute $f^m(x) = x^{a^m} (\text{mod } N)$ for each m in polynomial time even if $m = O(N)$, because the QTM can factor N in polynomial time. Then, the QTM can find the period of $f(x) = x^a (\text{mod } N)$ in polynomial time from Theorem 5.6. It means that there may be generators that are not secure on QTMs even if they are secure on ordinary computers.

Finally, from Theorem 5.6, we can obtain the following corollary, if the generating function of the recurrence $x_{k+1} = f(x_k)$ is given or it is obtained in polynomial time.

Corollary 5.4 [84]. *Given x_0, let f be a function such that $G(z)$ is the generating function of the recurrence $x_{k+1} = f(x_k)$ for $k = 0, 1, 2, \ldots$, i.e.,*

$$G(z) = \sum_{k=0}^{\infty} x_k z^k = \sum_{k=0}^{\infty} \frac{G^{(k)}(0)}{k!} z^k,$$

where $G^{(k)}(0) = \left. \frac{d^k G(z)}{dz^k} \right|_{z=0}$. Moreover, let $\frac{G^{(k)}(0)}{k!}$ be able to computed in polynomial time for $k \leq n^{O(1)}$. Then, a QTM can find (s, r) of the cycle problem for f and x_0 in polynomial time with bounded error probability.

Proof. Since $f^m(x_0) = \frac{G^{(m)}(0)}{m!}$, a QTM can compute $f^m(x_0)$ for $m \leq n^{O(1)}$ in polynomial time. Then, from Theorem 5.6, the QTM can find (s, r) of the cycle problem for f and x_0 in polynomial time with bounded error probability. $\square$

Chapter 6

Quantum Search Algorithms

6.1 Introduction

Some techniques to solve problems efficiently on QTMs have been proposed. Especially, the most famous technique is a quantum Fourier transform. This is used in Chapter 5 and Shor showed that by this technique, a QTM can factor integers and find discrete logarithms in polynomial time [108, 110].

Another well-known technique is a quantum iterating method. A *quantum iterating method* was proposed by Grover and is a method to increase the probability of accepting states by using an algorithm repeatedly. He showed that for an unsorted table T of n distinct items, $T[0], T[1], \ldots, T[n-1]$, there is a quantum search algorithm that finds the index m of a query item $q(=T[m])$ in expected time $O(n^{1/2})$ with bounded error probability [60]. Ordinary computers need $O(n)$ time to solve it. Moreover, Dürr and Høyer showed that for an unsorted table T of n items, there is a quantum search algorithm that finds the minimum value of T in expected time $O(n^{1/2})$ with bounded error probability [50]. Their algorithm uses Grover's search algorithm as a subroutine.

In this chapter, we show that for an unsorted table T of n items and a query item q (where q may not necessarily exist in T), there is a quantum search algorithm that finds a pair of indices (j, k) corresponding to two successive items, $T[j]$ and $T[k]$, which satisfy that $T[j] \leq q \leq T[k]$, in expected time $O(n^{1/2})$ with bounded error probability. Our algorithm also uses Grover's search algorithm as a subroutine and extends Dürr & Høyer's minimum search algorithm. Moreover, we also show that QTMs can solve some problems in computational geometry more efficiently than ordinary computers.

6.2 Search Algorithms on QTMs

6.2.1 Grover's Quantum Search Algorithm

An unsorted table T contains n distinct items, $T[0], T[1], \ldots, T[n-1]$, of which just one item satisfies a particular property. A problem is to identify that item. Once an item is examined, it is possible to tell whether it satisfies the condition in $O(1)$ time or not.

Ordinary computers need $O(n)$ time to solve the search problem [60], on the other hand, Grover showed that it can be solved in $O(n^{1/2})$ time on a QTM.

Theorem 6.1 [60]. *For an unsorted table T of n distinct items, $T[0], T[1], \ldots, T[n-1]$, there is a quantum search algorithm that finds the index m of a query item $q (= T[m])$ in expected time $O(n^{1/2})$ with bounded error probability.* □

Let q be a query item in the table T and m be the index corresponding to q, i.e., $q = T[m]$. Then, in order to prove Theorem 6.1, he constructed the following algorithm. Here, for any item p, we define $C(p)$ by

$$C(p) = \begin{cases} 1 & \text{if } p = q, \\ 0 & \text{otherwise.} \end{cases}$$

GROVER'S QUANTUM SEARCH ALGORITHM:

1. Initialize the system to

$$\frac{1}{n^{1/2}} \sum_{i=0}^{n-1} |i, 0\rangle.$$

2. Repeat the following procedures $O(n^{1/2})$ times.

   (a) Compute $C(i)$ in quantum parallel, i.e., for each configuration $|i, 0\rangle$,

   $$|i, 0\rangle \to |i, C(T[i])\rangle.$$

   (b) Rotate the phase by π radians if $C(T[m]) = 1$, otherwise leave the configuration unaltered.

   (c) Apply the following transforms FRF, where F is a Fourier transform and R is a rotation matrix:

   $$R_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j = 0, \\ -1 & \text{if } i = j \neq 0, \end{cases}$$

   and

   $$F_{ij} = \frac{1}{2^{n/2}} (-1)^{\bar{i} \cdot \bar{j}},$$

   where $\bar{i}$ is the binary representation of i and $\bar{i} \cdot \bar{j}$ denotes the bitwise dot product of two n bit strings, $\bar{i}$ and $\bar{j}$.

3. Measure the second qubit. Then, the second qubit of the final configuration is $C(T[m]) = 1$(i.e., $T[m] = q$) with a probability of at least $1/2$.

In [30], Boyer et al. analyzed Grover's search algorithm and provided a simple closed-form formula for the probability of accepting states after any given number of iterations of the algorithm. Moreover, they extended Grover's algorithm to multiple solutions, i.e., some of items may be the same.

Theorem 6.2 [30]. *For an unsorted table T of n items, $T[0], T[1], \ldots, T[n-1]$, there is a quantum search algorithm that finds an index corresponding to a query item in expected time $O\left(\left(\frac{n}{t}\right)^{1/2}\right)$ with bounded error probability, where t is the number of items in the table T corresponding to the query item.* □

We call this algorithm an *extended Grover's search algorithm.*

Furthermore, Dürr and Høyer used the extended Grover's search algorithm as a subroutine and showed that for an unsorted table T of n items, there is a quantum search algorithm that finds the minimum value of T in $O(n^{1/2})$ time.

Theorem 6.3 [50]. *For an unsorted table T of n items, there is a quantum search algorithm that finds the minimum value of T in expected time $O(n^{1/2})$ with bounded error probability.* □

6.2.2 Extended Quantum Search Algorithm

In this subsection, let us consider the following problem:
PROBLEM:

> Given an unsorted table T of n items, $T[0], T[1], \ldots, T[n-1]$, and a query item q, find a pair of indices (j, k) corresponding to two successive items, $T[j]$ and $T[k]$, which satisfy that $T[j] \leq q \leq T[k]$.

The query item q may not necessarily exist in the table T. If q exists in the table T, an algorithm returns an index m corresponding to $q(= T[m])$, i.e., $j = k = m$. Note that the items in the table and the query item may not necessarily be numbers like the problems in the next section (e.g., they may be coordinates). In the following, we show that a QTM can solve this problem in $O(n^{1/2})$ time.

First, we provide a quantum search algorithm for solving this problem. In this algorithm, we also use the extended Grover's search algorithm as a subroutine. We assume that we can examine whether the query item q is equal to an item in the table for infinite-precision in $O(1)$ time or not. For simplicity, we also assume that all the items in the table T are distinct. The algorithm will execute in $O(n^{1/2})$ time, even if all the items in the table T are not distinct.

EXTENDED QUANTUM SEARCH ALGORITHM:

1. Choose an index m for $0 \leq m \leq n-1$ at random.

2. If $T[m] > q$, $j \leftarrow *_j$ and $k \leftarrow m$, where $*_j$ means that $T[*_j] = -\infty$.
3. If $T[m] < q$, $j \leftarrow m$ and $k \leftarrow *_k$, where $*_k$ means that $T[*_k] = \infty$.
4. If $T[m] = q$, return (m, m).
5. Repeat the following procedures $O(\log_2 n)$ times and interrupt it when the total executing time becomes in $O(n^{1/2})$ time.
   (a) Initialize the system to
   $$\frac{1}{n^{1/2}} \sum_{i=0}^{n-1} |i, j, k\rangle.$$
   (b) Apply the extended Grover's search algorithm in order to search the index of an item between $T[j]$ and $T[k]$ and obtain the index m satisfying $T[j] < T[m] < T[k]$.
   (c) If $T[m] > q$, $k \leftarrow m$.
   (d) If $T[m] < q$, $j \leftarrow m$.
   (e) If $T[m] = q$, return (m, m).
6. Return (j, k).

Obviously, if $q = -\infty$ (resp. $q = \infty$), we can obtain the minimum value (resp. the maximum value) of the table. Moreover, we can easily make a modified algorithm executing in $O(n^{1/2})$ time to find the $(k-1)$th smallest item or the $(k+1)$th smallest item when the kth smallest item is given.

Next, we prove that the algorithm mentioned above can execute in $O(n^{1/2})$ time to solve the problem. The following two lemmas determine the upper bound of the number of iterations for solving the problem.

Lemma 6.4 [85]. *For an unsorted table of n distinct items, we choose one item t at random. The item t separates the table into three parts (i.e., the set of items that are greater than t, the set of items that are equal to t, and the set of items that are less than t). Then, the expected number of items in the set that includes the kth smallest item is at most $\frac{n}{2} + \frac{k(n-k)}{n}$. Therefore, for any $k (1 \leq k \leq n)$, the expected number of items in the set is at most $\frac{3}{4}n$.*

Proof. The expected number of items is
$$\frac{1}{n}\left(\sum_{t=1}^{k-1}(n-t) + 1 + \sum_{t=k+1}^{n}(t-1)\right).$$

Then,

$$\begin{aligned}
&\frac{1}{n}(\sum_{t=1}^{k-1}(n-t)+1+\sum_{t=k+1}^{n}(t-1)) \\
&= \frac{1}{n}(\sum_{t=1}^{k-1}(n-t)+1+\sum_{t=1}^{n}(t-1)-\sum_{t=1}^{k}(t-1)) \\
&= \frac{1}{n}(n(k-1)-\frac{(k-1)k}{2}+1+\frac{n(n+1)}{2}-n \\
&\quad -\frac{k(k+1)}{2}+k) \\
&= \frac{n}{2}+\frac{1}{n}(nk-k^2-\frac{3}{2}n+k+1) \\
&< \frac{n}{2}+\frac{k(n-k)}{n} \\
&\le \frac{3}{4}n.
\end{aligned}$$

□

Lemma 6.5 [85]. *For a table of* n *items, there is an algorithm to output a table of size* αn*, where* $0 < \alpha < 1$*. Then, in order to make a table of a constant size (i.e., at most size* c*), the minimum value of iterations of the algorithm is at most* $\left\lceil -\frac{log_2 n - log_2 c}{\log_2 \alpha}\right\rceil$.

Proof. Let r be the minimum value of iterations to make a table of constant size. Since $\alpha^r n \le c$,

$$r = \left\lceil -\frac{log_2 n - log_2 c}{\log_2 \alpha}\right\rceil.$$

□

From these two lemmas, we can prove that our search algorithm mentioned above can solve the given problem in $O(n^{1/2})$ time.

Theorem 6.6 [85]. *For an unsorted table* T *of* n *items,* $T[0], T[1], \ldots, T[n-1]$*, and a query item* q*, there is a quantum search algorithm that finds a pair of indices* (j, k) *corresponding to two successive items,* $T[j]$ *and* $T[k]$*, which satisfy that* $T[j] \le q \le T[k]$*, in expected time* $O(n^{1/2})$ *with bounded error probability.*

Proof. The main procedure of this algorithm is LINE 5, especially LINE 5-b. From Theorem 6.2, LINE 5-b can be executed in $c\left(\frac{n}{t}\right)^{1/2}$ time for some constant c, where t is the number of items between $T[j]$ and $T[k]$. Then, by using Lemma 6.4, the expected time $E(n)$ of LINE 5 is

$$E(n) < \sum_{i=1}^{r} c \left(\frac{n}{\left(\frac{3}{4}\right)^i n} \right)^{1/2},$$

where r is the value determined from Lemma 6.5, i.e., the minimum value satisfying that $(3/4)^r n \leq 1$. Then,

$$E(n) < \frac{c\left(\frac{4}{3}\right)^{1/2}\left(\left(\frac{4}{3}n\right)^{1/2} - 1\right)}{\left(\frac{4}{3}\right)^{1/2} - 1} = O(n^{1/2}).$$

□

Our quantum search algorithm can search a pair of indices $(j,\ k)$ satisfying that $T[j] \leq q \leq T[k]$, even if the query item does not necessarily exist in the table. Moreover, if we can find or make a suitable query item to search an item, this algorithm can search the particular item such as the maximum value of the table that is smaller than a given item.

6.3 Applications

By using this quantum search algorithm, we can efficiently solve some problems in computational geometry on QTMs (e.g., see [51, 98]). In this section, for example, let us consider the following fundamental problems.

1. Problem(CLOSEST PAIR)

   Given n points in the plane, find two points whose mutual distance is the smallest.

2. Problem(SET DIAMETER)

   Given n points in the plane, find two points that are the farthest apart.

3. Problem(LINE-SEGMENT INTERSECTION TEST)

   Given n line segment in the plane, decide whether any two intersect.

4. Problem(ELEMENT UNIQUENESS)

   Given n numbers, decide whether any two are equal.

Throughout this section, we assume that one examination of each problem (e.g., For ELEMENT UNIQUENESS, checking whether two items are equal or not) can be executed for infinite-precision in $O(1)$ time.

It is known that ordinary computers can solve each of the problems above in $O(n \log_2 n)$ time (e.g., see [98, 37, 107]), on the other hand, we can show that a QTM solves it in $O(n)$ time.

Theorem 6.7 [85]. *There is a quantum algorithm that solves each of the problems above in expected time $O(n)$ with bounded error probability.*

Proof. For n points, the number of examinations to solve the problem is

$$\binom{n}{2} = O(n^2).$$

Then, since a QTM can check all the examinations in parallel, from Theorem 6.6, the QTM can solve it in expected time

$$O\left((n^2)^{1/2}\right) = O(n).$$

□

Next, we generalize the problems above to $d(> 2)$-dimensional space.

1. Problem(d-DIMENSIONAL CLOSEST PAIR)

   Given n points in d-dimensional space, find two points whose mutual distance is the smallest.

2. Problem(d-DIMENSIONAL SET DIAMETER)

   Given n points in d-dimensional space, find two points that are the farthest apart.

3. Problem(d-DIMENSIONAL LINE-SEGMENT INTERSECTION TEST)

   Given n line segment in d-dimensional space, decide whether any two intersect.

For $d(> 2)$-dimensional space, in some cases, the complexity of the problems increases on ordinary computers. For instance, ordinary computers can solve SET DIAMETER in the plane in $O(n \log_2 n)$ time, on the other hand, in order to solve it in $d(> 2)$-dimensional space, they need $O(n^{2-\varepsilon})$ time for $0 < \varepsilon < 1$ [126]. For $d > 3$, there is a randomized algorithm with $O(n^{\lfloor d/2 \rfloor})$ expected time [37, 107]. However, the complexity for solving them does not increase on the QTMs. Note that we use the assumption that one examination of each problem can be executed in $O(1)$ time.

Corollary 6.1 [85]. *There is a quantum algorithm that solves each of the problems above in expected time $O(n)$ with bounded error probability.* □

If one examination of each problem spends $P(d)$ time for d-dimensional space, our search algorithm needs $O(P(d)n)$ time to solve the problem.

Finally, let us consider the following search problem.

1. Problem(NEAREST NEIGHBOR SEARCH)

   Given N points in $d(\geq 1)$-dimensional space, how quickly can the nearest neighbor of a new given query point q be found?

This problem is known as the *post-office problem*: given a set of n points in the plane (called post-offices or sites), determine for an arbitrary point (x, y) which post-office is the closest to (x, y). Ordinary computers can solve it in $O(n)$ time [69]. On the other hand, it can be solved in $O(n^{1/2})$ time on a QTM because there is a minimum search algorithm executing in $O(n^{1/2})$ time on the QTM. Moreover, for one dimensional space, our search algorithm can solve it directly in $O(n^{1/2})$ time.

Theorem 6.8 [85]. *There is a quantum algorithm that solves NEAREST NEIGHBOR SEARCH in expected time $O(n^{1/2})$ with bounded error probability.*

Proof. For n points, the number of checking the distance from each point to the query point is n. Then, since a QTM can check all the points in parallel, from Theorem 6.6, the QTM can solve it in expected time $O(n^{1/2})$. $\square$

Chapter 7

The Complexity of NP-complete Problems

7.1 Introduction

Although NP-complete problems appear in many situations (e.g., scheduling, artificial intelligence, and pattern recognition, etc.), nobody knows whether DTMs and PTMs can solve them efficiently or not. It is one of the most important issues in theoretical computer science to find a method for solving NP-complete problems efficiently. In this chapter, we investigate methods to solve NP-complete problems on QTMs.

Even if a QTM can compute all the values of a function by using quantum parallel computation, we cannot, in general, obtain all the values of the function simultaneously. Moreover, according to current quantum physics, it is not certain whether we can efficiently read each value in the obtained superposition. These are because the following measurement problem has not been completely solved.

Measurement Problem in Quantum Physics :

What will happen when we measure a quantum physical object ?
Explain it in terms of quantum physics.

Therefore, we study the relationships between the assumptions on measurement and the efficiency of quantum computation. Especially, we study the satisfiability problem.

The *satisfiability problem* (SAT) is to determine whether a given Boolean formula is satisfiable. A *Boolean formula* is a formula composed of variables, parentheses, and operators $\wedge$ (AND), $\vee$ (OR) and $\neg$ (NOT). A Boolean formula is said to be *satisfiable* if there is an assignment of 0's and 1's to the variables that gives the formula the value 1. SAT is a typical NP-complete problem [56].

In this chapter, we propose the following two assumptions on measurement.

1. Assumption Π_1 :

A superposition of physical states is preserved after measurements, and all of the states in the superposition can be measured in time proportional to the number of the states in the superposition.

2. Assumption Π_2 :

   We can measure the existence of a specific physical state C in a given superposition with certainty in polynomial time, if the state C exists in the superposition.

Consequently, we show that there is a QTM that solves SAT in $O(2^{n/4})$ time under Assumption Π_1 and there is a QTM that solves SAT in $n^{O(1)}$ time under Assumption Π_2, where n is the length of an instance of SAT.

The assumptions mentioned above are not widely supported in current quantum physics, however, nobody knows whether these assumptions are valid or not. This is because *interpretations of measurement* have not been fixed yet among physicists. The measurement problem is one of the central issues in quantum physics and several interpretations of measurement exist. In fact, Aharonov et al. [1] have proposed a new interpretation of measurement, i.e., they conjectured that a physical state actually exists as a superposition and can be measured without collapsing the superposition for a special case. In this situation, it is important to find various relationships between the restrictions on measurement and the efficiency of quantum computation.

7.2 Solving SAT under the First Assumption

In this section, first, we propose the following assumption.

Assumption Π_1 : A superposition of configurations is preserved after measurements and all of the configurations in the superposition can be measured in time proportional to the number of the configurations in the superposition.

Then, we show a method to solve SAT in $O(2^{n/4})$ time on a QTM under Assumption Π_1, where n is the length of an instance of SAT, i.e., n is the total length of a description of a logical formula f whose satisfiability should be decided and a description of m that is the number of variables in f [82].

Let a QTM U have an input tape T_1, a work tape T_2, and a history tape T_3, and H_1, H_2, and H_3 be the heads on the tapes T_1, T_2, and T_3, respectively (see Fig. 7.1). All of the heads H_1, H_2, and H_3 can read and write. As shown in Fig. 7.1, a program P that U executes is written on the input tape T_1. On the right-hand side of P, infinitely many blank symbols are written. Note that the length of the description of P is a constant that is independent of the length of an input given on T_2.

The program P consists of state transition rules of a one-tape standard DTM and sentences corresponding to the unitary transforms of Eq.(3.1). Let

$V(n, i)$

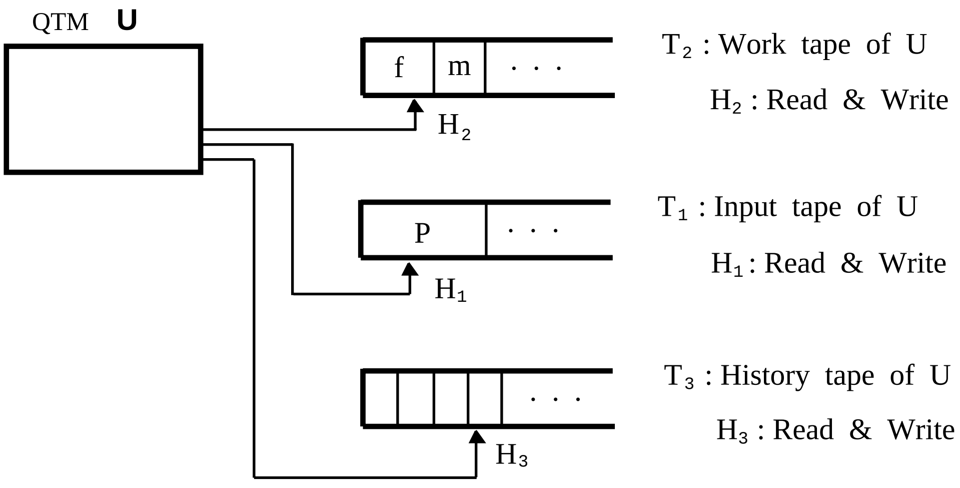

Figure 7.1: The QTM U.

be the sentence that represents that

"apply the transform V_n of Eq.(3.1) on the ith qubit of the work tape",

where $0 \leq n \leq 7$ and $\alpha = \pi/4$ in Eq.(3.1).

The QTM U starts the execution given a logical formula f and the number m of variables in f on the work tape T_2, and the program P on the input tape T_1. In fact, the number m of variables in f need not be supplied as an input because U can initially scan the description of f on T_2 from left to right and can count the number of variables appearing in that description. However, in order to simplify the presentations, we assume that m is also supplied as input.

The *time complexity* of the QTM U is the sum of the number of steps executed by U until it finally halts and the number of steps needed to measure the output. The time complexity of U is represented as a function of the length of the input, n, (in this case, the total length of the descriptions of f and m) given on T_2.

Theorem 7.1 [82]. *Under Assumption* Π_1*, the QTM* U *can solve SAT in* $O(2^{n/4})$ *time, where* n *is the length of an instance of SAT.*

Proof. We show a program P that U executes in order to solve SAT in $O(2^{n/4})$ time. Let us consider the satisfiability of an m-variable logical formula $f(x_1, x_2, \ldots, x_m)$, where x_i, $i = 1, 2, \ldots, m$ are Boolean variables. Without loss of generality, we can assume that f is in conjunctive normal form. A formula f is in *conjunctive normal form* (*CNF*) if it is of the form $f = F_1 \wedge F_2 \wedge \cdots \wedge F_q$. Each F_i is a clause of the form $(x_{i1} \vee x_{i2} \vee \cdots \vee x_{in_i})$, where each x_{ij} is a literal (i.e., either a variable or the negation of a variable) and n_i is the number of literals in clause F_i. We also assume that the variables $x_1, \ldots, x_m$ are

```
begin
%%% Preparation of partial assignments
1      for $i = 1$ to $\lceil \frac{m}{4} \rceil$ do
2          $V(4, i)$
       od ;
%%% Computation of the values of $f$
3      Reduce an instance of SAT on $\lfloor \frac{3}{4} m \rfloor$-variable formula
       to the corresponding maximum independent set problem.
4      Solve the obtained maximum independent set problem
       by using the algorithm of Tarjan & Trojanowski.
end.
```

Figure 7.2: The program that the QTM U executes under Assumption Π_1.

named in such a way that if $i < j$ the number of occurrences of x_j in f is not larger than that of x_i.

The QTM U writes an assignments to the variables in f together with the corresponding value of f on the tape T_2. We show the program executed by U in Fig. 7.2. We explain the behavior of U according to this program.

An Execution

1. The initial configuration

   The descriptions of the logical formula f and the number m of the variables are given as input on the work tape T_2, and the program P that U executes is given on the input tape T_1. We identify a configuration of U with a description of an assignment to the variables in f and the corresponding value of f, which are written on the tape T_2 (this description is written on the right-hand side of the descriptions of f and m on T_2). Let $|x_1, \ldots, x_m, x_{m+1}\rangle$ be a register corresponding to $x_1, \ldots, x_m$ and the value of f under the assignment in this configuration. Now, let

   $$|\underbrace{0, 0, \ldots, 0}_{m}; 0\rangle$$

   be the initial configuration of U. In order to simplify the presentation, we separate the assignments for the variables from the value of f by semicolon.

2. Making a superposition of partial assignments

   There are 2^m different assignments for the m variables in f. Initially, U makes a superposition of all the configurations in which $\lceil \frac{m}{4} \rceil$ variables $x_1, \ldots, x_{\lceil \frac{m}{4} \rceil}$ are fixed. U will perform this by applying

$$V_4 = \frac{1}{2^{1/2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

to each qubit corresponding to $\left\lceil \frac{m}{4} \right\rceil$ variables in order. U can execute a transform by V_4 in a single step. By the execution of the **for** loop, the initial configuration is transformed as follows (see Fig. 7.3):

$$|0, 0, \ldots, 0; 0\rangle$$
$$\overset{U_{x_1}, \ldots, U_{x_{\lceil \frac{m}{4} \rceil}}}{\longrightarrow} \quad \frac{1}{\sqrt{2^{\lceil \frac{m}{4} \rceil}}} \sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_{\lceil \frac{m}{4} \rceil}=0}^{1} |x_1, x_2, \ldots, x_{\lceil \frac{m}{4} \rceil}, 0 \ldots, 0; 0\rangle,$$

where U_{x_i} is a unitary transform represented by a matrix

$$M_{x_i} = \underbrace{I \otimes \cdots \otimes I}_{i-1} \otimes V_4 \otimes \underbrace{I \otimes \cdots \otimes I}_{m-i}.$$

Since U can execute each transform U_{x_i} in a single step, it can execute all the transforms $U_{x_1}, \ldots, U_{x_{\lceil \frac{m}{4} \rceil}}$ in $\left\lceil \frac{m}{4} \right\rceil$ steps.

3. Computation of the values of f

   Now, in the logical formula f, the variables $x_1, \ldots, x_{\lceil \frac{m}{4} \rceil}$ are fixed and the other $\left\lfloor \frac{3}{4} m \right\rfloor$ variables are free. By the assumption on the names of the variables, the length of the description of f will be less than or equal to $\frac{3}{4}n$ if we evaluate f as much as possible.

   In the third line in Fig. 7.2, U transforms this logical formula in quantum parallel to the corresponding instance of the maximum independent set problem. The *maximum independent set problem* is to find a maximum-size independent set in a given graph $G = (V, E)$. A set S of vertices is *independent* if the edge $(u, v) \notin E$ for all $u, v \in S$. The maximum independent set problem is known to be NP-complete.

   The transformation will be performed in the following way. Let $f = F_1 \wedge F_2 \wedge \cdots \wedge F_q$ be an expression in CNF. Each F_i is a clause of the form $(x_{i1} \vee x_{i2} \vee \cdots \vee x_{in_i})$, where each x_{ij} is a literal and n_i is the number of literals in clause F_i. U constructs an undirected graph $G = (V, E)$ whose vertices are pairs of integers (i, j) for $1 \leq i \leq q$ and $1 \leq j \leq n_i$. The vertex (i, j) represents the jth literal of the ith clause. The edges of the graph are $((i, j), (k, l))$ if $i = k$ or $x_{ij} = \neg x_{kl}$. This construction has the property that G has a maximum independent set of size q if and only if f is satisfiable. This transformation can be executed in polynomial time (e.g., see [61]). Note that the number of the vertices in the instance is less than or equal to $\frac{3}{4}n$, because the length of the description of f is less than or equal to $\frac{3}{4}n$.

   Next, in the fourth line in Fig. 7.2, U solves those instances of the maximum independent set problem in quantum parallel by using the algorithm of Tarjan and Trojanowski [118]. This process can be performed in $O\left(2^{\frac{1}{3}\frac{3n}{4}}\right) = O(2^{n/4})$ time. In Fig. 7.3, we illustrate the whole computation executed by U.

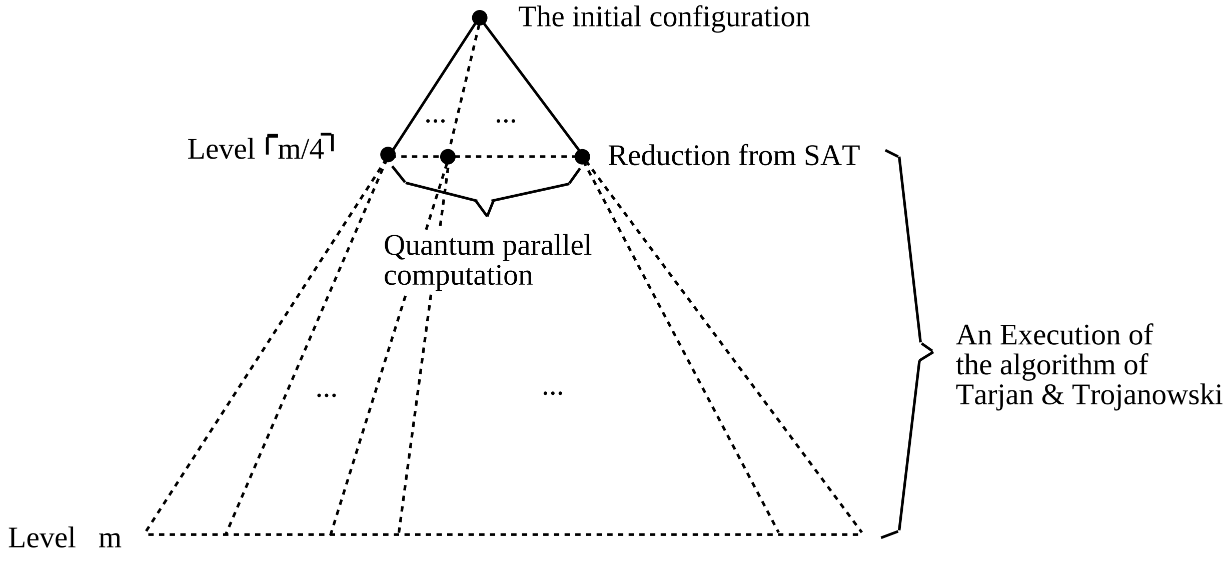
The initial configuration
Level ⌈m/4⌉
Reduction from SAT
Quantum parallel computation
An Execution of the algorithm of Tarjan & Trojanowski
Level m

Figure 7.3: The computation executed by the QTM U.

A Measurement

For some positive constant c depending on the input f, U completes all the computation above within $c \cdot 2^{n/4}$ time. Then, we will measure all of the superposed $2^{\lceil \frac{m}{4} \rceil}$ configurations. If all the output qubits of these configurations are zeros, f will be unsatisfiable. On the other hand, if at least one output qubit is one, f will be satisfiable. We can measure each of these configurations, because the superposed configurations are preserved after each measurement by Assumption Π_1. Furthermore, these measurements can be executed in $O(2^{n/4})$ time by Assumption Π_1.

The Time Complexity of the QTM U

In Fig. 7.2, the **for** loop is executed $\left\lceil \frac{m}{4} \right\rceil$ times in total, thus U executes it within $O(n)$ time. As mentioned above, U can execute the third line in polynomial time and the fourth line in $O(2^{n/4})$ time. Finally, U can measure all the configurations in $O(2^{n/4})$ time. Therefore, U can execute the procedure in Fig. 7.2 in $O(2^{n/4})$ time in total. This completes the proof of Theorem 7.1. □

Corollary 7.1 [82]. *If there is a deterministic algorithm to solve the maximum independent set problem in $O(2^{\varepsilon n})$ time $(0 < \varepsilon < 1)$, the QTM U can solve SAT in $O(2^{\frac{\varepsilon}{\varepsilon+1} n})$ time under Assumption Π_1.*

Proof. The QTM U executes a quite similar procedure as shown in Fig. 7.2. In this case, U makes a superposition of $2^{\lceil \frac{\varepsilon}{\varepsilon+1} m \rceil}$ different partial assignments for $x_1, \ldots, x_{\lceil \frac{\varepsilon}{\varepsilon+1} m \rceil}$ variables. Then, U executes the same procedure shown in Fig. 7.2. Solving the instances

```
begin
%%% Preparation of all the assignments
1       for $i = 1$ to $m$ do
2           $V(4, i)$
        od ;
%%% Computation of the values of $f$
3       $x_{m+1} := f(x_1, \ldots, x_m)$ ;
%%% Preparations for a measurement
4       for $j = m$ to 1 do
5           $V(0, j)$
        od
end.
```

Figure 7.4: The program that the QTM U executes under Assumption Π_2.

of the maximum independent set problem and the measurement can be executed in $O(2^{\frac{\varepsilon}{\varepsilon+1}n})$ time in this case. □

7.3 Solving SAT under the Second Assumption

In this section, we show a method to solve SAT in polynomial time on a QTM under the following stronger assumption Π_2 [81].

Assumption Π_2 : Let C be a specific configuration. We can measure the existence of C in a superposition with certainty in polynomial time in the input size of C, if C exists in the superposition.

Theorem 7.2 [81]. *Under Assumption Π_2, the QTM U can solve SAT in $n^{O(1)}$ time, where n is the length of an instance of SAT.*

Proof. We show a program P that U executes in order to solve SAT in polynomial time. We consider the satisfiability of an m-variable logical formula $f(x_1, x_2, \ldots, x_m)$, where x_i, $i = 1, 2, \ldots, m$ are Boolean variables.

The QTM U writes an assignments to the variables in f together with the corresponding value of f on the tape T_2. We show the program executed by U in Fig. 7.4. We explain the behavior of U according to this program.

An Execution

1. The initial configuration

   The initial configuration is the same as in the proof of Theorem 7.1, i.e.,

$$|\underbrace{0,0,\ldots,0}_{m};0\rangle.$$

2. Making a superposition of all the assignments

   There are 2^m different assignments for m variables in f. Initially, U makes a superposition of configurations corresponding to all of these assignments. U will perform this by applying

   $$V_4 = \frac{1}{2^{1/2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

   to each qubit corresponding to m variables in order. U can execute the transform by V_4 in a single step. By the execution of the first **for** loop, the initial configuration is transformed as follows:

   $$|0,0,\ldots,0;0\rangle \overset{U_{x_1},\ldots,U_{x_m}}{\longrightarrow} \frac{1}{2^{m/2}}\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_m=0}^{1}|x_1,x_2,\ldots,x_m;0\rangle.$$

   Since U can execute each transform U_{x_i} in a single step, it can execute all the transforms $U_{x_1},\ldots,U_{x_m}$ in m steps.

3. Computation of the values of f

   Let U_f be a transform corresponding to the computation of the value of the logical formula f, then,

   $$\begin{aligned}&\frac{1}{2^{m/2}}\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_m=0}^{1}|x_1,x_2,\ldots,x_m;0\rangle\\ &\overset{U_f}{\longrightarrow}\ \frac{1}{2^{m/2}}\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_m=0}^{1}|x_1,x_2,\ldots,x_m;f(x_1,x_2,\ldots,x_m)\rangle \equiv \psi.\end{aligned}$$

   In order to execute the third line of the program, U simulates a program of a one-tape standard DTM M_f computing the values of f. U simulates M_f, regarding T_2 as the tape of M_f. Recall that f and m are written on T_2, followed by an assignment $|a_1,\ldots,a_m\rangle$ ($a_i \in \{0,1\}$, $1 \le i \le m$) for $x_1,\ldots,x_m$.

   First, M_f returns the head to the leftmost square of the tape and then executes the following procedure: (1) it finds the leftmost variable x_a appearing in f, (2) it finds the value for x_a (0 or 1) in $|a_1,\ldots,a_m\rangle$, and (3) it rewrites x_a of (1) to the value found in (2). Next, M_f executes the same procedure for the variable appearing next to x_a, and so on. By repeating the same procedure, M_f rewrites all the variables appearing in f to the corresponding values in $|a_1,\ldots,a_m\rangle$. After that, M_f returns the head to the leftmost square of the tape and then evaluates the value of f (i.e., a_{m+1}) under the assignment $|a_1,\ldots,a_m\rangle$.

   Obviously, U can execute the whole computation above in $O(n^2)$ time.

4. Preparations for a measurement

   The QTM U performs the reverse transform of the procedure performed in Step 2 by applying

$$V_0 = \frac{1}{2^{1/2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

   to the qubits corresponding to the m variables from $|x_m\rangle$ to $|x_1\rangle$ in order. Since U can perform a transform corresponding to V_0 in a single step, it can execute the **for** loop of the fourth line in Fig. 7.4 in m steps. In this transform, the superposition ψ of U's configurations will be transformed as follows:

   (a) if the values of f for 2^m different assignments are all 0's, ψ will be transformed to the single configuration $|0, 0, \ldots, 0; 0\rangle$,

   (b) if the values of f for 2^m different assignments are all 1's, ψ will be transformed to the single configuration $|0, 0, \ldots, 0; 1\rangle$,

   (c) otherwise, ψ will be transformed to a superposition of several configurations.

A Measurement

When U completes all the computations mentioned above, we will measure whether there is a configuration $c_1 = |0, 0, \ldots, 0; 1\rangle$ in the finally obtained superposition of U's configurations. If c_1 exists in the superposition, we conclude that f is satisfiable, and if not, f is unsatisfiable. The correctness of this decision is shown by the claim below.

Claim 1. *f is satisfiable if and only if there is the configuration $|0, 0, \ldots, 0; 1\rangle$ in the finally obtained superposition of the configurations.*

Proof. ($\Leftarrow$) It is obvious, because if f is unsatisfiable, only $|0, 0, \ldots, 0; 0\rangle$ will exist in the finally obtained superposition.

($\Rightarrow$) We will show that if f is satisfiable, there is $|0, 0, \ldots, 0; 1\rangle$ in the finally obtained superposition.

First, let us consider an application of the transform corresponding to V_0 to each configuration. Let $1 \le i \le m$. If for a configuration $|x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_m; 1\rangle$, where $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m \in \{0, 1\}$, U executes $V(0, i)$, the configuration will be transformed as follows:

$$\begin{aligned} &|x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_m; 1\rangle \\ &\to \frac{1}{2^{1/2}}(|x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_m; 1\rangle - |x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_m; 1\rangle). \end{aligned}$$

On the other hand, if U executes $V(0, i)$ for a configuration $|x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_m; 1\rangle$, the configuration will be transformed as follows:

$$|x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_m; 1\rangle$$

$$\to \frac{1}{2^{1/2}}(|x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_m; 1\rangle + |x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_m; 1\rangle).$$

Note that in both cases, there are always two configurations in which $x_i = 0$ and $x_i = 1$ after the execution of $V(0, i)$. Furthermore, the amplitude for the configuration in which $x_i = 0$ is always positive. Therefore, after the U's computation of f, if there is at least one configuration of the form $|x_1, x_2, \ldots, x_m; 1\rangle$ in the obtained superposition of configurations (i.e., ψ above) and U applies V_0 to $|x_m\rangle, \ldots, |x_1\rangle$ in order, the configuration $|0, 0, \ldots, 0; 1\rangle$ will always appear in the end. Moreover, since an amplitude of a configuration in which $x_i = 0$ is always positive for any i, the configuration $|0, 0, \ldots, 0; 1\rangle$ will not be cancelled during the transforms even if we have more than one configurations in which the value of f is 1 before the transforms. (end of claim)

The Time Complexity of the QTM U

Since the second line in Fig. 7.4 is executed m times in total, U can execute the first **for** loop within $O(m)$ steps. As mentioned above, U can execute the third line within $O(n^2)$ time. Moreover, U can also execute the **for** loop of the fourth line within $O(m)$ steps. Therefore, U can execute the procedure in Fig. 7.4 within polynomial time in total. Finally, by Assumption Π_2, we can measure whether there is a configuration $c_1 = |0, 0, \ldots, 0; 1\rangle$ in the finally obtained superposition of U's configurations in polynomial time. This completes the proof of Theorem 7.2. □

Corollary 7.2 [81]. *Under Assumption* Π_2, *NP* $\subseteq$ *EQP.*

Proof. First, the QTM U transforms an instance of any NP-complete problem given on T_2 as input to an equivalent instance of SAT in polynomial time. U executes this process by simulating an appropriate one-tape standard DTM. After U writes the obtained logical formula f on T_2, it executes the same procedure shown in Fig. 7.4. □

This means that

$$\text{NP} \not\subseteq \text{EQP} \Rightarrow \neg\text{Assumption } \Pi_2.$$

Namely, if NP $\not\subseteq$ EQP is shown, it follows that Assumption Π_2 is not valid in quantum physics. In this sense, our results establish an interesting relationship between quantum physics and computational complexity theory.

Example : In Fig. 7.5, we show the changes of the superpositions of configurations of the QTM U in the case of a function f such that $f(0, 0) = f(1, 1) = 0$ and $f(0, 1) = f(1, 0) = 1$.

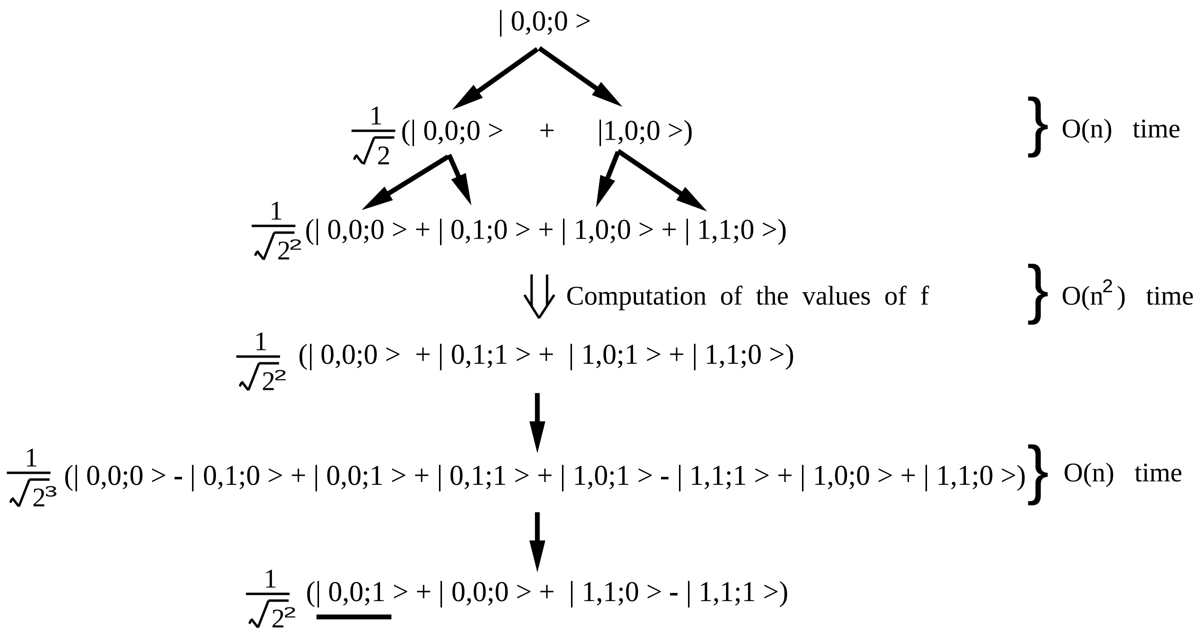
$| 0,0;0 >$
$\frac{1}{\sqrt{2}}(| 0,0;0 > + |1,0;0 >)$
O(n) time
$\frac{1}{\sqrt{2^2}}(| 0,0;0 > + | 0,1;0 > + | 1,0;0 > + | 1,1;0 >)$
Computation of the values of f
O(n^2) time
$\frac{1}{\sqrt{2^2}}(| 0,0;0 > + | 0,1;1 > + | 1,0;1 > + | 1,1;0 >)$
$\frac{1}{\sqrt{2^3}}(| 0,0;0 > - | 0,1;0 > + | 0,0;1 > + | 0,1;1 > + | 1,0;1 > - | 1,1;1 > + | 1,0;0 > + | 1,1;0 >)$
O(n) time
$\frac{1}{\sqrt{2^2}}(\underline{| 0,0;1 >} + | 0,0;0 > + | 1,1;0 > - | 1,1;1 >)$

Figure 7.5: The changes of the superposition of configurations of the QTM U.

Chapter 8

Conclusions

In this thesis, we showed some results on the complexity and algorithms based on QTMs. As main results, first, we showed that the periods of periodic functions that have some input properties, $f(x) = f(x+r)$ and $x = f^r(x)$, can be found in polynomial time on a QTM, even if ordinary computers cannot find it in polynomial time. Therefore, some kinds of functions shown in Chapter 5 are not appropriate for pseudo-random generators on the QTM, because the period of a recurrence $x_{i+1} = f(x_i)$ can be efficiently found on the QTM. Moreover, in some case, we will be able to extend in a recurrence $x_{i+1} = f(x_i, x_{i-1}, \ldots, x_{i-j})$ for $i \geq j$.

Next, we showed an efficient quantum search algorithm. Given an unsorted table T of n items, $T[0], \ldots, T[n-1]$ and a query item q, we showed that there is a quantum search algorithm that finds a pair of indices (j,k) corresponding to two successive items, $T[j]$ and $T[k]$, which satisfy that $T[j] \leq q \leq T[k]$, in $O(n^{1/2})$ time. This search algorithm extends Dürr & Høyer's minimum search algorithm, and can obtain the indices corresponding to the successor and the predecessor of q even if q does not necessarily exist in the table. Furthermore, we also applied our search algorithm to some problems in computational geometry. However, there are many problems that need to search items efficiently in several fields. We will be able to apply our algorithm to these problems. Moreover, if a table has a particular structure, we will be able to make a more efficient improved search algorithm.

When we solve these problems above, we applied the quantum Fourier transform and the quantum iterating method. However, there are other methods to solve problems on QTMs faster than ordinary computers (e.g., [41]). Combining these methods, QTMs may be able to efficiently solve problems that cannot be efficiently solved on ordinary computers.

Finally, we studied the relationships between the assumptions on measurement and the efficiency of quantum computation. We showed that, if the maximum independent set problem can be solved in $O(2^{\varepsilon n})$ $(0 < \varepsilon < 1)$ time, a QTM can solve SAT in $O(2^{\frac{\varepsilon}{\varepsilon+1}n})$ time under Assumption Π_1. However, we cannot expect that the QTM solves SAT faster than $O(2^{\varepsilon' n})$ time by the same method, because it is known that Tarjan & Trojanowski type algorithms for finding the maximum independent set must require $2^{\varepsilon n}$ time in the worst case for some small positive ε [64, 87, 103, 118, 124]. We also showed that a QTM can solve SAT in polynomial time under Assumption Π_2,

These two assumptions are not widely supported in current quantum physics. Since NP-complete problems are one of the most familiar problems in theoretical computer science, it is the most important open question to show whether NP-complete problems can be solved in polynomial time on a QTM under only assumptions supported in current quantum physics (i.e., deciding whether NP $\subseteq$ BQP). It will be also important for constructing quantum computers to find an efficient algorithm for solving NP-complete problems on a QTM. However, there are also some weak indications that QTMs may not efficiently solve NP-complete problems. In fact, it is shown that there is an oracle R such that $\mathrm{NP}^R \not\subseteq \mathrm{BQP}^R$[22].

Quantum computers does not exist yet. At the present time, they are only one of mathematical computing models, computing models based on quantum physics. Namely, many results of the complexity and algorithms on quantum computers are also theoretical results. Moreover, some physicists insist that we will not be abel to realize quantum computers if we can not control coherent states (e.g., [74, 75, 121]). On the other hand, there are also many physicists and computer scientists that study methods for realizing quantum computers. Some researchers propose physical systems for realizing quantum gates [4, 6, 36, 92, 114] and some demonstrate the quantum gates proposed above in the laboratory [88, 120].

The studies of quantum computers have been initiated only fifteen years ago. Therefore, the computing powers, properties, and definitions of quantum computers should be studied in detail in the future. Moreover, other quantum computing models such as quantum circuits and quantum cellular automata should be studied [48, 49, 125, 127]. If we can establish good results in this kind of field, we will be able to claim that quantum computers are effective not only as theoretical models of computers but also as practical models of computers.

Appendix A

Quantum Theory

In order to understand the formalizations of quantum computers, we should understand the fundamentals of quantum physics. However, since it is not a purpose of this thesis to denote completely quantum physics, we describe only the fundamental parts of quantum physics needed to understand quantum computers. For a more complete overview on quantum physics, the reader is referred to [43, 63, 80, 94, 100, 101].

A.1 Hilbert Spaces

A *Hilbert space* is defined as a non-empty set $\mathcal{H}$ with the following three properties (e.g., see [38, 47, 63]).

1. A Hilbert space $\mathcal{H}$ is a complex vector space.

   Let V be a non-empty set, $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$, and $a, b \in \mathbf{C}$. Then, a *complex vector space* is the set V with $a\mathbf{x}, \mathbf{x}+\mathbf{y} \in V$, and the following properties:

   (a) $\mathbf{x}+\mathbf{y} = \mathbf{y}+\mathbf{x}$.

   (b) $\mathbf{x}+(\mathbf{y}+\mathbf{z}) = (\mathbf{x}+\mathbf{y})+\mathbf{z}$.

   (c) For any $\mathbf{x}$, there is $\mathbf{0} \in V$ such that $\mathbf{x}+\mathbf{0} = \mathbf{x}$.

   (d) For any $\mathbf{x}$, there is $-\mathbf{x} \in V$ such that $\mathbf{x}+(-\mathbf{x}) = \mathbf{0}$.

   (e) $a(\mathbf{x}+\mathbf{y}) = a\mathbf{x}+a\mathbf{y}$.

   (f) $(a+b)\mathbf{x} = a\mathbf{x}+b\mathbf{x}$.

   (g) $a(b\mathbf{x}) = (ab)\mathbf{x}$.

   (h) $1\mathbf{x} = \mathbf{x}$.

2. An inner product is defined on $\mathcal{H}$.

   Let $\mathbf{x}, \mathbf{y} \in \mathcal{H}$. Then, an *inner product* of $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} x_i{}^* y_i,$$

where x_i is the ith element of $\mathbf{x}$ and $x_i{}^*$ is the complex conjugate of x_i. Moreover, let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{H}$ and $a, b \in \mathbf{C}$. Then, the inner product satisfies the following properties:

(a) $(\mathbf{y}, \mathbf{x}) = (\mathbf{x}, \mathbf{y})^*$.

(b) $(\mathbf{x}, a\mathbf{y} + b\mathbf{z}) = a(\mathbf{x}, \mathbf{y}) + b(\mathbf{x}, \mathbf{z})$.

(c) $(\mathbf{x}, \mathbf{x}) \geq 0$, and $(\mathbf{x}, \mathbf{x}) = 0$ implies $\mathbf{x} = \mathbf{0}$.

We call $(\mathbf{x}, \mathbf{x})^{1/2}$ a *norm* of $\mathbf{x}$ and denote it by $\| \mathbf{x} \|$. Moreover, a vector space with the inner product is called an *inner product space* or a *pre-Hilbert space.*

3. A Hilbert space $\mathcal{H}$ is complete with respect to the norm.

   A Cauchy sequence, $\mathbf{x}_1, \mathbf{x}_2, \ldots$, in an inner product space V is as follows:

   > A sequence of vectors, $\mathbf{x}_1, \mathbf{x}_2, \ldots$, in V is called a *Cauchy sequence* if for any $\varepsilon > 0$, there is a positive integer N such that
   >
   > $$\| \mathbf{x}_n - \mathbf{x}_m \| < \varepsilon$$
   >
   > for all $m, n > N$.

   Then, an inner product space V is called *complete* if every Cauchy sequence in V converges to an element of V.

A Hilbert space whose amplitudes are limited to real numbers is called a *real Hilbert space.* The well-known three-dimensional Euclid space is one of real Hilbert spaces.

In quantum physics, we often denote a vector $\mathbf{x} \in \mathcal{H}$ by $|\mathbf{x}\rangle$ and denote the inner product of $|\mathbf{x}\rangle$ and $|\mathbf{y}\rangle$ by $\langle \mathbf{x}|\mathbf{y}\rangle$, which are called *Dirac notation* [43]. For instance, let

$$|\mathbf{x}\rangle = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad |\mathbf{y}\rangle = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Then, since $\langle \mathbf{x}|$ is the transposed conjugate of $|\mathbf{x}\rangle$, i.e.,

$$\langle \mathbf{x}| = (x_1{}^*, x_2{}^*, \ldots, x_n{}^*),$$

the inner product of $|\mathbf{x}\rangle$ and $|\mathbf{y}\rangle$ is

$$\langle \mathbf{x}|\mathbf{y}\rangle = (x_1{}^*, x_2{}^*, \ldots, x_n{}^*) \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \sum_{i=1}^{n} x_i{}^* y_i.$$

A.2 Tensor Products

Let $\mathbf{a}$ and $\mathbf{b}$ be vectors in Hilbert spaces $\mathcal{H}_1^m$ and $\mathcal{H}_2^n$, respectively, where $\mathcal{H}_1^m$ (resp. $\mathcal{H}_2^n$) is an m-dimensional space (resp. an n-dimensional space). Then, a vector $\mathbf{a} \otimes \mathbf{b}$ in a new mn-dimensional Hilbert space $\mathcal{H}_1^m \otimes \mathcal{H}_2^n$ is called a *tensor product* of $\mathcal{H}_1^m$ and $\mathcal{H}_2^n$ (e.g., see [63, 80]). In quantum physics, the tensor product is used when we construct a composite system of some physical systems. In the following, let $\alpha, \beta \in \mathbf{C}$, $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_2 \in \mathcal{H}_1^m$, $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \in \mathcal{H}_2^n$, and $\mathbf{x}, \mathbf{y} \in \mathcal{H}_1^m \otimes \mathcal{H}_2^n$. Here, we only denote principal properties.

1. A tensor product is *linear*.

   (a) $(\alpha \mathbf{a}_1 + \beta \mathbf{a}_2) \otimes \mathbf{b}_1 = (\alpha \mathbf{a}_1) \otimes \mathbf{b}_1 + (\beta \mathbf{a}_2) \otimes \mathbf{b}_1$.

   (b) $\mathbf{a}_1 \otimes (\alpha \mathbf{b}_1 + \beta \mathbf{b}_2) = \mathbf{a}_1 \otimes (\alpha \mathbf{b}_1) + \mathbf{a}_1 \otimes (\beta \mathbf{b}_2)$.

   (c) $\alpha(\mathbf{a}_1 \otimes \mathbf{b}_1) = (\alpha \mathbf{a}_1) \otimes \mathbf{b}_1 = \mathbf{a}_1 \otimes (\alpha \mathbf{b}_1)$.

2. Let $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m\}$ and $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}$ be sets of basis vectors for $\mathcal{H}_1^m$ and $\mathcal{H}_2^n$, respectively. Then, the *basis* for $\mathcal{H}_1^m \otimes \mathcal{H}_2^n$ is the set of vectors $\mathbf{e}_i \otimes \mathbf{f}_j$ for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, and every vector $\mathbf{x} \in \mathcal{H}_1^m \otimes \mathcal{H}_2^n$ can be written as

$$\mathbf{x} = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \mathbf{e}_i \otimes \mathbf{f}_j.$$

3. An *inner product* is defined as follows:

   (a) $(\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2) = (\mathbf{a}_1, \mathbf{a}_2)(\mathbf{b}_1, \mathbf{b}_2)$.

   (b) $(\mathbf{a}_1 \otimes \mathbf{b}_1, (\alpha \mathbf{a}_2 \otimes \mathbf{b}_2 + \beta \mathbf{a}_3 \otimes \mathbf{b}_3)) = \alpha(\mathbf{a}_1, \mathbf{a}_2)(\mathbf{b}_1, \mathbf{b}_2) + \beta(\mathbf{a}_1, \mathbf{a}_3)(\mathbf{b}_1, \mathbf{b}_3)$.

4. A tensor product of operators can also be defined.

   Let $\hat{A}_1$ and $\hat{A}_2$ be operators on $\mathcal{H}_1^m$ and $\mathcal{H}_2^n$, respectively. Then, a *tensor product* $\hat{A}_1 \otimes \hat{A}_2$ is defined as follows:

   (a) $(\hat{A}_1 \otimes \hat{A}_2)\mathbf{a}_1 \otimes \mathbf{b}_1 = (\hat{A}_1 \mathbf{a}_1) \otimes (\hat{A}_2 \mathbf{b}_1)$.

   (b) $(\hat{A}_1 \otimes \hat{A}_2)(\alpha \mathbf{x} + \beta \mathbf{y}) = \alpha(\hat{A}_1 \otimes \hat{A}_2)\mathbf{x} + \beta(\hat{A}_1 \otimes \hat{A}_2)\mathbf{y}$.

For instance, let us consider a case where $m = 3$ and $n = 2$. Let

$$\left\{ \mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}$$

be a basis on $\mathcal{H}_1^3$ and

$$\left\{ \mathbf{f}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{f}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

be a basis on $\mathcal{H}_2^2$, respectively. Then, $\{\mathbf{e}_1 \otimes \mathbf{f}_1, \mathbf{e}_1 \otimes \mathbf{f}_2, \mathbf{e}_2 \otimes \mathbf{f}_1, \mathbf{e}_2 \otimes \mathbf{f}_2, \mathbf{e}_3 \otimes \mathbf{f}_1, \mathbf{e}_3 \otimes \mathbf{f}_2\}$ is a basis on the six-dimensional Hilbert space $\mathcal{H}_1^3 \otimes \mathcal{H}_2^2$. When two vectors $\mathbf{a} \in \mathcal{H}_1^3$ and $\mathbf{b} \in \mathcal{H}_2^2$ are

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \text{ and } \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

respectively, the tensor product of these two vectors is as follows:

$$\mathbf{a} \otimes \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \\ a_3 b_1 \\ a_3 b_2 \end{pmatrix}.$$

Moreover, the tensor product of two matrices is as follows: when

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \text{ and } B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix},$$

then,

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & a_{13}B \\ a_{21}B & a_{22}B & a_{23}B \\ a_{31}B & a_{32}B & a_{33}B \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} & a_{13}b_{11} & a_{13}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} & a_{13}b_{21} & a_{13}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} & a_{23}b_{11} & a_{23}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} & a_{23}b_{21} & a_{23}b_{22} \\ a_{31}b_{11} & a_{31}b_{12} & a_{32}b_{11} & a_{32}b_{12} & a_{33}b_{11} & a_{33}b_{12} \\ a_{31}b_{21} & a_{31}b_{22} & a_{32}b_{21} & a_{32}b_{22} & a_{33}b_{21} & a_{33}b_{22} \end{pmatrix}.$$

A.3 Unitary Operators

First, we denote a linear operator. A linear operator has the following properties (e.g., see [47, 63]).

1. A *linear operator* $\hat{A}$ on a Hilbert space $\mathcal{H}$ associates a vector, denoted by $\hat{A}\mathbf{a}$, with any vector $\mathbf{a} \in \mathcal{H}$ such that

$$\hat{A}(\alpha\mathbf{a} + \beta\mathbf{b}) = \alpha\hat{A}\mathbf{a} + \beta\hat{A}\mathbf{b} \quad \text{for any } \alpha, \beta \in \mathbf{C} \text{ and } \mathbf{a}, \mathbf{b} \in \mathcal{H}.$$

2. The *sum* of a pair of operators, $\hat{A}$ and $\hat{B}$, is the operator $\hat{A} + \hat{B}$ defined by

$$(\hat{A}+\hat{B})\mathbf{a} = \hat{A}\mathbf{a} + \hat{B}\mathbf{a} \quad \text{for any } \mathbf{a} \in \mathcal{H}.$$

3. The *product* of $\hat{A}$ and $\hat{B}$ is the operator $\hat{A}\hat{B}$ defined by

   $$(\hat{A}\hat{B})\mathbf{a} = \hat{A}(\hat{B}\mathbf{a}) \quad \text{for any } \mathbf{a} \in \mathcal{H}.$$

   The product of an operator $\hat{A}$ with a complex number α is the operator $\alpha\hat{A}$ defined by

   $$(\alpha\hat{A})\mathbf{a} = \alpha(\hat{A}\mathbf{a}) \quad \text{for any } \mathbf{a} \in \mathcal{H}.$$

4. A non-zero vector $\mathbf{a} \in \mathcal{H}$ is an *eigenvector* of $\hat{A}$ with *eigenvalue* a if

   $$\hat{A}\mathbf{a} = a\mathbf{a}.$$

5. The set of *matrix elements* of an operator $\hat{A}$ on $\mathcal{H}$ is the collection of all numbers $(\mathbf{a}, \hat{A}\mathbf{b})$.

6. The *adjoint* of an $\hat{A}$ is the operator $\hat{A}^\dagger$ defined by the condition on its matrix elements as follows:

   $$(\mathbf{a}, \hat{A}^\dagger\mathbf{b}) = (\hat{A}\mathbf{a}, \mathbf{b}) \quad \text{for any } \mathbf{a}, \mathbf{b} \in \mathcal{H}.$$

   An operator $\hat{A}$ is *self-adjoint* if

   $$\hat{A}^\dagger = \hat{A}.$$

A unitary operator is a kind of linear operators. That is, a linear operator $\hat{U}$ on $\mathcal{H}$ is *unitary* if it satisfies the following properties:

1. It is invertible.

   $$\hat{U}^\dagger\hat{U} = \hat{U}^\dagger\hat{U} = I,$$

   where I is the unit operator.

2. It preserves all inner products.

   $$(\hat{U}\mathbf{a}, \hat{U}\mathbf{b}) = (\mathbf{a}, \mathbf{b}) \quad \text{for any } \mathbf{a}, \mathbf{b} \in \mathcal{H}.$$

Therefore, the norm of a vector is preserved since

$$\| \hat{U}\mathbf{a} \| = \left(\hat{U}\mathbf{a}, \hat{U}\mathbf{a}\right)^{1/2} = (\mathbf{a}, \mathbf{a})^{1/2} = \| \mathbf{a} \| .$$

A.4 Quantum Physics

The following four rules deal with the general mathematical framework of quantum physics [63].

Rule 1. The predictions of results of measurements made in an isolated system are probabilistic in nature. In situations where the maximum amount of information is available, this *probabilistic information* is mathematically represented by a vector in a Hilbert space $\mathcal{H}$ that forms a *state space.*

Rule 2. The *observables* (i.e., the physical quantities) of the system are mathematically represented by self-adjoint operators that act on the Hilbert space $\mathcal{H}$.

Rule 3. (i) The only possible *result of measuring an observable* A is one of the eigenvalues of the self-adjoint operator $\hat{A}$ that represents it. The probability that a measurement of A will yield a particular eigenvalue a_k corresponding the eigenvector $|a_k\rangle$, $\Pr(A = a_k)$, is

$$\Pr(A = a_k) = |\langle a_k|\psi\rangle|^2 = |c_k|^2,$$

where $|\psi\rangle = \sum_{i=1}^{\infty} c_i|a_i\rangle$.

(ii) If an observable A and a quantum state are represented by the self-adjoint operator $\hat{A}$ and the normalized vector $|a\rangle$, respectively, then, the *expected results* $\langle A\rangle_a$ of measuring A is

$$\langle A\rangle_a = \langle a|\hat{A}|a\rangle.$$

Rule 4. In an isolated system (i.e., in the absence of any external influence), the state vector $|\psi_t\rangle$ changes in time t according to the *time-dependent Schrödinger wave equation*

$$i\hbar\frac{d|\psi_t\rangle}{dt} = \hat{H}|\psi_t\rangle, \tag{A.1}$$

where $\hat{H}$ is called a *Hamiltonian operator* (and also a self-adjoint operator).

The fourth rule describes how a quantum state evolves in time. Since the Schrödinger wave equation is linear, $\alpha|\psi_{t_0}\rangle + \beta|\varphi_{t_0}\rangle$ evolves into $\alpha|\psi_t\rangle + \beta|\varphi_t\rangle$ during time $t - t_0$. Further, since the equation is also a first-order differential equation in time, the state at any given time t_0 determines the state at any later time t uniquely. When $|\psi_{t_0}\rangle$ and $|\psi_t\rangle$ are the state in time t_0 and t, respectively. then,

$$|\psi_t\rangle = \hat{U}(t, t_0)|\psi_{t_0}\rangle, \tag{A.2}$$

where

$$\hat{U}(t, t_0) = e^{-\frac{i}{\hbar}\hat{H}(t-t_0)},$$

and this operator is a unitary operator called a *time evolution operator.*

Bibliography

[1] Y. Aharonov, J. Anandan, and L. Vaidman, "Meaning of the wave function", *Phys. Rev.*, **A 47**, pp. 4616-4626, 1993.

[2] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, "RSA and Rabin functions: certain parts are as hard as the whole", *SIAM J. Comput.*, **17**, pp. 194-209, 1988.

[3] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural complexity I*, 2nd ed., Springer-Verlag, Berlin, 1995.

[4] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. W. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation", *Phys. Rev.*, **A 52**, pp. 3457-3467, 1995.

[5] A. Barenco, D. Deutsch, A. Ekert, and Jozsa, "Conditional quantum dynamics and logic gates", *Phys. Rev. Lett.*, **74**, pp. 4083-4086, 1995.

[6] A. Barenco, "A universal two-bit gate for quantum computation", *Proc. R. Soc. London*, **A 449**, pp. 679-683, 1995.

[7] A. Barenco, A. Ekert, K.-A. Suominen, and P. Törmä, "Approximate quantum Fourier transform and decoherence", *Phys. Rev.*, **A 54**, pp. 139-146, 1996.

[8] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill, " Efficient networks for quantum factoring", *Phys. Rev.*, **A 54**, pp. 1034-1063, 1996.

[9] P. A. Benioff, "The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines", *J. Stat. Phys.*, **22**, pp. 563-591, 1980.

[10] P. A. Benioff, "Quantum mechanical Hamiltonian models of discrete processes", *J. Math. Phys.*, **22**, pp. 495-507, 1981.

[11] P. A. Benioff, "Quantum mechanical Hamiltonian models of discrete processes that erase their own histories: application to Turing machines", *Int. J. Theor. Phys.*, **21**, pp. 177-201, 1982.

[12] P. A. Benioff, "Quantum mechanical Hamiltonian models of Turing machines", *J. Stat. Phys.*, **29**, pp. 515-546, 1982.

[13] P. A. Benioff, "Quantum mechanical models of Turing machines that dissipate no energy", *Phys. Rev. Lett.*, **48**, pp. 1581-1585, 1982.

[14] P. A. Benioff, "Quantum mechanical Hamiltonian models of computers", in: *New Techniques and Ideas in Quantum Measurement Theory*, D. M. Greenberger(ed.), Annals New York Academy of Sciences, **480**, pp. 475-486, 1986.

[15] C. H. Bennett, "Logical reversibility of computation", *IBM J. Res. Dev.*, **17**, pp. 525-532, 1973.

[16] C. H. Bennett, "The thermodynamics of computation–a review", *Int. J. Theor. Phys.*, **21**, pp. 905-940, 1982.

[17] C. H. Bennett, "Notes on the history of reversible computation", *IBM J. Res. Dev.*, **32**, pp. 16-23, 1988.

[18] C. H. Bennett, "Time/space trade-offs for reversible computation", *SIAM J. Comput.*, **18**, pp. 766-776, 1989.

[19] C. H. Bennett, "How to define complexity in physics, and why", in: *Complexity, Entropy and The Physics of Information*, W. H. Zurek(ed.), Addison-Wesley, Redwood City, pp. 137-148, 1990.

[20] C. H. Bennett, P. Gács, M. Li, P. M. B. Vitányi, and W. H. Zurek, "Thermodynamics of computation and information distance", in: *Proc. of the 25th ACM Symp. on Theory of Computing*, pp. 21-30, 1993.

[21] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed state entanglement and quantum error correction", submitted to Phys. Rev. A (available at lanl e-print quant-ph/9604024).

[22] C. H. Bennett, E. Bernstein, G. Brassard, and U. V. Vazirani, "Strength and weaknesses of quantum computing", *SIAM J. Comput.*, to appear (available at lanl e-print quant-ph/9701001).

[23] E. Bernstein and U. V. Vazirani, "Quantum complexity theory", in: *Proc. of the 25th ACM Symp. on Theory of Computing*, pp. 11-20, 1993.

[24] A. Berthaume and G. Brassard, "Oracle quantum computing", in: *Proc. of the Second Workshop on Physics and Computation*, pp. 195-199, 1992.

[25] A. Berthaume and G. Brassard, "The quantum challenge to structural complexity theory", in: *Proc. of the seventh Annual IEEE Conf. on Structure in Complexity*, pp. 132-137, 1992.

[26] A. Berthaume, D. Deutsch, and R. Jozsa, "The stabilisation of quantum computations", in: *Proc. of the Third Workshop on Physics and Computation*, pp. 60-62, 1994.

[27] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits", *SIAM J. Comput.*, **13**, pp. 850-864, 1984.

[28] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator", *SIAM J. Comput.*, **15**, pp. 364-383, 1986.

[29] D. Boneh and R. J. Lipton, "Quantum cryptanalysis of hidden linear functions", *Lecture Notes in Computer Science*, **963**, Springer-Verlag, Berlin, pp. 424-437, 1995.

[30] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching", at the Fourth Workshop on Physics and Computation, Boston, USA, Nov., 1996, to appear (available at lanl e-print quant-ph/9605034).

[31] A. R. Calderbank and P. W. Shor, " Good quantum error-correcting codes exist", *Phys. Rev.*, **A 54**, pp. 1098-1105, 1996.

[32] I. L. Chuang, R. Laflamme, P. W. Shor, and W. H. Zurek, "Quantum computers, factoring, and decoherence", *Science*, **270**, pp. 1633-1635, 1995.

[33] I. L. Chuang and Y. Yamamoto, "Simple quantum computer", *Phys. Rev.*, **A 52**, pp. 3489-3496, 1995.

[34] I. L. Chuang, R. Laflamme, J.-P. Paz, and Y. Yamamoto, "Effects of loss and decoherence on a simple quantum computer", manuscript, 1996 (available at lanl e-print quant-ph/9602018).

[35] I. L. Chuang and Y. Yamamoto, "Quantum bit regeneration", *Phys. Rev. Lett.*, **76**, pp. 4281-4284, 1996.

[36] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions", *Phys. Rev. Lett.*, **74**, pp. 4091-4094, 1995.

[37] K. L. Clarkson and P. W. Shor, "Applications of random sampling in computational geometry, II", *Discrete and Computational Geometry*, **4**, pp. 387-421, 1989.

[38] L. Debnath and P. Mikusiński, *Introduction to Hilbert spaces with applications*, Academic Press, San Diego, 1990.

[39] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer", *Proc. R. Soc. London*, **A 400**, pp. 97-117, 1985.

[40] D. Deutsch, "Quantum computational networks", *Proc. R. Soc. London*, **A 425**, pp. 73-90, 1989.

[41] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation", *Proc. R. Soc. London*, **A 439**, pp. 553-558, 1992.

[42] D. Deutsch, A. Barenco, and A. Ekert, "Universality in quantum computation", *Proc. R. Soc. London*, **A 449**, pp. 669-677, 1995.

[43] P. A. M. Dirac, *The principles of quantum mechanics*, 4th ed., Oxford Univ. Press, Glasgow, 1958.

[44] D. P. DiVincenzo and J. A. Smolin, "Results on two-bit gate design for quantum computers", in: *Proc. of the Third Workshop on Physics and Computation*, pp. 14-19, 1994.

[45] D. P. DiVincenzo, "Tow-bit gates are universal for quantum computation", *Phys. Rev.*, **A 51**, PP. 1015-1022, 1995.

[46] D. P. DiVincenzo and P. W. Shor, "Fault tolerant error correction with efficient quantum codes", *Phys. Rev. Lett.*, **77**, PP. 3260-3263, 1996.

[47] N. Dunford and J. T. Schwartz, *Linear operators*, I, II, and III, John Wiley & Sons, New York, 1957, 1963, and 1971.

[48] C. Dürr and M. Santha, "A decision procedure for well-formed linear quantum cellular automata", in: *Proc. of the 37th Ann. Symp. on Theoretical Aspects of Computer Science*, pp. 281-292, 1996.

[49] C. Dürr and M. Santha, "A decision procedure for unitary linear quantum cellular automata", in: *Proc. of the 37th Ann. Symp. on Foundations of Computer Science*, 1996, to appear.

[50] C. Dürr and P. Hϕyer, "A quantum algorithm for finding the minimum", manuscript, 1996(lanl e-print quant-ph/9607014).

[51] H. Edelsbrunner, *Algorithms in combinatorial geometry*, Springer-Verlag, Berlin, 1987.

[52] A. Ekert and C. Macchiavello, "Quantum error correction for communication", *Phys. Rev. Lett.*, **77**, pp. 2585-2588, 1996.

[53] A. Ekert and R. Jozsa, "Quantum computation and Shor's factoring algorithm", *Rev. Mod. Phys.*, **68**, pp. 733-753 , 1996.

[54] R. P. Feynman, " Simulating physics with computers", *Int. J. Theor. Phys.*, **21**, pp. 467-488, 1982.

[55] R. P. Feynman, " Quantum mechanical computers", *Foundation of Physics*, **16**, pp. 507-531, 1986.

[56] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, New York, 1979.

[57] A. Garg, "Decoherence in ion trap quantum computers", *Phys. Rev. Lett.* **77**, pp. 964-967, 1996.

[58] J. Gill, "Computational complexity of probabilistic Turing machines", *SIAM J. Comput.*, **6**, pp. 675-695, 1977.

[59] D. Gottesman, "Class of quantum error-correcting codes saturating the quantum Hamming bound", *Phys. Rev.*, **A 54**, pp. 1862-1868, 1996.

[60] L. K. Grover, "A fast quantum mechanical algorithm for database search", in: *Proc. of the 28th ACM Symp. on Theory of Computing*, pp. 212-219, 1996.

[61] J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, 1979.

[62] R. J. Hughes, D. F. V. James, E. H. Knill, R. Laflamme, and A. G. Petschek, "Decoherence bounds on quantum computation with trapped ions", *Phys. Rev. Lett.* **77**, pp. 3240-3243, 1996.

[63] C. J. Isham, *Lectures on quantum theory*, Imperial College Press, London, 1995.

[64] T. Jian, "An $O(2^{0.304n})$ algorithm for solving maximum independent set problem", *IEEE Trans. on Computers*, **C-35**, pp. 847-851, 1986.

[65] D. S. Johnson, "A catalog of complexity classes", in: *Handbook of Theoretical Computer Science Vol. A: Algorithms and Complexity*, J. van Leeuwen(ed.), MIT Press, Cambridge, pp. 67-161, 1990.

[66] R. Jozsa, "Characterizing classes of functions computable by quantum parallelism", *Proc. R. Soc. London*, **A 435**, pp. 563-574, 1991.

[67] R. Jozsa and B. Schumacher, "A New proof of the quantum noiseless coding theorem", *J. Mod. Opt.*, **41**, pp. 2343-2349, 1994.

[68] A. Y. Kitaev, "Quantum measurement and the Abelian stabilizer", manuscript, 1995 (available at lanl e-print quant-ph/9511026).

[69] D. E. Knuth, *The art of computer programming. Vol. III: Sorting and searching*, Addison-Wesley, Reading, 1973.

[70] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, "Perfect quantum error correction code", *Phys. Rev. Lett.*, **77**, pp. 198-201, 1996.

[71] R. Landauer, " Computation and physics: Wheeler's meaning circuits?", *Foundation of Physics*, **16**, pp. 551-564, 1986.

[72] R. Landauer, " Dissipation and noise immunity in computation and communication", *Nature*, **335**, pp. 779-784, 1988.

[73] R. Landauer, " Reversible computation: implications for measurement, Communication, and Physical Law", in: *Proc. of the 3rd Int. Symp. Foundation of Quantum Mechanics*, pp. 407-411, 1989.

[74] R. Landauer, " Information is physics", *Physics Today*, pp. 23-29, May 1991.

[75] R. Landauer, " Is quantum mechanics useful?", *Phil. Trans. R. Soc. London*, **A 353**, pp. 367-376, 1995.

[76] R. Y. Levine and A. T. Sherman, "A note on Bennett's time/space trade-offs for reversible computation", *SIAM J. Comput.*, **19**, pp. 673-677, 1990.

[77] S. Lloyd, "A potentially realizable quantum computer", *Science*, **261**, pp. 1569-1571, 1993.

[78] S. Lloyd, "Envisioning a quantum supercomputer", *Science*, **263**, p. 695, 1994.

[79] S. Lloyd, "Almost any quantum logic gate is universal", *Phys. Rev. Lett.*, **75**, pp. 346-349, 1995.

[80] A. Messiah, *Mécanique Quantique*, Dunod, Paris, 1959.

[81] T. Mihara and T. Nishino, "Quantum computation and NP-complete problems", *Lecture Notes in Computer Science*, **834**, Springer-Verlag, Berlin, pp. 387-395, 1994.

[82] T. Mihara and T. Nishino, "On a method of solving SAT efficiently using the quantum Turing machine", in: *Proc. of the Third Workshop on Physics and Computation*, pp. 177-185, 1994.

[83] T. Mihara, "Are measurements effective on quantum computation?", *IEICE Trans. Inf. & Sys.*, **E79-D**, pp. 382-384, 1996.

[84] T. Mihara, "On the complexity of finding cycles in periodic functions using the quantum Turing machine", *IEICE Trans. Inf. & Sys.*, **E79-D**, pp. 579-583, 1996.

[85] T. Mihara, "An extended quantum search algorithm", submitted to *IEICE Trans. Inf. & Sys.*.

[86] C. Miquel, J. P. Paz, and R. Perazzo, "Factoring in a dissipative quantum computer", manuscript, 1996 (available at lanl e-print quant-ph/9601021).

[87] B. Monien and E. Speckenmeyer, "Solving satisfiability in less than 2^n steps", *Discrete App. Math.*, **10**, pp. 287-295, 1985.

[88] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland, "Demonstration of a fundamental quantum logic gate", *Phys. Rev. Lett.*, **75**, pp. 4714-4717, 1995.

[89] K. Morita and M. Harao, "Computation universality of one-dimensional reversible (injective) cellular automata", *Trans. of IEICE*, **E72**, pp. 758-762, 1989.

[90] G. M. Palma, K.-A. Suominen, and A. Ekert, "Quantum computers and dissipation", *Proc. R. Soc. London*, **A 452**, pp. 567-584, 1996.

[91] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, Reading, 1994.

[92] T. Pellizzari, S. A. Gardiner, J. I. Cirac, and P. Zoller, "Decoherence, continuous observation, and quantum computing: A cavity QED model", *Phys. Rev. Lett.*, **75**, pp. 3788-3791, 1995.

[93] A. Peres, "Reversible logic and quantum computers", *Phys. Rev.*, **A 32**, pp. 3266-3276, 1985.

[94] A. Peres, *Quantum theory: concept and methods*, Kluwer Academic, Dordrecht, 1993.

[95] M. B. Plenio and P. L. Knight, "Realistic lower bounds for the factorization time of large numbers on a quantum computer", *Phys. Rev.*, **A 53**, pp. 2986-2990, 1996.

[96] M. B. Plenio, V. Vedral, and P. L. Knight, "Optimal realistic quantum error correction code", manuscript, 1996 (available at lanl e-print quant-ph/9603022).

[97] M. B. Plenio and P. L. Knight, "Decoherence limits to quantum computation using trapped ions", manuscript, 1996 (available at lanl e-print quant-ph/9610015).

[98] F. P. Preparata and M. I. Shamos, *Computational geometry*, Springer-Verlag, Berlin, 1985.

[99] R. L. Rivest, "Cryptography", in *Handbook of Theoretical Computer Science Vol. A: Algorithms and Complexity*, J. van Leeuwen(ed.), MIT Press, Cambridge, pp. 717-755, 1990.

[100] J. J. Sakurai, *Modern quantum mechanics*, Benjamin/Cummings, Reading, 1985.

[101] L. I. Schiff, *Quantum mechanics*, 3rd ed., McGraw-Hill, New York, 1968.

[102] F. Schipp, W. R. Wade, and P. Simon, *Walsh series: an introduction to dyadic harmonic analysis*, Adam Hilger, Bristol, 1990.

[103] R. Schroeppel and A. Shamir, "A T=O($2^{n/2}$), S=O($2^{n/4}$) algorithm for certain NP-complete problems", *SIAM J. Comput.*, **10**, pp. 456-464, 1981.

[104] B. Schumacher, "Quantum coding", *Phys. Rev.*, **A 51**, pp. 2738-2747, 1995.

[105] B. Schumacher and M. A. Nielsen, "Quantum data processing and error correction", manuscript, 1996 (available at lanl e-print quant-ph/9604022).

[106] R. Sedgewick, T. G. Szymanski, and A. C. Yao, "The complexity of finding cycles in periodic functions", *SIAM J. Comput.*, **11**, pp. 376-390, 1982.

[107] R. Seidel, "Backwards analysis of randomized geometric algorithms", in: *New trends in discrete and computational geometry*, J. Pach(ed.), Springer-Verlag, Berlin, pp. 37-67, 1993.

[108] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring", in: *Proc. of the 35th Ann. Symp. on Foundations of Computer Science*, pp. 124-134, 1994.

[109] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory", *Phys. Rev.*, **A 52**, pp. R2493-R2496, 1995.

[110] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. Comput.*, to appear (available at lanl e-print quant-ph/9508027).

[111] P. W. Shor and J. A. Smolin, "Quantum error-correcting codes need not completely reveal the error syndrome", submitted to Phys. Rev. Lett. (available at lanl e-print quant-ph/9604006).

[112] P. W. Shor, "Fault-tolerant quantum computation", in: *Proc. of the 37th Ann. Symp. on Foundations of Computer Science*, Nov., 1996, to appear (available at lanl e-print quant-ph/9605011).

[113] D. R. Simon, "On the power of quantum computation", in: *Proc. of the 35th Ann. Symp. on Foundations of Computer Science*, pp. 116-123, 1994.

[114] T. Sleator and H. Weinfurter, "Realizable universal quantum logic gates", *Phys. Rev. Lett.*, **74**, pp. 4087-4090, 1995.

[115] J. A. Smolin and D. P. DiVincenzo, "Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate", *Phys. Rev.*, **A 53**, pp. 2855-2856, 1996.

[116] A. Steane, "Multiple particle interference and quantum error correction", *Proc. R. Soc. Lond.*, to appear (available at lanl e-print quant-ph/9601029).

[117] A. Steane, "Simple quantum error correcting codes", manuscript, 1996 (available at lanl e-print quant-ph/9605021).

[118] R. E. Tarjan and A. E. Trojanowski, "Finding a maximum independent set", *SIAM J. Comput.*, **6**, pp. 537-546, 1977.

[119] T. Toffoli, "Bicontinuous extensions of invertible combinatorial functions", *Math. Syst. Theor.*, **14**, pp. 13-23, 1981.

[120] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimber, "Measurement of conditional phase shifts for quantum logic", *Phys. Rev. Lett.*, **75**, pp. 4710-4713, 1995.

[121] W. G. Unruh, "Maintaining coherence in quantum computers", *Phys. Rev.*, **A 51**, pp. 992-997, 1995.

[122] L. Vaidman, L. Goldenberg, and S. Wiesner, "Error prevention scheme with four particles", manuscript, 1996 (available at lanl e-print quant-ph/9603031).

[123] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations", *Phys. Rev.*, **A 54**, pp. 147-153, 1996.

[124] J. Vyskoč, "An O($n^{\lg k} \cdot 2^{n/2}$) time and O($k \cdot 2^{n/4}$) space algorithm for certain NP-complete problems", *Theor. Comput. Sci.*, **51**, pp. 221-274, 1987.

[125] J. Watrous, "On one-dimensional quantum cellular automata", in: *Proc. of the 36th Ann. Symp. on Foundations of Computer Science*, pp. 528-537, 1995.

[126] A. C. Yao, "On constructing minimum spanning trees in k-dimensional spaces and related problems", *SIAM J. Comput.*, **11**, pp. 721-736, 1982.

[127] A. C. Yao, "Quantum circuit complexity", in: *Proc. of the 34th Ann. Symp. on Foundations of Computer Science*, pp. 352-361, 1993.

[128] W. H. Zurek, "Reversibility and stability of information processing systems", *Phys. Rev. Lett.*, **53**, pp. 391-394, 1984.

[129] W. H. Zurek, "Thermodynamic cost of computation, algorithmic complexity and the information metric", *Nature*, **341**, pp. 119-124, 1989.

[130] W. H. Zurek, "Algorithmic information content, Church-Turing thesis, physical entropy, and Maxwell's demon", in: *Complexity, Entropy and The Physics of Information*, W. H. Zurek(ed.), Addison-Wesley, Redwood City, pp. 73-89, 1990.

Publications

[1] 三原孝志, 西野哲朗, “量子コンピュータについて (A brief survey)”, 93 年夏の LA シンポジウム, 宮城県, 1993 年 7 月 (情報基礎理論ワークショップ論文集, pp. 106-111, 1993).

[2] 三原孝志, 西野哲朗, “量子コンピュータを用いた NP 完全問題の多項式時間解法”, JAIST Research Report IS-RR-93-0012F, JAIST, 1993.

[3] 三原孝志, 西野哲朗, “量子 Turing 機械による NP 完全問題の多項式時間解法について”, 94 年冬の LA シンポジウム, 京都大学数理解析研究所, 1994 年 2 月 (京都大学数理解析研究所講究録 871 計算量理論, pp. 30-36, 1994).

[4] 三原孝志, 西野哲朗, “万能量子 Turing 機械を用いた NP 完全問題の多項式時間解法について”, 電子情報通信学会コンピュテーション研究会資料 COMP-94-5, pp. 41-50, 1994.

[5] 三原孝志, 西野哲朗, “量子 Turing 機械による SAT の効率的解法について”, 94 年夏の LA シンポジウム, 長野県, 1994 年 7 月 (情報基礎理論ワークショップ論文集, pp. 57-60, 1994).

[6] T. Mihara and T. Nishino, “Quantum computation and NP-complete problems”, *at the Fifth Ann. Int. Symp. on Algorithms and Computation*, Aug. 1994 (*Lecture Notes in Computer Science*, **834**, Springer-Verlag, Berlin, pp. 387-395, 1994).

[7] T. Mihara and T. Nishino, “Interpretations of the Quantum Theory and NP-Complete Problems”, JAIST Research Report IS-RR-94-0025F, JAIST, 1994.

[8] T. Mihara and T. Nishino, “On a method of solving SAT efficiently using the quantum Turing machine”, in: *Proc. of the Third Workshop on Physics and Computation*, pp. 177-185, Nov. 1994.

[9] T. Mihara, “On the Computational Power of Quantum Turing Machine”, 96 年冬の LA シンポジウム, 京都大学数理解析研究所, 1996 年 1 月 (京都大学数理解析研究所講究録 950 計算モデルと計算の複雑さに関する研究, pp. 33-38, 1996).

[10] T. Mihara, “Are measurements effective on quantum computation?”, *IEICE Trans. Inf. & Sys.*, **E79-D**, pp. 382-384, 1996.

[11] T. Mihara, "On the complexity of finding cycles in periodic functions using the quantum Turing machine", *IEICE Trans. Inf. & Sys.*, **E79-D**, pp. 579-583, 1996.