The Role of Interference and Entanglement in Quantum Computing.

Jurgen Van Gael

May 17, 2005

Preface

Prelude

The year is 1900. The world's most renowned mathematician, David Hilbert, proposes 10 challenging mathematical problems for the nascent century at the second international congress in Paris. One of these problems, the *Entscheidungsproblem*, asks if there exists an algorithm that decides whether a first-order statement is derivable from the axioms of first-order predicate calculus. After more than 25 years, a young mathematician from Cambridge, Alan Turing, discovers a solution. Moreover, a solution that will inspire researchers, several generations later, to invent a revolutionary computational model based on quantum mechanics.

In order to solve the Entscheidungsproblem, Alan Turing formalized the notion of an algorithm. His approach considered which calculations could be performed by an idealized human computor[1], a computor who follows simple rules and is not limited in time and space. Starting from the fundamental observations of physical reality that operations must be easily implementable and only make local changes, he defined the *Turing machine* and formulated a thesis which is currently known as the *Church-Turing thesis*[2]:

> The class of functions computable by a Turing machine correspond exactly to the class of functions which we would naturally regard as computable by an algorithm.

The Church-Turing thesis is a connection between the mathematical notion of an algorithm and the intuitive notion of what is thought of as an algorithm. The Church-Turing thesis and Turing machines allowed researchers in the '40's and 50's to create a theory of computability - what is computable and what not. When computability theory was mature, starting from the '60s, researchers started to characterize the resources needed to solve a problem, giving birth to the active research area called complexity theory. An important concept from complexity theory that recurs in this thesis is the notion of an *efficient* computation: an algorithm that decides every possible input in time polynomial in the size of the input. Since the invention of Turing machines, many computer scientists and mathematicians devised other computational models such as boolean circuits, random access machines, . . . and noticed that all these models have the exact same computational power as Turing machines, thereby supporting the idea of the Turing machine as a powerful and robust model of computation. Moreover, all these models were proved to be *polynomially* equivalent to each other and Turing machines meaning that the different computational models could efficiently simulate one another. Only *probabilistic Turing machines* seemed to challenge the Church-Turing thesis since efficient randomized algorithms

[1] The term computor was used for people who calculated in the absence of calculating machines.

[2] Alonzo Church approached the Entscheidingsproblem from a different angle. Gödel, among other people, challenged Church's assumptions and only after Turing showed his solution to be equivalent to Church's solution, Gödel was convinced of its applicability.

were discovered for problems which didn't seem to allow an efficient solution on a deterministic Turing machine[3]. A relatively easy modification of the thesis is thus believed to apply: the *Strong Church-Turing thesis*:

> Any model of computation can be simulated on a probabilistic Turing machine with at most a polynomial increase in the number of elementary operations.

What is interesting for our story is that Turing machines have close correspondence to physical reality. We will see that researchers at the end of the twentieth century started from more fundamental physical observations in order to create a model of computation that is believed to be even more powerful than a probabilistic Turing machine, a model of computation based on quantum mechanics. From now on, the term classical computation will refer to probabilistic Turing machines. The next section summarizes some ideas from classical computation in order to contrast them with results from quantum computation.

Computer Science before 1994

The first problem we want to review is *searching* as it probably is *the* most important concept for computer science. The structure of the search space dictates the speed with which a solution can be found. For example, it is much easier to look up a phone number in a phone book given someone's name than it is the other way around; clearly this is because the phone book is sorted alphabetically by name whilst the phone numbers are more or less randomly distributed throughout the phone book. Let's formalize the search problem a little further. A database with one marked item, x, is represented by a function $f_x : \mathbb{F}_2^n \to \mathbb{F}_2$ such that $\forall a \in \mathbb{F}_2^n$:

$$f_x(a) = \begin{cases} 0 & a \neq x \\ 1 & a = x \end{cases}$$

We define a *query* or database access as a function call which given a number $a \in \mathbb{F}_2^n$ returns $f_x(a)$. We are asked to develop an algorithm that finds x in as few queries as possible. Note that $|\mathbb{F}_2^n| = 2^n$.

If we assume that every element in $\mathbb{F}_2^n$ is as likely to be the marked item, it is safe to call this an unstructured search space. Intuitively, knowledge of the function value of an element $a \neq x$ gives us no information whatsoever to guide our next queries. Every deterministic algorithm, one which cannot use a random source of bits, will be programmed to run through all elements of $\mathbb{F}_2^n$ in a particular order to find x. It is clear that in the worst case x could be the last element of the programmed order forcing the algorithm to make 2^n queries to find x in the database. On average the algorithm will query the database 2^{n-1} times to find x. An interesting question would be to ask whether querying at random can buy us anything? An algorithm that can sample $a \in \mathbb{F}_2^n$ from a uniform distribution has a probability of $\frac{1}{2^n}$ to find the correct element. In order to increase the probability of success to a constant we must sample multiple times. Let the stochastic variable T denote the number of times we have to query the database in order to find x. The distribution that applies to this problem is the geometric distribution[4] with a success probability of $\frac{1}{2^n}$. We thus expect to query the database $\frac{1}{\frac{1}{2^n}} = 2^n$ times. If we

[3]Recent research suggests that deterministic polynomial time algorithms are able to simulate randomized polynomial time algorithms.

[4]The geometric distribution tells us how many times (T) we expect to toss a coin before we get heads given that the success probability for a single coin toss is p: $E[T] = \frac{1}{p}$

would sample from a nonuniform distribution, the probability of success would be greater than $\frac{1}{2^n}$ for some elements but smaller for others. We therefore cannot get a *worst-case* speedup by sampling from a nonuniform distribution. This analysis intuitively suggests that searching an unstructured search space cannot be done faster than exhaustively querying all elements. A slightly more involved argument can prove that no possible algorithm can search the space in less than $\Theta(2^n)$ queries.

Another problem, one which mathematicians have been struggling with for hundreds of years, is the problem of *factoring*. Given a number n, can one efficiently find numbers p, q such that $n = pq$? The apparent infeasibility of this problem is the basis for numerous cryptographic algorithms. Proving an exponential lower bound on the running time appears to be difficult and no proof is expected soon. This leaves us with some probabilistic algorithms, based on deep number theoretical insights, which unfortunately have an exponential running time: $\Theta(e^{n^{1/3}\log^{2/3}(n)})$ for the fastest algorithm ...

... that is until Peter Shor's discovery of 1994 ...

Enter ... 'The Quantum'

Assuming the strong Church-Turing thesis holds, if we can prove that a problem cannot be solved using polynomial resources on a probabilistic Turing machine, we know that it cannot be solved using polynomial resources on *any* computing device. The discovery of a factorization algorithm for quantum computers with a running time of $\Theta(n^2 \log(n) \log\log(n))$ must come as a rather big surprise then. It means that exponentially less elementary operations are required in order to find a factor for a given input on this new computational model. In 1995, it was also discovered that unstructured search on a quantum computer has a query complexity of $\Theta(\sqrt{2^n})$; this means that the quantum algorithm will return the desired element in significantly fewer queries than any classical algorithm. Although the speedup for the former discovery is spectacular, especially the latter discovery clashes with common sense.

Acknowledgements

Education is an admirable thing, but it is well to remember from time to time that nothing that is worth knowing can be taught.
-Oscar Wilde

Many people have been part of my life as a student and it is my pleasure to sincerely thank every one of them for making my graduation possible.

First of all, it is difficult to overstate my gratitude to Professor Dieter van Melkebeek. His unlimited efforts made it possible to visit the university of Wisconsin. When I lost myself in the complexity of the subject, Professor van Melkebeek's advice, encouragement and inspiration made it all clear and simple again. Finally, I am indebted to him for his patience during the past few weeks when he carefully proofread this text. I am thankful for his good company and I am looking forward to work with him in the future.

I also want to thank the university of Wisconsin, everyone at the computer sciences department and the people from the Qubic workgroup for giving me the opportunity to join them for a semester. I especially enjoyed the many quantum and non-quantum converstations with my office mates, Bess, Chris and Scott.

Next I would like to thank Professor Piessens and Professor Van Assche for their support on the home front. I am indebted to the people at AVNet for their equipment and technical support which made communicating between Leuven and Madison so easy.

I am very grateful to my parents without whom none of my adventures would have been possible. The knowledge that they will always be there to pick up the pieces is what allowed me to repeatedly risk getting shattered.

Finally, many thanks go to my girlfriend Karlien for her infinite patience while I was in Madison. At every step of the way I was showered with love and support. Thanks for putting up with all of this!

Contents

Chapter 1

Approach and Overview of the Thesis

Our prelude has been imprecise in several ways. What does it mean for a quantum computer to return a certain value? What are the elementary operations? What is a *quantum model of computation* and most importantly, how is this model different from the classical model of computation? The next chapters of this thesis will answer these questions in detail. The last issue mentioned above eventually dictates the theme of the thesis: what is the nature of the speedup of the quantum model of computation.

Since the discovery of different quantum algorithms there has been doubt about the nature of their speedup over their classical counterparts. Different quantum information scientists have different opinions but almost all agree that the answer must be sought by studying two phenomena that make quantum computers different from classical computers: interference and entanglement. Understanding the mechanism that causes quantum computational speedup is important for two reasons. First of all, we want to be able to theoretically characterize the power of quantum computers; secondly, our insights might prove useful in developing about new quantum algorithms.

Quantum computing is not a standard ingredient of the undergraduate curriculum. My first goal for the thesis was to master the quantum mechanical formalism and understand the underlying mathematical principles. A next step was to work through the basics of quantum computation, sometimes touching on the field of quantum information theory. The standard reference [28] was an excellent introduction. If not mentioned otherwise, the basic results from quantum computing come from [28, 26, 24, 18, 22]. Because the main question of the thesis is still an unsolved problem and many interesting results were discovered during the past few years, a second goal of the thesis was to acquaint myself with current research. The internet arXiv [2] and recent journals provided all the necessary articles to accomplish this goal. The last goal was to find an original approach to the topic and recognize open problems and possible approaches for solving them.

Quantum computing is a technical subject. Because it is an interdisciplinary science between theoretical physics and theoretical computer science, it takes some effort to understand the concepts and issues of both fields. Some technical lemma's and proofs are necessary in order to explain the intricacies of quantum computing. Nonetheless, as I learned that concepts and intuition are more important than technical details, I have tried to write a text that reflects this belief.

The main matter of the thesis opens with a historic overview that led to the discovery of quantum computing. Next up is a brief introduction to the concepts of quantum mechanics and how they apply to quantum computing. I found that working out the exercises in [28, 24] proved invaluable to understanding the rich structure of quantum mechanics. Nonetheless, I hope the simple examples which are worked out in this thesis give some feel of the mathematical structure of quantum mechanics. After the introduction, we build a formal model for quantum computers based on classical Boolean circuits. Finally, we end the chapter with a demonstration of the capabilities of quantum computers by explaining the Deutsch-Jozsa problem. This preliminary chapter is not a general introduction to the theory of quantum computing: from the start it tries to give the right intuition to understand our conclusions.

The third chapter is complementary to the second in that it intuitively explains the nature of different phenomena in quantum computing. We first show how interference is somewhat similar to probabilistic states. However, the use of complex amplitudes in quantum mechanics allows for destructive interference while probabilistic states are limited to constructive interference. Next, we show how superposition is not that special for quantum computing. Many researchers have argued that creating a superposition over an exponential number of states and then computing a function corresponds to some sort of massive parallelism. Unfortunately, this sort of parallelism is known for probabilistic computing too. We therefore discard quantum parallelism as a source of the speedup of quantum algorithms over classical algorithms. Next, we discuss entanglement and how it's discovery stunned the physics community. Different measurements on entangled states are dependent on each other in a very intricate way. This effect has been the source of many reductions in communication complexity for certain problems. It is therefore natural to regard entanglement as a possible source of the speedup of quantum algorithms over classical algorithms. We finally end the chapter by introducing the formalism from [25]. This formalism considers every classical computation as a path in a tree which has some probability of happening. It is clear from this formalism that once there is some probability that a certain decision is made, this probability is bound to exist until the algorithm accepts or rejects. On the other hand, it will be shown that every quantum computation has a complex probability amplitude. This means that if two computations with a positive and negative probability amplitude end in the same state, they destructively interfere and the probability amplitude of all other computational paths increases. What is interesting about this formalism is that the only difference between the computational models is the norm of the state vector they preserve.

In chapter 4 we apply all the learned concepts to discuss the two main algorithms: Grover's search algorithm and Shor's factoring algorithm. We start by deriving the complexity of Grover's search algorithm. Next, we generalize Grover's search algorithm which gives us the amplitude amplicitation algorithm. This algorithm amplifies the success probability of another algorithm faster than classical probability amplification can. We then apply amplitude amplification to a random walk algorithm for SAT. By doing this we introduce a technique which allows us to postpone any intermediate measurement to the end of the algorithm. This will bey a key step in our conclusion. Finally, we briefly discuss Shor's factoring algorithm. Because the details of it's complexity analysis are slightly more involved and not critical to our conclusion, we skip their formal proofs.

In chapter 5 we draw the main conclusion for this thesis by generalizing results from [34, 12]. First we note that every quantum mechanical system that is composed of n qubits is described by a Hilbert space that is a tensor product of n two dimensional Hilbert spaces. However, when we use the technique from section 4.2.2, which we formalize in section 5.1, we are able to introduce a new interpretation. In section 5.2, we discuss how every tensor product Hilbert space is isomorphic to a *simple* Hilbert space: a Hilbert space that is not a tensor product of several smaller Hilbert spaces. In this simple Hilbert space we can perform a full analysis of the algorithms and we only need interference. Entanglement cannot even be properly defined for a simple Hilbert space. Nonetheless, section 5.3 shows that only when we need to embed the simple Hilbert space in a real physical system, we need to decompose it in a tensor product of smaller Hilbert spaces. We heavily rely on the ideas in [12] which prove that it is impossible to implement a scalable quantum computer with quantum systems that aren't composite. In other words, our conclusion is that interference is the only phenomenon that is important for the analysis of quantum algorithms. Only when we design quantum computers we must be careful about how to handle entanglement. This suggests that from a computer science perspective – with the emphasis on algorithm analysis and design – we must acquaint ourselves with interference in order to discover and analyze new algorithms. The conlusion in chapter four is the most important contribution to the field of quantum computing in this text.

Alas, I believe that the conclusion from chapter 5 does little to quantitatively characterize the power of interference and more in general the power of quantum computation. Complexity researchers are ultimately interested where the power of quantum computation sits in the hierarchy of complexity classes. Therefore, our final chapter starts with section 6.1, discussing existing results in quantum complexity theory. Section 6.2 shows how to simulate a quantum computer with a classical computer. These simulation algorithms allow us to derive naive bounds on the power of BQP. Because a more detailed characterization of BQP seems more difficult, we continue to study certain *subsets* of general quantum computers for which we can characterize their power precisely. Section 6.3 introduce a new formalism to describe quantum state: the stabilizer formalism. Stabilizer circuits can use all but one gate from a very common universal quantum gate set. Nonetheless, this simplification makes the mathematical analysis of stabilizer circuits much easier and we prove that their power is equivalent to $\oplus\mathsf{L}$. Next, we study a second subset of general quantum computers in section 6.4: the p-blocked circuits. This subset is already more powerful than stabilizer circuits but it's mathematical analysis is less elegant. Finally, we end the thesis in section 6.5 with some open problems and suggests further research towards the resolution of the general problem of quantum computational power.

Before we get started we note that all chapters assume undergraduate level knowledge of the theory of computing; we refer to [27, 13] for introduction and reference. We use the standard complexity theoretic assumption that SAT requires exponential lower bounds which implies that $\mathsf{P} \neq \mathsf{NP}$. On the other hand, there exists an enormous philosophical debate on the interpretation of quantum mechanics. We intentionally avoid discussing any alternative interpretations of quantum mechanics and how they could relate to quantum computation so as to not lose ourselves in and endless and often unscientific argument.

Chapter 2

Preliminaries

2.1 Historical Perspective

The dawn of the twentieth century gave birth to several revolutions in physics. One of these revolutions, quantum mechanics, became the standard framework for describing matter on the smallest scales. The brightest minds in physics - Einstein, Schrödinger, Heisenberg, and many others - struggled for years to formalize and understand the theory of quantum mechanics; a theory that is fundamentally different from classical physics and counterintuitive in many ways. The applications of quantum mechanics are widespread and important, e.g., quantum mechanics is solely responsible for the micro-electronics that permeate our daily lives.

The story of quantum computation starts in the early eighties with physicist Richard Feynman. Richard Feynman grew an interest in computation and asked himself whether a classical computer can efficiently simulate physical reality: given an initial state and its evolution equations, can a computer efficiently predict what the final state of the physical system will be? When we limit physical reality to classical systems obeying Newton's equations, the answer is *in principle* yes: computers are able to calculate orbits of satellites, simulate aerodynamic flows, ... We say in principle as some systems are very sensitive to slight changes in initial condition and as computers only have a finite precision, there are some technical difficulties here. This phenomenon is known as chaos.

A positive answer to the question whether quantum systems could be simulated on a classical computer would be a tremendous discovery as chemists, pharmacists, biologists, to name a few, would be able to simulate experiments that involve risk, efficiently on a computer. Alas, the answer is no: a quantum state needs an exponential number of variables to be represented on a classical computer. Updating an exponential number of variables certainly takes an exponential amount of time. Therefore, an efficient simulation of a complete description of a quantum system is impossible. The interesting observation made by Richard Feynman states that the quantum system itself efficiently solves its own evolution equations! This is unfortunate for the chemist who wants to simulate a dangerous experiment. Nonetheless, if we could 'program' a difficult problem in a quantum evolution, we might be able to find a quantum system that solves it more efficiently than a classical computer. The search for such problems is on.

Around the same time as Richard Feynman's discoveries, a good friend of his, Edward Fredkin, showed that classical computers can be simulated using classical physics. Edward Fredkin, together with his student Tomasso Toffoli constructed the *billiard ball computer.* They showed how to simulate a Boolean logic circuit with billiard balls and carefully placed bouncing walls in a plane, modulo friction and chaos. What is remarkable about their achievement is how they worked around the time reversibility that is inherent in classical physics. Boolean logic circuits are not necessarily reversible: given the output to the circuit it is often impossible to tell what the input is. Still Fredkin and Toffoli managed to transform any possible Boolean logic circuit into a billiard ball computer. This discovery links classical physics to the strong Church-Turing thesis which we defined in the preface.

Summarizing, we find that classical physics can be simulated by Turing machines and Turing machines can be simulated by carefully designed classical experiments. This implies that the strong Church-Turing thesis is in some sense equivalent to classical physics. On the other hand, there exist physical realizable systems that transcend the realm of classical computation. In hindsight the obvious question to ask is whether there exists an even stronger form of the Church-Turing thesis that incorporates the power of quantum mechanics. David Deutsch started to investigate the fundamental question whether other physical theories would mean different computational models and found the first problems that can be programmed into quantum mechanics and be solved more efficiently [16].

A final development that stimulated research in quantum computing came from Moore's law. This law states that the number of transistors on a microchip doubles every eighteen months. Of course, the size of atoms limits this miniaturization process and if this trend continuous, one expects to reach the realm of quantum mechanics by the year 2020. Classical electromagnetic theory does not suffice to predict what happens to electric currents at this scale. Two typical quantum mechanical effects prohibit reliable micro-electronics to perform their work correctly. The first effect occurs when measuring the state of a quantum system: the measurement disturbs the state and possibly moves it to a different state. A computer accesses the value of a bit all the time and it is inconceivable that some of those measurements would randomly flip the bit. The second effect allows a quantum system to exist in several different states at once. It is meaningless for reliable computation to have a bit that is both 0 and 1 at the same time.

These theoretical and practical motivations led to the formulation of a computational model based on quantum mechanics. We will now briefly introduce the quantum mechanical principles and a model for classical computation. From there on, we can define a quantum model of computation and describe our first 'quantum algorithm'.

2.2 Quantum Mechanics

We refer to [28] for a full overview of quantum mechanics; our introduction treats quantum mechanics as a mathematical formalism. Although these principles might seem simple to the mathematician or computer scientist, the reader must remember that it took decades for many of the brightest minds and thousands of experiments to discover these principles and test them. There are an infinite amount of mathematical formalisms, some differing only very little from others, from which a physicist can choose to describe reality, but only very few give correct predictions. In this section we will simplify and abstract quantum mechanics and use four postulates that allow us to build a model for quantum computers.

Before we move on to the postulates of quantum mechanics we introduce a common and convenient notation for vectors: the Dirac *bra-ket* notation. A vector, say v, in this notation is represented by a *ket*, $|v\rangle$. The dual to this vector is represented by a *bra*, $\langle v|$. The scalar product of a vector and its dual is thus represented as $\langle v|v\rangle$ while their outer product corresponds to a matrix and is represented by $|v\rangle\langle v|$. A linear combination of two vectors is represented by $a|v\rangle + b|w\rangle$ while the tensor product of two vectors is both represented as $|v\rangle \otimes |w\rangle$ and $|vw\rangle$.

Back to quantum mechanics. The first postulate concerns the state in which physical systems can be in.

Postulate 2.1. *[28] Associated to any isolated physical system is a complex vector space with an inner product (a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which by convention is a unit vector in the state space.*

The simplest quantum system, analogue to the classical bit, is the 2-dimensional complex Hilbert space, commonly called the quantum bit or *qubit.* In order for us to describe different states, we always fix an orthogonal basis in this Hilbert space which we call the standard basis,

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The first postulate implies that any linear combination of basis vectors, itself a vector in Hilbert space, corresponds to a physically realizable state[1]. These linear combinations are called *superpositions* and in our example are of the form,

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \qquad \text{with } \alpha, \beta \in \mathbb{C}. \tag{2.1}$$

The condition that $|\Psi\rangle$ be a unit vector implies that $||\Psi\rangle| = \sqrt{\langle\Psi|\Psi\rangle} = 1$ or $|\alpha|^2 + |\beta|^2 = 1$. Note that because $\alpha, \beta \in \mathbb{C}$, a qubit's state has continuous values while the classical bit has only discrete values. We often call α, β the *amplitudes* of the corresponding *components*, $|0\rangle, |1\rangle$, of the state.

Example 2.1. Suppose we have a qubit in state,

$$\frac{|0\rangle - i|1\rangle}{\sqrt{2}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-i}{\sqrt{2}} \end{bmatrix}.$$

The $|0\rangle$ component has an amplitude of $\frac{1}{\sqrt{2}}$ while the $|1\rangle$ component has an amplitude of $\frac{-i}{\sqrt{2}}$. It is clear that,

$$\left|\frac{1}{\sqrt{2}}\right|^2 + \left|\frac{-i}{\sqrt{2}}\right|^2 = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{-i}{\sqrt{2}}\frac{i}{\sqrt{2}} = 1.$$

⊠

Our next postulate defines how we must approach composite quantum systems.

Postulate 2.2. *[28] The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have n systems, and system i is prepared in state $|\Psi_i\rangle$, then the joint state of the system is in a state $|\Psi_0\rangle \otimes |\Psi_1\rangle \otimes \cdots \otimes |\Psi_n\rangle$.*

Suppose we have n qubits, therefore there are 2^n basis states. Take two tensor products of n basis states, $|x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$ and $|y_1\rangle \otimes |y_2\rangle \otimes \cdots \otimes |y_n\rangle$ with $x_i, y_j \in \{0, 1\}$. Their scalar product is $\langle x_1|y_1\rangle \otimes \langle x_2|y_2\rangle \otimes \cdots \otimes \langle x_n|y_n\rangle$ so if any of the $x_i \neq y_i$, it is 0. This implies that the 2^n different tensor products of n basis states are all orthogonal. Therefore, the tensor product structure implies that the dimension of our composite Hilbert space is 2^n.

In order to represent the tensor product of vectors or matrices we use the Kronecker product. Suppose we have two vectors,

$$|a\rangle = \begin{bmatrix} x \\ y \\ \vdots \end{bmatrix}, |b\rangle = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}.$$

[1] A possible physical implementation for the qubit could be the spin of an electron.

The Kronecker expansion of their tensor product is,

$$|a\rangle \otimes |b\rangle = \begin{bmatrix} xu \\ xv \\ \vdots \\ yu \\ yv \\ \vdots \end{bmatrix}.$$

A similar expansion applies for matrices,

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots \\ a_{21}B & & \\ \vdots & & \end{bmatrix} \qquad \text{with } A = \begin{bmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & & \\ \vdots & & \end{bmatrix}.$$

Let's give two simple examples that show what a composite system and its Kronecker product representation looks like.

Example 2.2. Suppose qubit one is in a state $|0\rangle$ and qubit two is in state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. The composite system is in state,

$$|0\rangle \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{|00\rangle + |01\rangle}{\sqrt{2}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix}.$$

⊠

Example 2.3. Our next example shows a remarkable consequence of the mathematical structure of quantum mechanics. Suppose a composite system is in state $|\Psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. It is easy to show that this state cannot be written as a tensor product of two states. Suppose it could be written as $|\Psi\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)$. Expanding the tensor product gives us $|\Psi\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$. We now recognize that it is impossible for $ac = bd = 1/\sqrt{2}$ and $ad = bc = 0$.

This does not contradict the second postulate which states that the *state space* of a composite system is a tensor product of state spaces of its subsystems. Our example shows that a linear combination of vectors in the subsystems does not necessarily factor into a tensor product of two other vectors in the subsystems.

This remarkable property of certain quantum states is called an *entanglement*; the two qubits do not have a separate well defined state, they exist in a joint state. We will come back to this phenomenon in section 3.2 as it plays an important role in quantum mechanics and possibly in quantum computing. ⊠

A third ingredient of any physical theory describes the equations that dictate evolution. Physicists use the Schrödinger equation or path integrals to calculate the evolution of a quantum state. For quantum computing, a discretized time-independent version of these equations is sufficient and leads to the following postulate.

Postulate 2.3. *[28] The evolution of a closed quantum system is described by a unitary transformation. Given a state $|\Psi(t)\rangle$ at time t and evolution U, at time t' the state will be $|\Psi(t')\rangle = U|\Psi(t)\rangle$.*

Because unitary matrices are defined as matrices for which $U^\dagger U = I$ applies, unitary matrices have an inverse ($U^{-1} = U^\dagger$). Therefore, every quantum mechanical evolution is reversible. When we develop a model for quantum computation we will have to take this condition into account.

Example 2.4. A very important evolution for quantum computing will be the unitary transformation on a qubit represented by the *Hadamard* matrix,

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

The Hadamard transformation applied to the standard basis states results in,

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

⊠

The final and most puzzling postulate defines measurement in quantum mechanics.

Postulate 2.4. *[28] A measurement is described by an observable, M, a Hermitian operator on the state space of the system being observed. The observable has a spectral decomposition*

$$M = \sum_m m P_m, \tag{2.2}$$

where P_m is the projector onto the eigenspace of M with eigenvalue m. The possible outcomes of the measurement correspond to the eigenvalues, m, of the observable. Upon measuring the state $|\Psi\rangle$, the probability of getting result m is given by $p(m) = \langle\Psi|P_m|\Psi\rangle$. Given that outcome m occurred, the state of the quantum system immediately after the measurement is

$$\frac{P_m|\Psi\rangle}{\sqrt{p(m)}}. \tag{2.3}$$

We will clarify the postulate by an example.

Example 2.5. Suppose we want to measure a quantum system with the observable,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

X has two eigenvalues $(1, -1)$ with corresponding eigenvectors $|e_0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}, |e_1\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The spectral decomposition of X is,

$$X = 1P_0 + (-1)P_1 \qquad \text{with } P_0 = |e_0\rangle\langle e_0|, \quad P_1 = |e_1\rangle\langle e_1|.$$

Let us apply this measurement on a quantum system in the standard basis state $|0\rangle$. The measurement has two possible outcomes: $1, -1$. The probability that outcome 1 occurs is,

$$\langle 0|P_0|0\rangle = \frac{1}{2},$$

The state after the measurement becomes,

$$\frac{1}{\sqrt{1/2}} P_0|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}.$$

A similar argument applies for outcome -1 but the state after the measurement becomes $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$.
⊠

The measurement postulate implies many interesting properties, our next example is an easy consequence of the above. Let us calculate the average value of an arbitrary observable A on a state $|\Psi\rangle$,

$$\begin{aligned} E[A] &= \sum_m m \Pr[\text{Measurement returned } m] \\ &= \sum_m m \langle\Psi|P_m^A|\Psi\rangle \\ &= \langle\Psi| \left(\sum_m m P_m^A \right) |\Psi\rangle \\ &= \langle\Psi|A|\Psi\rangle, \end{aligned}$$

with P_m^A the projectors corresponding to the observable A. This is a useful formula which simplifies many calculations.

An observable induces a probability distribution on quantum states. Often we write about a *standard basis measurement*, meaning that we measure an observable with eigenvectors $|x\rangle$ for every $x \in \{0,1\}^n$. The probability distribution induced by the standard basis measurement is easy to recognize. The projector corresponding to measurement outcome $|x\rangle$ is $|x\rangle\langle x|$. Therefore, the probability of measuring a component $|x\rangle$ in an arbitrary superposition $\sum_y a_y|y\rangle$ is,

$$\sum_{y,z} a_y^* a_z \langle y|x\rangle\langle x|z\rangle = \sum_{y,z} a_y^* a_z \delta_{yx}\delta_{xz} = a_x^* a_x = |a_x|^2.$$

In words, the square of the norm of a component's amplitude is the probability that that particular component will be measured in the standard basis. We therefore often call the amplitude, the *probability amplitude*.

Example 2.6. Suppose we want to measure the state $\frac{|00\rangle + i|11\rangle}{\sqrt{2}}$ in the standard basis. The probability amplitudes of the different components of the standard basis are $1/\sqrt{2}$ for $|00\rangle, |11\rangle$ and 0 for $|01\rangle, |10\rangle$. The probability distribution becomes:

$$\begin{aligned} \Pr[\text{Measurement yields } |00\rangle] &= \left(\frac{1}{\sqrt{2}}\right)^2 = 1/2 \\ \Pr[\text{Measurement yields } |01\rangle] &= 0 \\ \Pr[\text{Measurement yields } |10\rangle] &= 0 \\ \Pr[\text{Measurement yields } |11\rangle] &= \left(\frac{i}{\sqrt{2}}\right)\left(\frac{-i}{\sqrt{2}}\right) = 1/2. \end{aligned}$$

⊠

There are clearly two types of states: those which are eigenvectors of an observable and those which aren't. When measuring a system which is an eigenvector of the observable, the outcome is deterministic. The measurement returns the corresponding eigenvalue with probability one and projects the eigenvector onto itself: the system remains in the same state. Systems which are not an eigenvector of the observable might be projected into different states with certain probabilities. An important consequence of this is how quantum states can be distinguished.

Lemma 2.1. *Nonorthogonal vectors can* never *reliably be distinguished.*

Proof. Suppose $|\Psi_1\rangle, |\Psi_2\rangle$ are nonorthogonal vectors. Therefore $\langle\Psi_1|\Psi_2\rangle \neq 0$ or $|\Psi_2\rangle = \alpha|\Psi_1\rangle + \beta|\Phi\rangle$ with $\langle\Phi|\Psi_1\rangle = 0$ and $\alpha^2+\beta^2 = 1, \beta < 1$. Suppose there exist an observable with projectors P_i such that

$$\langle\Psi_1|P_1|\Psi_1\rangle = 1, \qquad \langle\Psi_2|P_2|\Psi_2\rangle = 1. \tag{2.4}$$

Because $\sum_i P_i = I$, $\sum_i \langle\Psi_1|P_i|\Psi_1\rangle = 1$ and therefore $\langle\Psi_1|P_2|\Psi_1\rangle = 0$. This implies that

$$\begin{aligned} \langle\Psi_2|P_2|\Psi_2\rangle &= (\alpha\langle\Psi_1| + \beta\langle\Phi|)P_2(\alpha|\Psi_1\rangle + \beta|\Phi\rangle) \\ &= \alpha^2\langle\Psi_1|P_2|\Psi_1\rangle + \alpha\beta\langle\Psi_1|P_2|\Phi\rangle + \alpha\beta\langle\Phi|P_2|\Psi_1\rangle + \beta^2\langle\Phi|P_2|\Phi\rangle \\ &= \beta^2\langle\Phi|P_2|\Phi\rangle. \end{aligned}$$

Because $\beta^2 < 1$ and $\langle\Phi|P_2|\Phi\rangle \leq 1$, this is in contradiction with equation (2.4). □

Example 2.7. Suppose we are given a quantum state $|\Psi\rangle$ and we must find out whether it is in state $|0\rangle$ or $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. We know that $\frac{\langle 0|0\rangle+\langle 0|1\rangle}{\sqrt{2}} \neq 0$ and the two states are not orthogonal. Every observable which returns state $|0\rangle$ with a nonzero probability, always has a nonzero probability of returning $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ because the latter also has a component along the $|0\rangle$ axis. ⊠

The famous *collapse of the wavefuntion* arises as a measurement projects the state of the physical system onto an eigenvector of the observable. From a philosophical point of view this postulate is most unsatisfactory. We already noted that unitary evolution is reversible, nonetheless a measurement induces a projection of a state onto another state which is an irreversible process. If we assume quantum mechanics to be a theory of physical reality, we should regard the measurement apparatus as macroscopic quantum system. Thus, when we use the measurement apparatus on a quantum state, they evolve according to some reversible, unitary evolution. On the other hand, measuring a quantum state might irreversibly project it into a different state. These two statements are certainly incompatible as one claims that the evolution is reversible while the other claims that it is irreversible. The, almost 100 year old, debate on this foundational problem is not over yet, but what is remarkable is that quantum information science has contributed tremendously to possible solutions of this apparent paradox.

2.2.1 Heisenberg's Uncertainty Relation

One of the most studied consequences of the mathematical structure of quantum mechanics is known as Heisenberg's uncertainty principle. The measurement principle describes how measuring an observable projects a quantum system's state into one of the eigenvectors of the observable. Unlike in classical physics where a measurement does not interfere with the system itself, quantum mechanical measurements in general modify the system they are measuring upon.

Suppose now that one wants to measure the value of two observables. A theorem from linear algebra states that if the observables commute, they have a common basis in which they are diagonal. Say the two observable are A, B and $A = UD_AU^\dagger$ and $B = UD_BU^\dagger$ where D_A, D_B

are diagonal matrices. Suppose we are interested in the average value when measuring both observables. When we measure $|\Psi\rangle$ with commuting observables, it does not matter in which order we measure because,

$$\begin{aligned}\langle AB\rangle &= \langle\Psi|AB|\Psi\rangle \\ &= \langle\Psi|UD_AU^\dagger UD_BU^\dagger|\Psi\rangle \\ &= \langle\Psi|UD_AD_BU^\dagger|\Psi\rangle \\ &= \langle\Psi|UD_BD_AU^\dagger|\Psi\rangle \\ &= \langle\Psi|BA|\Psi\rangle \\ &= \langle BA\rangle.\end{aligned}$$

On the other hand, if the observables do not commute, this might not be the case. Nonetheless, Heisenberg's uncertainty principle, and especially its interpretation, tells us a lot about these noncommuting measurements.

Theorem 2.1. *Suppose A and B are two Hermitian matrices corresponding to two observables and $|\Psi\rangle$ is an arbitrary quantum state. Heisenberg's uncertainty principle states that,*

$$\Delta A\Delta B \geq \frac{\langle\Psi|[A,B]|\Psi\rangle}{2}. \tag{2.5}$$

The correct interpretation of equation (2.5) is that if we prepare a large number of quantum states $|\Psi\rangle$, and perform an equal number of A and B measurements. Then the standard deviation, ΔA, of the A results times the standard deviation, ΔB, of the B results satisfies inequality (2.5). Heisenberg's uncertainty principle is applicable with all measurements we will discuss in this thesis.

There is a special case of Heisenberg's uncertainty principle which we will need later on: the time-energy Heisenberg uncertainty principle,

$$\Delta E\Delta t = \pi\hbar. \tag{2.6}$$

It is not obvious that this equation applies as it does not fit theorem 2.1's assumptions: time is not an observable. Unfortunately, the derivation of this form of Heisenberg's uncertainty principle is beyond the scope of this thesis. However, we will spend a few words on its important interpretation [35]. If we have a system with a spread in energy of ΔE, it takes time at least $\Delta t = \frac{\pi\hbar}{\Delta E}$ to reliably move from one orthogonal state to another. Therefore, if we want a quantum mechanical system with k distinguishable states, it must consist of k orthogonal states. If we assign each orthogonal state its own energy level, we must make a tradeoff. We can cram the energy levels close together, but then the time to reliably evolve the system from one orthogonal state to another increases, or, we can increase the spread in energy levels which in turn reduces the time to evolve the system between orthogonal states. This case of Heisenberg's uncertainty relation will return in our conclusion about the role of interference and entanglement in quantum computing.

2.2.2 Final Words on Quantum Mechanics

We have only very briefly introduced the formalism of quantum mechanics. Much more can be told but we refer to [28] and other excellent books on the topic. The previous sections covered enough quantum mechanics to introduce a model of computation based on quantum mechanics. Nonetheless, a final remark is needed. Quantum mechanics actually is a formalism for physical theories rather than a physical theory itself. Given a physical system quantum mechanics does not tell us what the state space is, which unitary operator describes its evolution nor which Hermitian operator describes a measurement. It is the physicist's job to find the right state space to describe a quantum system by experimentation. Of course, most quantum information scientists work the other way around. We start with an interesting Hilbert space, describe an evolution on it and then look for a way to implement it in a physical system. Luckily, as we will see in the following sections, a limited number of basic evolutions allows us to simulate an arbitrary evolution to any degree of accuracy. Nonetheless, designing a scalable implementation of these limited number of basic evolutions is extremely difficult.

2.3 Models for Classical Computation

We introduced the importance of Turing machines earlier but have not defined them properly. We also discussed how boolean circuits are equivalent to Turing machines. With our aim of defining a quantum model of computation it will be more useful to start from the boolean circuit model of computation than from a Turing machine [2].

Let's start with the definition of a boolean circuit from classical complexity theory [13].

Definition 2.1 (Boolean Circuit). *A boolean circuit is an acyclic graph $G = (V, E)$ where the nodes V are boolean gates or boolean inputs. The input nodes have indegree 0 and there is at least one node with outdegree 0. The boolean gates are either* AND *($\wedge$), * OR *($\vee$) or* NOT *($\neg$) gates. Input nodes are either* true *or* false*. We recursively define the value of each internal node $v(x_i)$:*

- *if x_i is an AND-gate and its children are x_k, x_l, $v(x_i) = v(x_k) \wedge v(x_l)$,*
- *if x_i is an OR-gate and its children are x_k, x_l, $v(x_i) = v(x_k) \vee v(x_l)$,*
- *if x_i is a NOT-gate and its child is x_k, $v(x_i) = \neg v(x_k)$.*

The size of a circuit is the number of gates in it.

Definition 2.2 (Family of circuits). *A family of circuits is an infinite sequence $\mathcal{C} = (C_0, C_1, \cdots)$ of boolean circuits, where C_n has n input variables.*

We will be concerned with two types of boolean circuits: boolean circuits for decision problems and boolean circuits for function problems.

Definition 2.3 (Uniform polynomial circuits). *We say that a language $L \subseteq \{0, 1\}^*$ has* uniform polynomial circuits *if there exists a family of circuits $\mathcal{C} = (C_0, C_1, \cdots)$ such that the following are true:*

- *the size of C_n is polynomial in n,*
- *there is a $O(\log(n))-$space bounded Turing machine N which on input 1^n outputs a description of C_n,*

Uniform polynomial circuits for decision problems require in addition to the above that,

- *there is exactly one node with outdegree 0, the output of C_n,*

[2] Quantum Turing machines have been defined [11] but are less intuitive and only scarcely used.

- *$\forall x \in \{0,1\}^*$, if $x \in L$: given x to the input nodes, the output of $C_{|x|}$ is true.*

Uniform polynomial circuits for function problems require in addition to the first two points above that there are exactly n nodes with outdegree 0: the output of C_n.

Basic results from complexity theory prove that all languages in P have uniform polynomial circuits.

Example 2.8. Figure 2.1 shows how a very simple circuit consisting of two AND-gates, two NOT-gates and one OR-gate to create an XOR-gate. ⊠

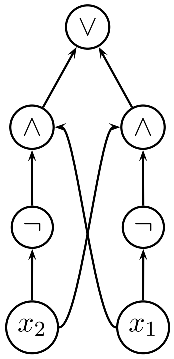


Figure 2.1: Classical XOR circuits.

Notice that we allow AND, OR and NOT gates in our circuit. This is redundant as $x_1 \vee x_2 = \neg((\neg x_1) \wedge (\neg x_2))$. Boolean circuits using only gates from the set {AND, NOT} can decide the exact same languages as the boolean circuits from our standard definition. Moreover, the size of the circuits increases only with a constant factor. We call each finite set of logic gates that is able to decide any language or calculate any boolean function a *finite universal gate set*.

Boolean circuits provide the foundation from which we can start defining quantum circuits. As our next section will show, we will switch the information carriers from bits to a qubits and introduce quantum logic gates to replace boolean gates. In this way, it is clear that quantum circuits very much resemble Boolean circuits, however, we are still one step from defining quantum circuits. Remember that quantum mechanics is a reversible theory: every quantum mechanical evolution has an inverse. Certainly most boolean circuits are not reversible: given the output to the circuit it is impossible to deduce the input: e.g., our previous XOR example. Luckily, we can make every classical boolean circuit reversible by taking two new ideas into account. First of all, we use universal *reversible* gate sets instead of e.g., {AND, NOT} which contains the irreversible AND gate. A frequently used universal gate set is the singleton {TOFFOLI}. Figure 2.2 illustrates how the TOFFOLI gate acts on 3 bits. It is not obvious that every irreversible circuit can be transformed into a reversible one. However, it was shown by Fredkin and Toffoli that any circuit computing a function $x \to f(x)$ can be transformed into a reversible circuit that computes $(x, y, z) \to (x, y \oplus f(x), z)$ where using z is a string of *ancilla* bits. We refer to [28], chapter 3 for a more in depth treatment of reversible computation.

An interesting property of this kind of circuit is that a bit is never discarded: every gate has the same number of inputs as outputs. In our next section we will see how this gives us a mathematical description which makes comparison with quantum computation easier.

Before we move on to defining quantum circuits we want to make a last remark. It is not hard to extend boolean circuits to allow for probabilistic computing. We add a RANDOM

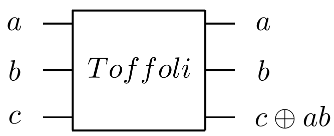


Figure 2.2: Toffoli Gate.

gate to our universal gate set which has no inputs and outputs 0 with probability 1/2 and 1 with probability 1/2. Probabilistic computing in the context of reversible boolean circuits is impossible as there is no acceptable way to reverse the operation of a probabilistic gate.

2.4 Models for Quantum Computation

The most widely used quantum computational model is based on classical reversible boolean circuits: *quantum circuits*. The differences between both models are subtle but easy to understand, yet it remains to be understood how these differences influence the computational power of the models.

Quantum circuits are very similar to their classical counterparts but differ in their basic building blocks: the information carrier in a quantum circuit is the qubit.

This new information carrier demands a review of our definition of gates. A *quantum gate* is a valid quantum mechanical evolution on a small set of qubits[3]. We will now look at some examples while introducing a convenient graphical notation to describe quantum gates and quantum circuits.

Example 2.9. (Single Qubit Operations)

An important class of single qubit operations are the *Pauli gates*. The operation of these gates on a single qubit are represented by the Pauli matrices.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.7}$$

It will pay off to know the intuitive effect of the different Pauli gates on an arbitrary qubit. The identity matrix does not actually correspond to a gate but rather to a wire: it leaves every quantum state invariant. The X-gate switches the amplitudes of the $|0\rangle$ and $|1\rangle$ basis states: $a|0\rangle + b|1\rangle \to b|0\rangle + a|1\rangle$. The Y-gate switches the amplitudes just as the X-gate but introduce a factor $-i$ in front of the $|0\rangle$ and i in front of the $|1\rangle$ components: $a|0\rangle + b|1\rangle \to -ib|0\rangle + ia|1\rangle$. The Z-gate introduces a phase of -1 in front of the $|1\rangle$ component: $a|0\rangle + b|1\rangle \to a|0\rangle - b|1\rangle$.

Three other gates that play an important role in quantum circuits are the Hadamard (H), Phase (S) and $\pi/8$ gate (T),

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix} \tag{2.8}$$

The Hadamard gate has a very intuitive effect on the basis states, it brings both states to an equal superposition of $|0\rangle, |1\rangle$: $|0\rangle \to \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ and $|1\rangle \to \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. The S-gate and T-gate introduce a factor of respectively i and $\exp(i\pi/4)$ in front of the $|1\rangle$ component. Every single qubit gate will be represented by a box in a quantum circuit, figure 2.3.

[3] There is one extra technical condition on the unitary evolutions: the evolution matrix must be efficiently computable. If this weren't the case, one could introduce quantum gates that computed uncomputable functions.

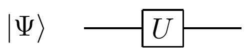


Figure 2.3: Single Qubit Gate.

$\boxtimes$

Example 2.10. (Controlled Operations)

Another important class of quantum gates are the so called controlled gates. The most important representative of this class is the *controlled-NOT*, or CNOT-gate. It's operation on a pair of qubits is represented by the following matrix,

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.9}$$

The first qubit of the CNOT-gate is often called the control qubit while the second is called the target qubit. The effect of the CNOT-gate on a composite system is to flip the target qubit if the first qubit is the $|1\rangle$ state: $|x\rangle \otimes |y\rangle \rightarrow |x\rangle \otimes |x \oplus y\rangle$. We mention without proof that there exists a construction to make a controlled version of every possible quantum gate [28]. The CNOT-gate and controlled-U gate are shown in figure 2.4.

Figure 2.4: CNOT and Controlled-U Gate.

$\boxtimes$

Example 2.11. The gate in figure 2.5 is called a *swap*-gate and is necessary for example in Shor's factorization algorithm. It intends to swap two qubits and it can be implemented with three CNOT-gates. Let us now check that it performs its intended operation. In order to prove

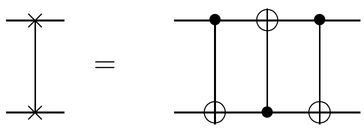


Figure 2.5: Swap Gate.

that the swap-gate performs its intended operation we follow the evolution of the four basis states. Linearity of quantum mechanics then assures that the swap-gate works for all other states.

$$\begin{aligned}
|00\rangle &\xrightarrow{CNOT1\to2} |00\rangle \xrightarrow{CNOT2\to1} |00\rangle \xrightarrow{CNOT1\to2} |00\rangle, \\
|01\rangle &\xrightarrow{CNOT1\to2} |01\rangle \xrightarrow{CNOT2\to1} |11\rangle \xrightarrow{CNOT1\to2} |10\rangle, \\
|10\rangle &\xrightarrow{CNOT1\to2} |11\rangle \xrightarrow{CNOT2\to1} |01\rangle \xrightarrow{CNOT1\to2} |01\rangle, \\
|11\rangle &\xrightarrow{CNOT1\to2} |10\rangle \xrightarrow{CNOT2\to1} |10\rangle \xrightarrow{CNOT1\to2} |11\rangle.
\end{aligned}$$

⊠

As with classical circuits, we want to limit the number of unitary evolutions one can choose, to design quantum circuits. We therefore introduce the notion of a *universal set of quantum gates*, namely a set of gates that can be used to perform general quantum computation. We first note that that there exist two types of universal sets of quantum gates. There are the real universal sets which are able to exactly implement every other possible quantum gate, e.g., {CNOT, all single qubit operations} [6, 28]. On the other hand, there are the more practical universal sets for quantum computation, e.g., {CNOT, H, S, T}. These universal sets of quantum gates are able to approximate any possible quantum gate to within any precision with only little overhead. A remarkable and deep theorem by *Solovay and Kitaev* [5, 28] states that approximating any single qubit gate to an accuracy of ϵ requires only $O(\log^c(1/\epsilon))$ gates from the set {H, S, T}. Therefore, the Solovay-Kitaev theorem implies that any quantum circuit with m CNOT gates and single qubit gates can be approximated to an accuracy ϵ using only $O(m \log^c(1/\epsilon))$ gates from the discrete set {CNOT, H, S, T}. This is interesting because physicists and engineers need only to find physical implementation for four gates instead of CNOT and *all* possible single qubit gates. Finally we note that {TOFFOLI, H} is a discrete set of quantum gates that is universal in a third way. A result from [11] states that one does not need complex amplitudes to achieve the full power of quantum computers. Quantum computers which states' are described with only real probability amplitudes are as powerfull as general quantum computers. The set {TOFFOLI, H} is universal for this restricted type of quantum computers [14]. What is interesting with the latter example is that the TOFFOLI gate is universal for reversible classical computation; all that is needed for universal quantum computation is the Hadamard gate.

A final ingredient in our quantum model of computation are measurements. We can place a measurement device anywhere in the quantum circuit and measure any set of qubits. Figure 2.6 show how we denote the measurement device.

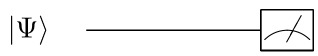


Figure 2.6: Measurement Device.

This finishes our construction of quantum circuits. From now on we are interested in finding polynomial sized quantum circuits which given a certain standard basis state as input, outputs the answer to a specific problem either exactly or with a high probability. We call the set of languages that is efficiently decided with certainty on a quantum computer the class EQP, and the class that is efficiently decided with high probability the class BQP[4].

2.5 The Deutsch-Jozsa Problem

We are now ready for a first example which demonstrates the power of quantum computers. This problem is of historical and theoretical importance because it was the first quantum algorithm that had a provable speedup compared to any classical counterpart and it introduces some concepts which will appear in other quantum algorithms too; unfortunately the problem it solves is of little practical importance.

[4]We define that a language is BQP if its instances are correctly decided with a probability of at least 2/3 in analogy with most definitions of BPP.

Suppose we are given a function $f : \{0,1\} \rightarrow \{0,1\}$ as a quantum oracle. This means that we are given a quantum operation, say O_f, that performs the following evolution on two qubits $(x, y) \rightarrow (x, y \oplus f(x))$. Note that there are exactly four possible functions:

- balanced: $f(0) = 0, f(1) = 1$,
- balanced: $f(0) = 1, f(1) = 0$,
- constant: $f(0) = 1, f(1) = 1$,
- constant: $f(0) = 0, f(1) = 0$,

We know that the function, f, is either constant or balanced[5] and we must find out which is true.

Classically we must query the oracle (or evaluate the function) twice in order to figure out whether f is constant or balanced. It is not hard to see that one query gives us no information towards deciding whether it is constant or balanced. However, there is a quantum circuit that decides the problem, querying the oracle only once, figure 2.7.

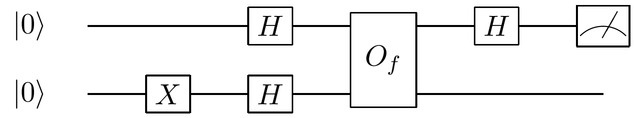


Figure 2.7: Circuit that solves the Deutsch-Jozsa problem.

Let us calculate the state of the qubits through the network,

$$
\begin{aligned}
|0\rangle|0\rangle &\xrightarrow{I\otimes X} |0\rangle|1\rangle \\
&\xrightarrow{H\otimes H} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
&\xrightarrow{O_f} \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
&\xrightarrow{H\otimes I} \pm|f(0) \oplus f(1)\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}.
\end{aligned}
$$

The three first gates are fairly simple and intuitive quantum operations. The first interesting evolution is the oracle query. Suppose we have a state $|x\rangle(|0\rangle - |1\rangle)$ and apply the oracle to it, we get $(-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$ because if $f(x) = 0$, $|0\rangle - |1\rangle \rightarrow |0\rangle - |1\rangle$ and if $f(x) = 1$, $|0\rangle - |1\rangle \rightarrow |1\rangle - |0\rangle$. We will see how this concept appears again when we discuss more complex quantum algorithms. What is interesting is after the third step, qubit one is, up to a global phase factor, in state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ if the function is constant and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ if the function is balanced: both states are orthogonal to each other. The final Hadamard gate on qubit one rotates the qubit such that a measurement in the standard basis reveals $|0\rangle$ or $|1\rangle$.

Using only one query operation this quantum circuit outputs a $|0\rangle$ if the function is constant and $|1\rangle$ if the function is balanced. On the one hand this example shows that quantum computers can be more powerful than classical computers. On the other hand, this toy problem is not

[5] A balanced function is a function where exactly half of the elements from the domain are mapped to 0 and the other half are mapped to 1.

suitable to prove that quantum computers are more powerful than classical computers for the interesting class of decision problems.

This ends our introduction of quantum mechanics and its corresponding model of computation. We have very briefly touched upon entanglement and one can imagine its importance for quantum computation. Our next chapter will investigate this and other phenomena so that we can study quantum algorithms and identify the elements that cause the speedup of quantum algorithms over classical algorithms.

Chapter 3

Quantum Mechanical Phenomena

When asking which phenomena are responsible for the speedup quantum algorithms have over classical algorithms, one gets as many answers as there are quantum information scientists: entanglement, superposition, interference, quantum parallelism, ... Where the previous chapter gave us a formal idea about quantum mechanics, this chapter discusses the intuition about quantum mechanical phenomena and compares them with classical counterparts.

3.1 Interference

Quantum mechanics harbors some of the most bizarre and counterintuitive phenomena known in physics. One of the first phenomena to be observed was interference. This phenomenon was already widely known in the context of waves but it came as a big surprise that matter sometimes also behaves as a wave in addition to its particle behavior and therefore also exhibits interference effects. Because the *matter wave* can spread in space, it can destructively and constructively interfere at certain positions.

Interference in the quantum mechanical formalism is relatively easy to understand yet its applications are stunning. We know the effect of a Hadamard transformation on the basis states,

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Now suppose we have a qubit in state

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

and we apply a Hadamard transformation. We thus get the state,

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

We see that in the last step the amplitudes of $|1\rangle$ have interfered destructively, moreover, as the amplitudes were equal in magnitude but opposite in sign, the state $H|\Psi\rangle$ does not have a component along $|1\rangle$ anymore. On the other hand, the amplitudes of the $|0\rangle$ state have interfered constructively. Intuitively we call the phenomenon that occurs when there are multiple ways to go from one state to another, in the above example $|0\rangle \to |1\rangle$ and $|1\rangle \to |1\rangle$, interference. Let us now illustrate this with an example that is known as the Mach-Zehnder interferometer [8].

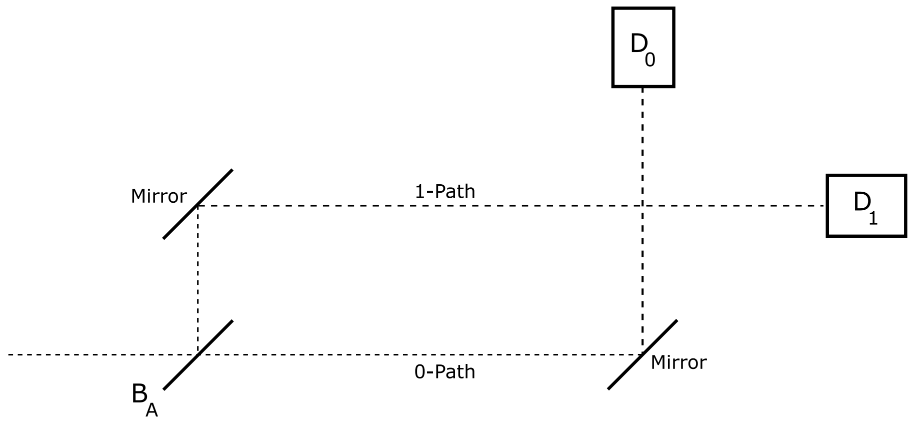


Figure 3.1: Mach Zehnder Interferometer.

3.1.1 Mach-Zehnder Interferometer

Suppose one builds the experimental setup shown in figure 3.1. Particles that move horizontally before the mirrors are in state $|0\rangle$ and those that move vertically are in state $|1\rangle$. After the mirrors, the vertically moving particles are in state $|0\rangle$ and the horizontally moving particles are in state $|1\rangle$. Therefore, the particles entering our setup from the left are in state $|0\rangle$.

Beamsplitter B_A splits the beam of particles into two beams, one going horizontally and one going vertically, corresponding to states $|0\rangle$ and $|1\rangle$. From physics, we know that we can model beamsplitters as Hadamard transformations, therefore,

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

We measure the arrival of particles with detectors D_0 and D_1. It is clear from the state after the beamsplitter that exactly half of the particles will be measured in D_0 and the other half in D_1. Our conclusion could be that the beamsplitter sent half of the particles to the $|0\rangle$ path and the other half to the $|1\rangle$ path. The detection of a particle at one of the detectors then measures whether that particle traveled along the 0-path or along the 1-path. Unfortunately this is a wrong conclusion as a modified experiment shows.

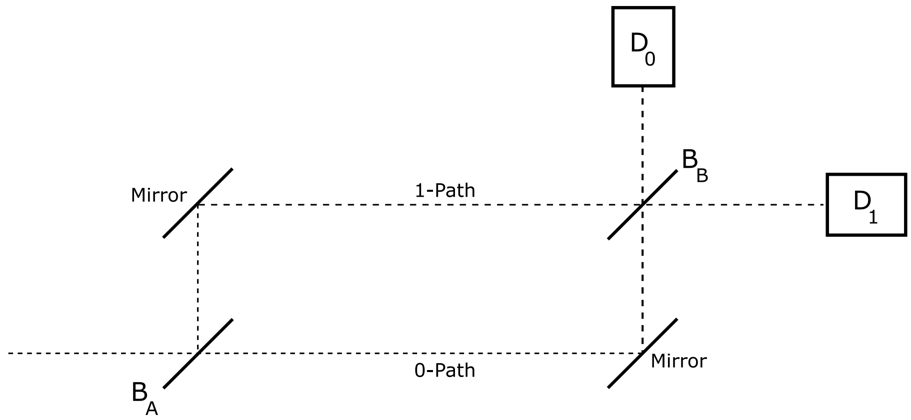


Figure 3.2: Mach Zehnder Interferometer.

Modify the experiment by inserting another beamsplitter, B_B, identical to B_A in front of the two detectors, figure 3.2. We also require that only one particle is in the setup at a time. This does not modify the results of the previous experiment: with a probability of 1/2 the particle arrives at detector D_0 and with a probability of 1/2 the particle arrives at detector D_1. In our modified experiment, the second beamsplitter splits both the horizontal and vertical beam into two new beams. Therefore, we expect the probability that a particle arrive in D_0 is,

$$\begin{aligned}&\Pr[\text{Particle along 0-path path}] \cdot \Pr[\text{Particle not deflected by } B_B] \\ &\quad + \Pr[\text{Particle along 1- path}] \cdot \Pr[\text{Particle deflected by } B_B] \\ &\quad = \frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2} = \frac{1}{2}.\end{aligned}$$

Alas, quantum mechanics has more surprises in store for us. Because the two beams interfere with each other at beamsplitter B_B we only detect particles in D_0, formally,

$$|0\rangle \xrightarrow{B_A} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \xrightarrow{B_B} |0\rangle.$$

This proves that a particle does not take one or the other path, it exists in a superposition of travelling along the 0-path and the 1-path at once and the two components of the superposition interfere constructively along the 0-path and destructively along the 1-path.

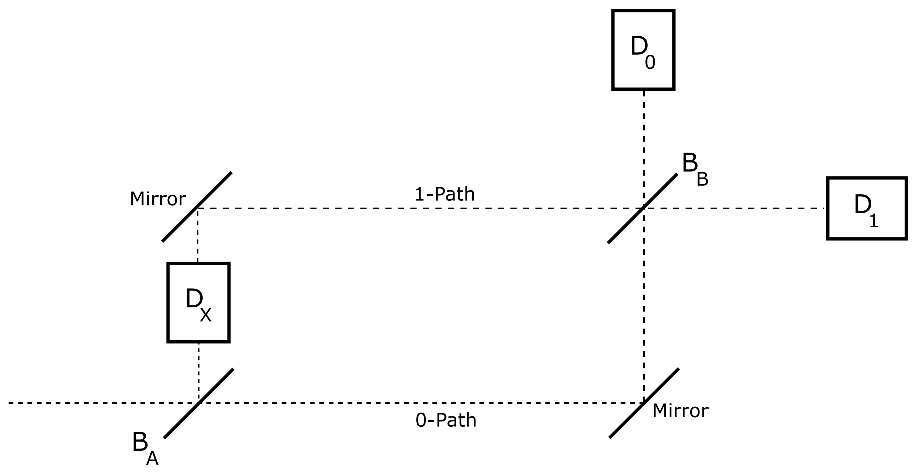


Figure 3.3: Mach Zehnder Interferometer.

Finally we introduce a detector D_X along the 1-path, figure 3.3. When a particle in a superposition arrives at D_X a measurement occurs. With probability 1/2 the particle is detected at D_X and thus projected into state $|1\rangle$ and with probability 1/2 the particle is not detected and thus projected into state $|0\rangle$. If the particle is free to travel further and it arrives at beamsplitter B_B, it will again be converted to a superposition and both D_0 and D_1 will register the arrival of half of the particles,

$$|0\rangle \xrightarrow{B_A} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \xrightarrow{D_X} \begin{cases} |0\rangle \xrightarrow{B_A} \frac{|0\rangle + |1\rangle}{\sqrt{2}} & \text{with probability } 1/2 \\ |1\rangle \xrightarrow{B_A} \frac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{with probability } 1/2 \end{cases}$$

3.1.2 Quantum Coin Flipping

Our next, more entertaining example, demonstrates quantum interference in a *gedankenexperiment* [17]. Alice and Bob agree to play a game where they are given a black box with a coin in it. Alice and Bob know that initially the coin is heads up. The only way to flip the coin is to reach inside the box without looking and flip it. There are three rounds in the game, during round one Alice can flip the coin, during round two Bob can flip the coin and in the last round Alice can flip the coin again. Alice wins the game when the coin ends heads up and Bob wins the game when the coin ends tails up.

It is clear that there is no obvious strategy for Alice nor Bob to win the game because Bob's coin in round two will be totally random oriented as well as Alice's coin in round three. The best strategy for Alice and Bob is to flip the coin with probability 1/2 so they will win the game with at least 1/2 probability.

Now Quincy enters the room with a similar box, only Quincy's box uses a quantum coin. Again, initially the quantum coin is heads up and Bob and Quincy can reach into the box without looking. Bob only knows classical operations and can flip the coin with his special quantum gloves that apply a σ_x evolution to the quantum coin. Quincy wins the game when the coin ends heads up and Bob wins the game when the coin ends tails up.

In this setting Quincy has a strategy with which he always wins the game. Quincy reaches into the box and applies a Hadamard transform. He then passes on the box to Bob who might or might not apply a σ_x evolution. Finally Quincy applies the Hadamard transform again and opens the box thereby measuring the quantum state. Formally, after Quincy applied the first Hadamard transform the coin is in state,

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

This state is an eigenvector with eigenvalue 1 of σ_x so Bob's coin flip acts as the identity on the state. Therefore it doesn't even matter whether Bob reaches inside the box or not! Because $H^2 = I$ Quincy's second Hadamard transforms the state back to its original state. Thus Quincy always wins the game.

With the Mach-Zehnder interferometer and our last example it should be clear that interference is somewhat similar to probabilistic systems. A probabilistic system can also be regarded as existing in two different states at once. Yet interference seems more general because of the possibility of destructive interference. Note that without mentioning it explicitly we already demonstrated the power of interference: section 2.5 on the Deutsch-Jozsa problem. The final Hadamard gate induces constructive and destructive interference to both the $|0\rangle$ and $|1\rangle$ component to reveal the type of function.

3.2 Entanglement

Even some of the brightest minds in the world had deep foundational problems with quantum mechanics. Einstein may be the most famous scientist who numerous times tried to find inconsistencies in the theory and ironically introduced an argument against his own theory of relativity. The thought experiment Einstein, Podolsky and Rosen [19] devised against quantum mechanics, was based on another quantum mechanical phenomenon that is currently known as entanglement. Since the 1960's, entanglement has played an important role in physics and lately has been at the center of research in quantum information theory.

Definition 3.1. *A quantum state in a composite Hilbert space is entangled if it cannot be written as a tensor product of two states.*

The most famous examples of entangled states are the Bell states,

$$|\Theta_+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, |\Theta_-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}, |\Phi_+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, |\Phi_-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}, \tag{3.1}$$

in honor of John S. Bell who proved a theory which we will come back to later. Note that these four Bell states are orthogonal and therefore define a basis, the Bell basis. We already showed in section 2.2 that these states cannot be written as a tensor product of states.

To a first approximation, entangled states can be interpreted as states where the measurement results are dependent. Take an unentangled bipartite state,

$$(a|0\rangle + b|1\rangle)(c|0\rangle + d|1\rangle) = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle. \tag{3.2}$$

Let us now call X the random variable corresponding to measuring qubit one in the standard basis, and call Y the random variable corresponding to qubit 2 in the standard basis. Equation (3.2) already reveals that the random variables X, Y are independent; $\Pr[X = x, Y = y] = \Pr[X = x]\Pr[Y = y]$. Suppose now that the bipartite state is entangled $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. Using the same random variables we can calculate the probability distribution, table 3.1.

Y\X	0	1
0	α^2	γ^2
1	β^2	δ^2

Table 3.1: Probability distribution table.

It is not hard to show that measurements on qubit one and two are independent if and only if the state is unentangled. The condition that the X, Y be independent means that each column and each row has a probability corresponding to it, such that an entry in the probability table is equal to the multiplication of the number of the respective row and column. In other words $\exists x, y, u, v : \alpha^2 = xu, \beta^2 = xv, \gamma^2 = yu, \delta^2 = yv$. That would mean that the state could be written as $xu|00\rangle + xv|01\rangle + yu|10\rangle + yv|11\rangle = (x|0\rangle + y|1\rangle)(u|0\rangle + v|1\rangle)$. Taking the contrapositive of this argument states that measurements on entangled states correspond to dependencies between the outcomes on qubit one and two.

This is a relatively simple observation but it takes us only halfway. Probabilistic computers are able to prepare similar states with similar dependencies between the bits. So why the fuss about entanglement? John S. Bell conceived of some elegant equations that strictly limit the dependencies between subsystems of classical probabilistic systems. The discovery that even the simplest entangled quantum systems violate those dependency constraints is a clear demonstration of the departure of quantum mechanics from the realm of classical probability theory.

3.2.1 CHSH Inequality

This section introduces us to some ideas with which John Bell revolutionized our perspective on quantum mechanics. The following argument is slightly different from John Bell's original derivation but it makes its point more clearly in the context of qubits.

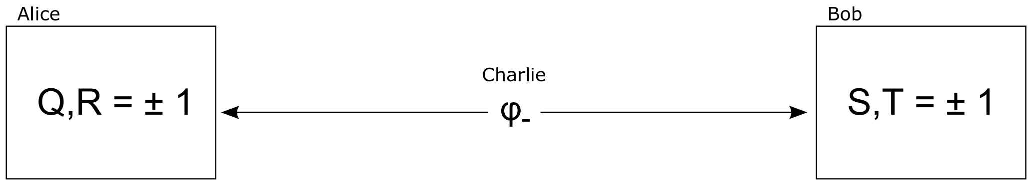


Figure 3.4: CHSH experiment.

Suppose Charlie prepares a $|\Phi_-\rangle$ state and sends qubit one to Alice and qubit two to Bob. Alice has measurement devices Q, R and Bob has measurement devices T, S all of which return ±1. Upon receiving her qubit, Alice flips a coin and measures her qubit with Q if it is tails up and with R if it is heads up. Bob does exactly the same but measures his qubit with T if his coin ends tails up and with S if it ends heads up. The measurement devices measure the qubits according to the following bases,

$$Q = Z, \qquad S = \frac{-Z - X}{\sqrt{2}}, \tag{3.3}$$

$$R = X, \qquad T = \frac{Z - X}{\sqrt{2}}. \tag{3.4}$$

There are four different measurements possible and we calculate the expected value of each upon measuring $|\Phi_-\rangle$. We illustrate how to calculate the average value of QS,

$$\begin{aligned}
\langle\Phi_-|QS|\Phi_-\rangle &= \frac{1}{2\sqrt{2}}\begin{bmatrix}0 & 1 & -1 & 0\end{bmatrix}\left(\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\otimes\begin{bmatrix}-1 & -1\\ -1 & 1\end{bmatrix}\right)\begin{bmatrix}0\\ 1\\ -1\\ 0\end{bmatrix}\\
&= \frac{1}{2\sqrt{2}}\begin{bmatrix}0 & 1 & -1 & 0\end{bmatrix}\begin{bmatrix}-1 & -1 & 0 & 0\\ -1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\\ -1\\ 0\end{bmatrix}\\
&= \frac{1}{2\sqrt{2}}\begin{bmatrix}0 & 1 & -1 & 0\end{bmatrix}\begin{bmatrix}-1\\ 1\\ -1\\ -1\end{bmatrix}\\
&= \frac{1}{\sqrt{2}}.
\end{aligned}$$

The calculation for the other average values is similar and shown in table 3.2.

Y\X	Q	R
S	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$
T	$-\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$

Table 3.2: Measurement outcomes that violate the CHSH inequality.

Because of linearity of expectation Alice and Bob find $E[QS + RS + RT - QT] = 2\sqrt{2}$. This seems acceptable were it not for the theorem by Clauser, Horne, Shimoni and Holt based on the work by John S. Bell [28].

Let us forget about quantum mechanics for a moment and derive an upper bound on the value $E[QS + RS + RT - QT]$. By definition,

$$E[QS + RS + RT - QT] = \sum p(Q = q, R = r, S = s, T = t)(QS + RS + RT - QT).$$

In classical probability theory, $p(Q = q, R = r, S = s, T = t)$ corresponds to the probability that the four measurements have these four outcomes. However, we will soo see that we must be carefull to assign outcomes to measurements we haven't performed in quantum mechanics. Basic algebra also tells us that $QS + RS + RT - QT = (R + Q)S + (R - Q)T$. At least one of $Q + R$ or $Q - R$ must be zero and the other $+2$ or -2. Therefore $QS + RS + RT - QT = \pm 2$ and thus,

$$\begin{aligned} E[QS + RS + RT - QT] &= \sum p(Q = q, R = r, S = s, T = t)(QS + RS + RT - QT) \\ &\leq \sum p(Q = q, R = r, S = s, T = t) \quad 2 \\ &= 2. \end{aligned}$$

This is in stark contrast with Alice and Bob's findings using the quantum mechanical formalism. One of quantum mechanics or our derivation must be wrong; luckily, we can let nature decide. Real experiments have shown that quantum mechanics is correct and our derivation of the upper bound of the expected value is wrong. A first error was introduced when we assumed the value of $p(Q = q, R = r, S = s, T = t)$: a quantum state only has a definite value if it is measured. Remember the Heisenberg inequality 2.2.1: when pairs of observable do not commute (as in our experiment) they cannot both have definite values. A second error is that we assume that measuring one qubit does not instantly affect the other qubit. This assumption seems obvious by Einstein's theory of relativity which states that no information can travel faster than the speed of light. The assumption that all observables have a definite value is called realism, the assumption that measuring one qubit does not affect the other is that of locality; nature has thus proved that quantum mechanics is a not a local realistic theory of reality.

The CHSH entanglement is one example in a series of proofs that show how quantum mechanics is inconsistent with our common sense. After its discovery, a whole research program to quantitatively and qualitatively analyze entanglement was started. For bipartite entanglement there exist complete theories to characterize entanglement; the multipartite case is a topic of active research.

3.2.2 Faster than light communication & Superdense coding

In this section we will show that an entangled state is a *joint* state of the qubits involved. In other words, the individual qubits do not have a definite value. Given one qubit, no measurement can possibly reveal any information about the entangled state yet by applying local transformations, we can change the joint state into several possible new joint states. This will introduce several new communication techniques of which we describe superdense coding here. Other include teleportation, quantum error correcting codes, ...

Suppose Bob visits Alice at the computer science department in Leuven to pick up a Bell state. When Bob inattentively leaves Alice's office with the Bell state he forgets the first qubit. He only notices his mistake when he has arrives at his lab in Madison, Wisconsin. Is there a way for Bob to know which state Alice gave him by only transforming and measuring his qubit? The answer is no; using the density matrix formalism (appendix A) we can show that no transformation nor measurement on a single subsystem can reveal the state of the joint system. The reduced density matrix for Bob's subsystem of $|\Theta_+\rangle$ is,

$$
\begin{aligned}
tr_B(\rho_{\theta_+}) &= \frac{1}{2}tr_B\left(|00\rangle\langle 00| + |11\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 11|\right) \\
&= \frac{I}{2}.
\end{aligned}
$$

The same calculation shows that the reduced density matrix for every Bell state is $I/2$. Thus, Bob has a maximally mixed state and any measurement will return a uniform probability distribution over all possible outcomes. Unfortunately for Bob, he will have to call Alice to know which state he currently shares with her. Luckily she doesn't mind telling Bob that she had prepared a $|\Theta_+\rangle$ state.

Because Alice and Bob now share an entangled state they decide to try out a technique Bob read about in [28] called superdense coding. The idea of superdense coding is to send only one qubit from one party to another thereby transferring two classical bits of information. The prerequisite is to have a qubit entangled up front. Basically Bob does the following: he knows that he and Alice share an entangled $|\Theta_+\rangle$ state. By applying a Z-gate to his qubit he introduces a -1 phase in front of the $|11\rangle$ component thereby transforming $|\Theta_+\rangle \to |\Theta_-\rangle$. He could also apply an X-gate to his qubit which switches the 0 and 1 components thereby transforming $|\Theta_+\rangle \to |\Phi_+\rangle$. Finally he can apply both gates thereby transforming $|\Theta_+\rangle \to |\Phi_-\rangle$. Basically, by only transforming his qubit, Bob can decide which of four states he shares with Alice. The superdense coding protocol Alice and Bob use, relate the two bit string 00 with $|\Theta_+\rangle$, 01 with $|\Theta_-\rangle$, 10 with $|\Phi_+\rangle$ and 11 with $|\Phi_-\rangle$. When Bob finishes transforming his qubit according to the two bit string he wants to transfer, he sends his qubit to Alice. Alice can now simply perform a measurement in the Bell basis to know which two bit string Bob wanted to send her.

This concludes our discussion of entanglement. There is definitely something counterintuitive going on about entanglement which only feeds our suspicion that it adds something non-classical to quantum computing.

3.3 Superposition & Quantum Parallelism

The last two phenomena we look into are closely related to interference, namely superposition and quantum parallelism. Superposition is the possibility of a quantum state to exist in two orthogonal states at once. We say that $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ exists in a superposition of the $|0\rangle$ and $|1\rangle$ state. What is interesting to note is that there exists a basis in which this state has a definite value, namely the basis consisting of $|\Psi\rangle$ itself and the state orthonormal to $|\Psi\rangle$. States for which this property applies are called superpositions.

The above implies the existence of an incoherent superposition. A state which is in an incoherent superposition does not have the property that there exists a basis in which the state has a definite value. An example of such a state is the maximally mixed state $\rho = I/2$. In appendix A it is shown that any measurement on $I/2$ returns a uniform distribution of all possible measurement outcomes. Therefore, we define mixed quantum states as incoherent superpositions.

The ability to create superpositions lies at the heart of quantum parallelism. Many authors have suggested that quantum computers are able to compute a function on an exponential amount of inputs at once. Say we have a function $f_{\mathsf{SAT}} : \{0,1\}^n \to \{0,1\}$ which on input x

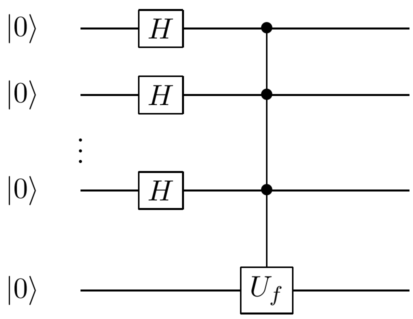


Figure 3.5: Quantum parallelism circuit.

checks whether x is a satisfying assignment for a certain SAT instance we are interested in. Could a quantum circuit such as in figure 3.5 bring us one step closer to solving this NPC problem? Let us now calculate the state at each step,

$$|\Psi_1\rangle = |0\rangle^{\otimes n}|0\rangle.$$

We then apply one Hadamard gate for each qubit,

$$|\Psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle.$$

Our last gate is a controlled-f_{SAT} gate which flips the $n+1$'st qubit if the assignment encoded by x is a satisfying assignment for our SAT instance. The circuit calculating the controlled-f_{SAT} gate has polynomial size because verifying whether a given assignment satisfies a SAT instance is in P and making a circuit controlled can be done in $O(\mathsf{poly})$ gates. This results in,

$$|\Psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f_{\mathsf{SAT}}(x)\rangle.$$

Unfortunately this is the end of the road. We have a circuit where each assignment in the superposition is correlated with a qubit encoding whether the assignment is satisfying or not but no measurement can extract any useful information out of it. Measuring the final state in the standard basis returns a state $|x\rangle|f_{\mathsf{SAT}}(x)\rangle$ with probability $\frac{1}{2^n}$. This can be achieved classically too by, for example, algorithm SimpleSAT.

Input: A SAT instance.
Output: True if the instance is satisfiable, false otherwise.
SimpleSAT(S)
(1) Set n the number of variables in S.
(2) Guess a random string x of length n.
(3) Check whether x is a satisfying assignment of S.

In analogy with quantum superpositions we can define a similar classical notion of superposition. If the state of a bit is probabilistic and exists in a probability distribution p_0, p_1, we write it as $p_0[0] + p_1[1]$. The superposition in step two of our algorithm can then be written as,

$$\frac{1}{2^{\frac{n}{2}}} \sum_{x \in \{0,1\}^n} [x].$$

The final step of the algorithm then calculates whether the state from step two is a satisfying assignment, resulting in,

$$\frac{1}{2^{\frac{n}{2}}} \sum_{x \in \{0,1\}^n} [x][f_{\mathsf{SAT}}(x)].$$

We then assume that a measurement occurs at the end of the circuit when the output is presented.

This way of reasoning about probabilistic circuits shows some similarity with quantum circuits. First of all, the concept of superposition is apparent to both. Secondly, because probability mass must be conserved when measuring a state, both probabilistic and quantum states must conserve a norm that depends on how the measurement is done. Our next section formalizes these ideas.

3.4 Computational Models Compared

We will now build a framework with which we can compare deterministic, probabilistic and quantum computation [25]. Their formal differences will become very simple and intuitive, leading to a better understanding of their respective effect on computational power.

Suppose we have a deterministic computer that runs in time $t(n)$ and space $s(n)$ on input x with $|x| = n$. We will use the notion of a Turing machine as our model of computation [27]. For convenience, we will assume that our Turing machines are *precise*, meaning that on every input x of length n, the Turing machine halts in *exactly* $t(n)$ steps and all its working tapes are exactly $s(n)$ symbols long. We refer to [13] for the construction of a precise Turing machine from an arbitrary Turing machine.

Define C the set of all possible configurations of the computer: $C = T \times S \times H$ with $T = \{0,1\}^{s(n)}$ the possible tape configurations of length $s(n)$, S the set of all states, H the set of all head positions. Associate a vector space of dimension $|C|$ to our computation and assign a basis vector to every element $c \in C$. Let c_x be the initial configuration of the Turing machine on input x and c_A be the unique accepting configuration. We define a $C \times C$ transition matrix T for the Turing machine and set $T(c_a, c_b) = 1$ if it induces a transition from configuration a to configuration b and $T(c_a, c_b) = 0$ otherwise.

Observation 3.1. *If the computation is reversible, every configuration has exactly one incoming transition and exactly one outgoing transition. Therefore, there is a single one in every row and column of the transition matrix, or, T is a permutation matrix.*

Lemma 3.1. *For any two configurations c_a and c_b, $T^r(c_a, c_b) = 1$ if and only if the computation starting in configuration a and running for r steps, is in configuration b.*

Proof. The proof is a special case of the proof of lemma 3.2. □

We can define complexity classes in this formalism as follows: a language L is in P if there exists a deterministic Turing machine corresponding to a transition matrix T such that for all inputs x, $T^{t(|x|)}(c_x, c_A) = 1$ with $t(n)$ a polynomial. We can represent the computation as a tree with transition probabilities equal to 1, figure 3.6.

A very simple modification to this formalism introduces probabilistic evolution: we allow $T(c_a, c_b) \in [0,1]$ and $\sum_i T(c_a, c_i) = 1$. We define the probability that a Turing machine in configuration a will evolve in configuration b as $\Pr[c_a \to c_b] = T(c_a, c_b)$. The restriction on the sum assures us that the computation will always evolve into *some* next state, in other words, that probability mass is preserved.

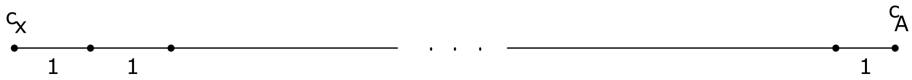


Figure 3.6: Deterministic computation tree.

Observation 3.2. *The transition matrices corresponding to a probabilistic Turing machine are stochastic: the elements of every row sum to one.*

It should be intuitively clear why the transition matrices are stochastic. A computation starts in a starting state with a probability of one. Every transition distributes some probability mass to one or more different subsequent states but the total probability mass should remain one. It is unacceptable that a particular step in the computation would only keep the algorithm running for 90 percent of the time. In terms of norms on a vector, the transition matrix must fix the 1-norm[1].

Lemma 3.2. *Define* $\Pr[c_a \xrightarrow{r} c_b]$ *as the probability that a probabilistic computation starting in state* c_a *ends up in state* c_b *after* r *steps; then for any two configuration* c_a *and* c_b*,*

$$\Pr[c_a \xrightarrow{r} c_b] = \sum_{c_1, c_2, \ldots, c_{r-1}} T(c_a, c_1) T(c_1, c_2) \ldots T(c_{r-1}, c_b). \tag{3.5}$$

Proof. We will prove the lemma using induction on the number of computational steps. The case $r = 1$ is clear: $\Pr[c_a \to c_b] = T(c_a, c_b)$ by definition.

Suppose now that for any c_a, c_b

$$\Pr[c_a \xrightarrow{r-1} c_b] = \sum_{c_1, c_2, \ldots, c_{r-2}} T(c_a, c_1) T(c_1, c_2) \ldots T(c_{r-2}, c_b). \tag{3.6}$$

Using conditional probabilities, we can write $\Pr[c_a \xrightarrow{r} c_b] = \sum_{c_{r-1}} \Pr[c_a \xrightarrow{r-1} c_{r-1}] \Pr[c_{r-1} \to c_b]$. We now use equation (3.6) and the definition to finish the proof of the lemma,

$$\begin{aligned}
\Pr[c_a \xrightarrow{r} c_b] &= \sum_{c_i} \Pr[c_a \xrightarrow{r-1} c_{r-1}] \Pr[c_{r-1} \to c_b] \\
&= \sum_{c_i} \left(\sum_{c_1, c_2, \ldots, c_{r-2}} T(c_a, c_1) T(c_1, c_2) \ldots T(c_{r-2}, c_{r-1}) \right) \cdot T(c_{r-1}, c_b) \\
&= \sum_{c_1, c_2, \ldots, c_{r-1}} T(c_a, c_1) T(c_1, c_2) \ldots T(c_{r-1}, c_b)
\end{aligned}$$

□

We now define the probabilistic class BPP in this formalism: a language L is in BPP if there exists a probabilistic Turing machine corresponding to a transition matrix T such that,

$$\forall x \in L, \qquad \sum_{c_1, c_2, \ldots, c_{(t(|x|)-1}} T(c_a, c_1) T(c_1, c_2) \ldots T(c_{t(|x|)-1}, c_b) \geq 2/3,$$

$$\forall x \notin L, \qquad \sum_{c_1, c_2, \ldots, c_{(t(|x|)-1}} T(c_a, c_1) T(c_1, c_2) \ldots T(c_{t(|x|)-1}, c_b) < 1/3.$$

[1] Recall that the ℓ_p norm of a vector u is defined as $\ell_p(u) = \left(\sum_i |u_i|^p\right)^{1/p}$.

Note that there actual is an extra restriction: we must restrict the transition probabilities to say the set $\{0, 1/2, 1\}$. We must do so because if we allow any possible probability distribution, we could program say the halting problem in the probability amplitudes and by performing enough sampling extract that information. Using only transition amplitudes from the set $\{0, 1/2, 1\}$ is flexible enough to have the full power of BPP. We will use the shorthand $T^{t(|x|)}(c_a, c_b) = \sum_{c_1, c_2, \ldots, c_{(t(|x|)-1}} T(c_a, c_1)T(c_1, c_2) \ldots T(c_{t(|x|)-1}, c_b)$ from now on. Again, we represent the computation as a tree, this time with transition probabilities between the nodes, figure 3.7.

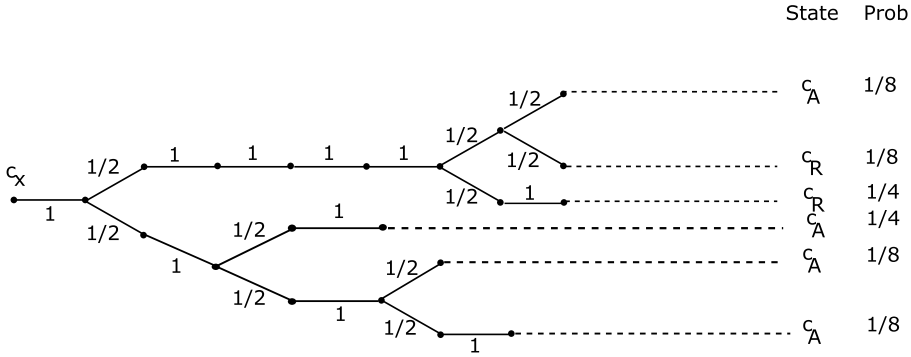


Figure 3.7: Probabilistic computation tree.

The computation tree clearly shows all possible computational paths. Moreover, it is easy to calculate the probability that a certain computational path is followed: one just multiplies the transition amplitudes corresponding to the edges of the computation path. Because probability amplitudes are positive, once a certain path is spawned, it has a positive probability mass which is bound to exist until the end of the computation. This will not be the case anymore for quantum computation.

Let us briefly allow ourselves the liberty to define something as the the '0-norm' as $\ell_0(u) = \left(\sum_i |u_i|^0\right)$. This definition is interesting as a computation leaving the '0-norm' invariant corresponds to a deterministic model of computation: only one u_i can be nonzero. We already pointed out that a computational model that preserves the 1-norm corresponds to probabilistic computation. An obvious question to ask would be whether a transition matrix preserving the ℓ_2 norm leads to an interesting model of computation. The set of matrices preserving the ℓ_2-norm correspond to the set of unitary matrices and thus this model captures exactly the power of quantum computation. An example of a quantum computation represented by a computation tree is shown in figure 3.8. What is important in this representation is that once we spawn a certain computational path, it might have a positive or negative probability amplitude. The final probability of measuring an outcome is therefore heavily influenced by how much these positive and negative probability amplitudes have interfered.

We must be careful when defining classes of languages in this formalism though. Deterministic and probabilistic Turing machines are relatively easy to define because the transition amplitudes have a clear and intuitive meaning. Unfortunately, because unitary matrices allow $T(c_a, c_b) < 0$ or even $T(c_a, c_b) \in \mathbb{C}$, and the fact that quantum mechanics is inherently time reversible, defining a quantum Turing machine introduces some technicalities that make it less elegant then quantum circuits [11]. Nonetheless we introduce an interpretation for these unitary transition matrices: we define $T(c_a, c_b)$ as the probability amplitude of the computation starting from c_a and ending in c_b. A lemma similar to 3.2 applies.

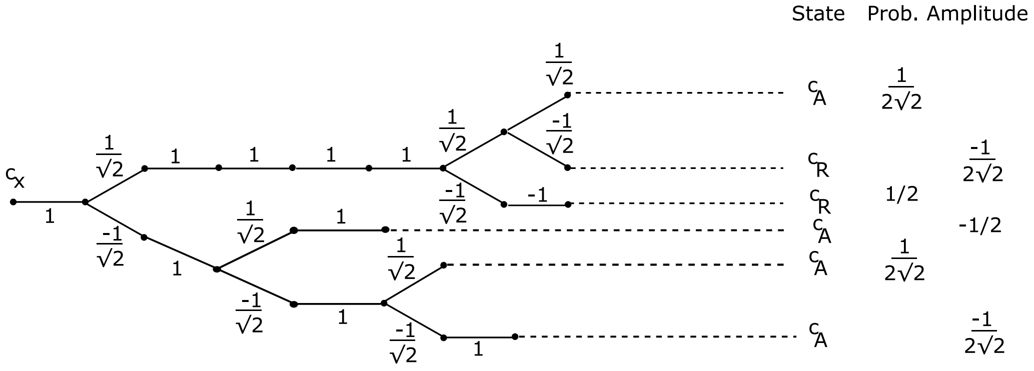


Figure 3.8: Quantum computation tree.

Lemma 3.3. *Define $A[c_a \xrightarrow{r} c_b]$ as the probability amplitude of a quantum computation starting in state c_a and ending in state c_b after r steps; then for any two configuration c_a and c_b,*

$$A[c_a \xrightarrow{r} c_b] = \sum_{c_1, c_2, \ldots, c_{r-1}} T(c_a, c_1)T(c_1, c_2) \ldots T(c_{r-1}, c_b). \tag{3.7}$$

Proof. We will prove the lemma using induction on the number of computational steps. The case $r = 1$ is clear: $A[c_a \rightarrow c_b] = T(c_a, c_b)$ by definition.

Suppose now that for any c_a, c_{r-1} the probability amplitude is,

$$A[c_a \xrightarrow{r-1} c_b] = \sum_{c_1, c_2, \ldots, c_{r-2}} T(c_a, c_1)T(c_1, c_2) \ldots T(c_{r-2}, c_b). \tag{3.8}$$

From quantum mechanics we know that we must multiply unitary matrices with a vector of probability amplitudes to compute an evolution. Therefore, we must write $A[c_a \xrightarrow{r} c_b] = \sum_{c_i} A[c_a \xrightarrow{r-1} c_{r-1}]A[c_{r-1} \rightarrow c_b]$. We now use equation (3.8) and the definition to finish the proof of the lemma,

$$\begin{aligned}
A[c_a \xrightarrow{r} c_b] &= \sum_{c_i} A[c_a \xrightarrow{r-1} c_{r-1}]A[c_{r-1} \rightarrow c_b] \\
&= \sum_{c_i} \left(\sum_{c_1, c_2, \ldots, c_{r-2}} A(c_a, c_1)A(c_1, c_2) \ldots A(c_{r-2}, c_{r-1}) \right) \cdot A(c_{r-1}, c_b) \\
&= \sum_{c_1, c_2, \ldots, c_{r-1}} A(c_a, c_1)A(c_1, c_2) \ldots A(c_{r-1}, c_b)
\end{aligned}$$

□

Conforming to the principles of quantum mechanics, we define $\Pr[c_a \xrightarrow{r} c_b] = |A[c_a \xrightarrow{r} c_b]|^2$.

We can now safely define quantum complexity classes in terms of this formalism: we say a language L is in BQP if there exists a quantum circuit with transition matrix T such that,

$$\forall x \in L, \qquad \left(\sum_{c_1, c_2, \ldots, c_{(t(|x|)-1}} T(c_a, c_1)T(c_1, c_2) \ldots T(c_{t(|x|)-1}, c_b) \right)^2 \geq 2/3$$

$$\forall x \notin L, \qquad \left(\sum_{c_1, c_2, \ldots, c_{(t(|x|)-1}} T(c_a, c_1)T(c_1, c_2) \ldots T(c_{t(|x|)-1}, c_b) \right)^2 < 1/3.$$

On the other hand, an interesting result by Bernstein & Vazirani [11] states that the power of BQP does not increase by allowing imaginary transition amplitudes. We will therefore mostly be concerned with only real transition amplitudes. On the other hand, the remark about arbitrary transition amplitudes applies. [3] showed that transition amplitudes from the set {-1, -4/5, -3/5, 0, 3/5, 4/5, 1} is sufficient to incorporate the full power of BQP.

It is not hard to imagine that certain components of $A[c_a \xrightarrow{r} c_b]$ will be positive, while others will be negative. Suppose one could design a quantum algorithm such that given an $x \in L$ as input, it destructively interferes for most rejecting paths. As such, the probability for a correct answer is boosted because of conservation of probability mass. Of course it is not at all clear how one could go about designing quantum algorithms or circuits such that these effects occur. Our next chapter will give some intuition to accomplish this but it remains a terribly difficult task to find new quantum algorithms that make use of interference.

Our next claim illustrates the power of our formalism.

Claim 3.1. *All deterministic reversible algorithms have a quantum analogue, or, any deterministic reversible classical Turing machine has an equivalent quantum Turing machine (one that decides the same language) that runs in the same time and space constraints.*

Every deterministic classical Turing machine has a reversible equivalent whose transition matrix is a permutation matrix. Intuitively, permutation matrices represent permutations of input configurations and are thus trivially ℓ_2-norm preserving. Therefore, mathematically speaking, all classical reversible gates are also valid quantum gates. Any classical algorithm can be implemented on a reversible classical circuit and therefore on a quantum circuit. Because the same time and space constraints apply, this claim implies that $\mathsf{P} \subseteq \mathsf{EQP}$. Actually even more applies, also nonreversible deterministic Turing machines have a corresponding quantum Turing machine. The only issue for nonreversible machines is that we need ancilla qubits to make it reversible before we can transform it into a quantum Turing machine; this in turn increases the space resources.

Recent research [32, 33] has investigated how the computational power changes when transition matrices preserve arbitrary ℓ_p-norms or when measurements deviate from the $|\Psi|^2$ probability rule. First of all, it was discovered that transition matrices that preserve the ℓ_p-norm with $p > 2$ are permutations of diagonal matrices. As this can be achieved with reversible deterministic computation too, unitary transition matrices are the end of the road. On the other hand, deviating from the $|\Psi|^2$ probability rule introduces some 'perverse' consequences. For any nonnegative integer, define BQP_p similar to BQP, except that the probability of measuring a basis state $|x\rangle$ equals $|\alpha_x|^p / \sum_y |\alpha_y|^p$. (Thus $\mathsf{BQP}_2 = \mathsf{BQP}$.) [33] shows that $\mathsf{PP} \subset \mathsf{BQP}_p \subset \mathsf{PSPACE}$ for all constants $p \neq 2$, with $\mathsf{BQP}_p = \mathsf{PP}$ when $p \in 4, 6, 8, \ldots$. These models of computation would be able to solve PP-Complete problems which are believed to be even harder than NP-Complete problems which in turn are believed to be not efficiently solvable. It seems that no trivial modifications to quantum mechanics lead to non-perverse models of computation.

By now we have an intuitive feeling for the different phenomena in quantum computing. We found that the concept of interference with its related notion of superposition will play an important role in quantum computing. Superposition in itself is not sufficient to cause any excitement as it has been known in the context of probabilistic computation for a long time. We therefore suggest to drop quantum parallelism from our list of potential quantum mechanical speedups. We will see that the technique we introduced in the section on quantum parallelism appears again in another quantum algorithm but there we show how interference distinguishes the algorithm from any possible classical approximation.

Entanglement seems another very likely candidate for quantum mechanical speedup. In the context of quantum communication, this phenomenon has been under careful study and introduces an exponential decrease in communication resources for several different problems. Nonetheless, in our next chapter we will suggest that its role for quantum computation is only accidental.

Chapter 4

Quantum Algorithms

All the foundational work has been done and we are ready to investigate some real quantum algorithms with an eye on what role interference and entanglement play in them. The two basic algorithms, *Grover's search algorithm* and the *quantum Fourier transform*, which forms the basis for *Shor's factoring algorithm*, are already ten years old and have been analyzed in detail. Unfortunately relatively few new algorithms have been found and one can ask why this is so? Peter Shor, the father of the quantum factoring algorithm proposes two answers to this question in [30]. First of all, quantum computers operate in a manner so different from classical computers that our techniques for designing algorithms and our intuitions for understanding the process of computation no longer work. A second reason is that we don't know if there are many problems for which quantum computers offer a substantial speed-up over classical computers. We hope that a characterization of the role interference and entanglement play in the speedup of existing quantum algorithms will give us a hint to which new problems might allow faster quantum algorithms.

Nonetheless, we will investigate Grover's algorithm, a generalization called *amplitude amplification*, and the quantum Fourier transform in order to identify the potential elements that cause the speedup of quantum algorithms over their classical counterparts. At the end of this chapter, we will work out our own perspective on the source of the speedup.

4.1 Grover's Search Algorithm

Grover's algorithm searches an unordered list for a specific item. In our prelude we were rather sloppy with our definition and never explicitly defined a robust complexity measure. Instead of counting the number of gates, we will use the number of times the algorithm queries the search space as the complexity of the algorithm, often called the query complexity. As we showed in the prelude, on a classical computer, searching an unordered list with N items cannot be done much faster than just querying all N elements. The discovery of a quantum algorithm with a query complexity of $O(\sqrt{N})$ in 1996 by Lov Grover came as a big surprise and the search for even faster methods was on. Solving NPC problems can be considered as searching a (probably) unordered search space and thus Grover's result applies. Unfortunately, the past ten years saw many proofs of the tightness of the square root speedup algorithm meaning that if they exists, efficient quantum algorithms for NPC problems must use different techniques than Grover's algorithm.

Suppose we want to search a space X with N items; as a convenience we take $N = 2^n$. We assume the items are indexed with numbers ranging from $0 \cdots N-1$, so we can represent the index with an n-qubit quantum register. Interpret a particular instance of a search problem as a function $f : \{0,1\}^n \to \{0,1\}$ that partitions the set X in good and bad items, with $f(x) = 1$ if $x \in X$ is good, and $f(x) = 0$ otherwise. We will limit ourselves to searching sets X with only one good item, say u, leaving the generalization of multiple good items to the next section. Therefore,

$$f(x) = \begin{cases} 0 & \text{if } x \neq u \\ 1 & \text{if } x = u \end{cases}$$

This function can be considered as a *black box* or *oracle*, O_f, whose internal workings should be implemented with a quantum circuit[1]. We define the query complexity of an algorithm as the number of oracle gates in the circuit.

Example 4.1. The search space of a SAT instance doesn't seem to have any structure. The most naive classical search algorithm therefore queries the database $O(2^n)$ with n the number of variables. Nonetheless, SAT neatly fits into the formalism needed for Grover's search. Suppose we interpret an n-bit string as an assignment of the n variables with 0 being false and 1 being true. We can then interpret the oracle as a polynomial sized circuit that given an assignment decides whether it is satisfying or not. Applying Grover's search to this problem would reduce the query complexity to $O(2^{n/2})$. ⊠

We require the oracle to perform an operation similar to a CNOT gate,

$$|x\rangle \otimes |b\rangle \xrightarrow{O_f} |x\rangle \otimes |b \oplus f(x)\rangle.$$

Before we delve into the details of the algorithm we will describe its workings from a bird's-eye perspective. Grover's algorithm starts by creating a uniform superposition over all possible items,

$$|\Psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle, \qquad \text{with} \qquad \forall x \in \{0,1\}^n : \alpha_x = \frac{1}{\sqrt{2^n}} \tag{4.1}$$

We have previously seen how this can be achieved. At this stage, a measurement would reveal u with a probability of $\frac{1}{2^n}$. Our goal is to describe an algorithm that succeeds with a probability of at least $\frac{2}{3}$. Grover's algorithm accomplishes this by iteratively applying an operator which gradually increases the probability amplitude α_u . This operator, often called a Grover iteration, consists of two smaller operations: a phase kickback (G_1) and an inversion around the average (G_2). Analysis of their effects will show how many times the Grover iteration is necessary for our algorithm to succeed with a high enough probability.

4.1.1 Phase Kickback

The first part of the Grover iteration performs what is called a phase kickback. It is a simple trick to introduce a phase difference depending on the outcome of a function. Figure 4.1 shows the circuit that performs the trick; note that it needs an ancilla qubit and how it resembles the Deutsch-Jozsa algorithm we discussed previously.

[1] [28] describes how one could go about implementing Grover search on a database with classical information.

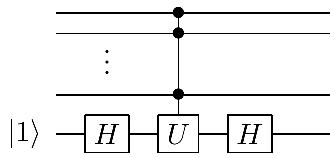


Figure 4.1: Quantum circuit for phase kickback.

Let $U = O_f$ which flips qubit $n+1$ when the first n qubits equal u. We thus get,

$$\begin{aligned}
|x\rangle|1\rangle &\xrightarrow{I^{\otimes n}H} |x\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
&\xrightarrow{U=O_f} (-1)^{f(x)}|x\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
&\xrightarrow{I^{\otimes n}H} (-1)^{f(x)}|x\rangle|1\rangle.
\end{aligned}$$

Only when the first qubit is $|u\rangle$, the bits of the second qubit are flipped. Flipping the bits of $|0\rangle - |1\rangle$ is the same as introducing a -1 phase. We therefore get a -1 phase in front of the state if and only if $f(x) = 1$ or when $x = u$. In all other cases $f(x) = 0$ and no phase is introduced. We applied a similar trick in section 2.5 for the Deutsch-Jozsa problem.

4.1.2 Inversion Around Average

Suppose the algorithm brings the n qubit register in a superposed state,

$$\sum_{k \in \{0,1\}^n} a_k|k\rangle,$$

where $a_k \in \mathbb{R}$. We will shortly see that this assumption is fulfilled by Grover's algorithm.

Next, let $\langle a\rangle = \frac{1}{2^n}\sum_{x\in\{0,1\}^n} a_x$ the average amplitude of the items in the superposition. An inversion around the average transforms every amplitude a_x to $2\langle a\rangle - a_x$, and this is exactly what the second part of Grover's iteration does.

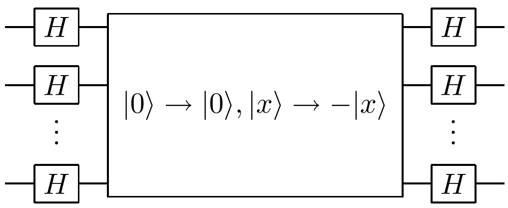


Figure 4.2: Quantum circuit for G_2.

It is nowhere clear though that this operation is unitary and thus can be performed by a quantum circuit. We will prove that the circuit shown in figure 4.2 performs the correct transformation. As shown in figure 4.2, the inversion around average operator can be written as,

$$G_2 = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2|\Psi\rangle\langle\Psi| - I, \tag{4.2}$$

with $|\Psi\rangle$ the uniform superposition such as in equation (4.1).

Lemma 4.1. *The circuit in figure 4.2 performs an inversion around the average.*

Proof. Let us start from an arbitrary superposition,

$$\sum_{k\in\{0,1\}^n} a_k|k\rangle \qquad \text{with} \qquad \langle a\rangle = \frac{1}{2^n}\sum_{x\in\{0,1\}^n} a_x.$$

We then apply the inversion around average operator, equation (4.2) on the superposition,

$$\begin{aligned}
(2|\Psi\rangle\langle\Psi| - I)\left(\sum_{k\in\{0,1\}^n} a_k|k\rangle\right) &= 2\left(\sum_{k\in\{0,1\}^n} a_k|\Psi\rangle\langle\Psi|k\rangle\right) - \left(\sum_{k\in\{0,1\}^n} a_k|k\rangle\right)\\
&= \frac{2}{2^n}\sum_{x,y,k\in\{0,1\}^n} a_k|x\rangle\langle y|k\rangle - \sum_{k\in\{0,1\}^n} a_k|k\rangle\\
&= \frac{2}{2^n}\sum_{x,y\in\{0,1\}^n} a_y|x\rangle - \sum_{k\in\{0,1\}^n} a_k|k\rangle\\
&= 2\sum_{x\in\{0,1\}^n}\left(\sum_{y\in\{0,1\}^n}\frac{a_y}{2^n}\right)|x\rangle - \sum_{x\in\{0,1\}^n} a_x|x\rangle\\
&= \sum_{x\in\{0,1\}^n}(2\langle a\rangle - a_x)|x\rangle.
\end{aligned}$$

This ends our proof that G_2 is unitary, performs an inversion around the average and can be implemented by a relatively simple quantum circuit. □

4.1.3 Grover Iteration

The next three representations intuitively show the effect of Grover's iteration on a uniform superposition.

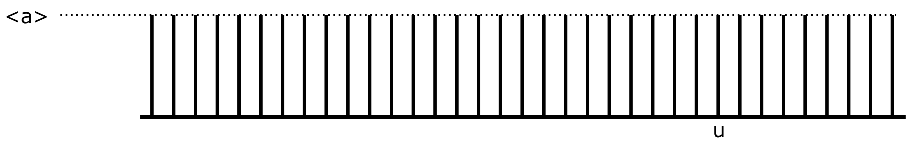


Figure 4.3: Probability distribution.

Figure 4.3 represents the probability amplitudes of a uniform superposition. After a phase kickback has been applied the amplitude of the u component becomes negative such as in figure 4.4.

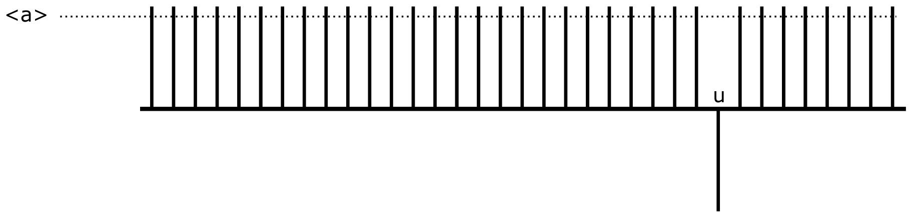


Figure 4.4: Probability distribution.

It is clear that the average amplitude has decreased. A flip around the average will therefore decrease the amplitudes of all items in the superposition except for the amplitude of u which increases above the average, figure 4.5. It shouldn't come as a surprise now that applying the Grover operation multiple times increases the probability amplitude (and therefore the probability of measuring) of u even further. However, there comes a point where the probability amplitude of u starts decreasing again. Mathematical analysis will show just how many times we must do so for the probability amplitude to be maximal.

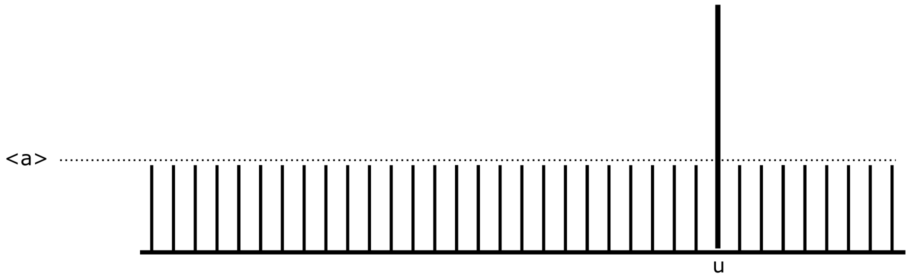


Figure 4.5: Probability distribution.

Our final analysis of Grover's algorithm needs only to consider a special subspace of the full Hilbert space. We will show how the Grover iteration is a rotation in the two-dimensional subspace spanned by $|\alpha\rangle = \frac{1}{\sqrt{2^n-1}} \sum_{x\in\{0,1\}^n, x\neq u} |x\rangle$, a superposition of all items which we are *not* looking for, and, $|\beta\rangle = |u\rangle$ the item we are looking for. In other words, our analysis restricted to the two-dimensional subspace is valid because the two-dimensional subspace is stable under a Grover iteration. We will return to the generalization of looking for multiple items later when we study amplitude amplification. The uniform superposition is a vector in this two-dimensional space too and can be written as $\cos\theta_0|\alpha\rangle + \sin\theta_0|\beta\rangle$ with,

$$\cos\theta_0 = \sqrt{\frac{2^n-1}{2^n}} \tag{4.3}$$

$$\sin\theta_0 = \sqrt{\frac{1}{2^n}}. \tag{4.4}$$

Figure 4.6 shows the two-dimensional space spanned by orthogonal vectors $|\alpha\rangle$ and $|\beta\rangle$. The uniform superposition lies somewhere in between $|\alpha\rangle, |\beta\rangle$, with a larger component along the $|\alpha\rangle$ axis. Applying G_1 on a superposition in the two-dimensional subspace flips the phase of the $|\beta\rangle$ component; this is equivalent to a reflection of the two-dimensional vector around the $|\alpha\rangle$ axis. Clearly the two-dimensional subspace is stable under G_1.

G_2 leaves the vector $|\Psi\rangle$ invariant but flips the phase of all other vectors orthogonal to $|\Psi\rangle$, more in particular vector $|\Psi^\perp\rangle$, with $|\Psi^\perp\rangle = -\sin\theta_0|\alpha\rangle + \cos\theta_0|\beta\rangle$ the vector in the two-dimensional subspace orthogonal to $|\Psi\rangle$. G_2 therefore reflects a superposition around the $|\Psi\rangle$ axis. Again, the two-dimensional subspace is stable under G_2. Because $|\Psi\rangle, |\Psi^\perp\rangle$ form an orthonormal basis for the two-dimensional subspace, we can expand the identity operator as $I = |\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|$; therefore $G_2 = 2|\Psi\rangle\langle\Psi| - I = 2|\Psi\rangle\langle\Psi| - \left(|\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|\right) = |\Psi\rangle\langle\Psi| - |\Psi^\perp\rangle\langle\Psi^\perp| =$

$$\begin{aligned}&(\cos\theta_0|\alpha\rangle + \sin\theta_0|\beta\rangle)\,(\cos\theta_0\langle\alpha| + \sin\theta_0\langle\beta|) - \\ &\quad(\sin\theta_0|\alpha\rangle - \cos\theta_0|\beta\rangle)\,(\sin\theta_0\langle\alpha| - \cos\theta_0\langle\beta|)\,.\end{aligned}$$

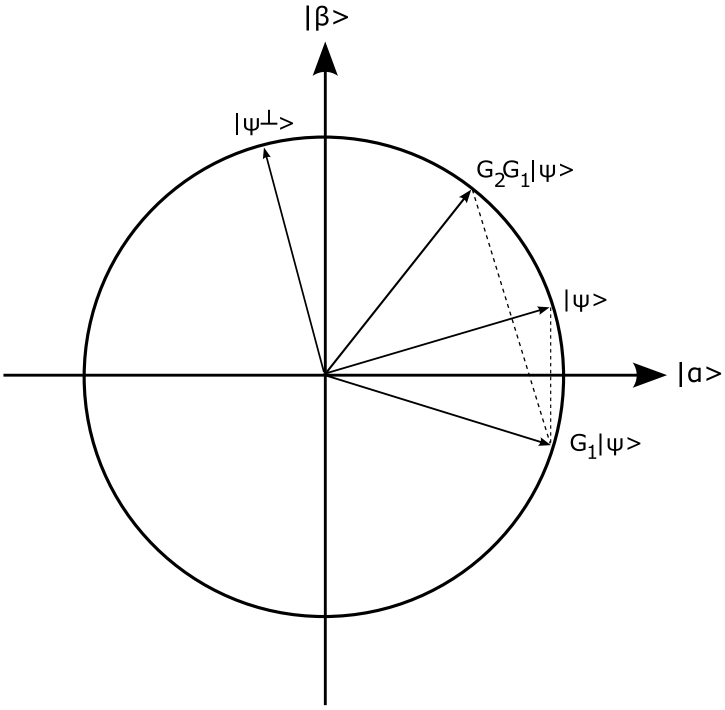


Figure 4.6: Grover's two-dimensional subspace.

Applying the Grover iteration on a state of the form $\cos\theta|\alpha\rangle + \sin\theta|\beta\rangle$ and rewriting the terms using the expansion of $|\Psi\rangle\langle\Psi| - \left|\Psi^{\perp}\right\rangle\left\langle\Psi^{\perp}\right|$,

$$\begin{aligned}
G\left(\cos\theta|\alpha\rangle + \sin\theta|\beta\rangle\right) &= G_2G_1\left(\cos\theta|\alpha\rangle + \sin\theta|\beta\rangle\right) \\
&= G_2\left(\cos\theta|\alpha\rangle - \sin\theta|\beta\rangle\right) \\
&= \left(|\Psi\rangle\langle\Psi| - \left|\Psi^{\perp}\right\rangle\left\langle\Psi^{\perp}\right|\right)\left(\cos\theta|\alpha\rangle - \sin\theta|\beta\rangle\right) \\
&= (\cos\theta\cos 2\theta_0 - \sin\theta\sin 2\theta_0)|\alpha\rangle + (\sin\theta\cos 2\theta_0 + \cos\theta\sin 2\theta_0)|\beta\rangle.
\end{aligned}$$

The whole operation can also be written in matrix form,

$$\begin{bmatrix}\cos 2\theta_0 & -\sin 2\theta_0 \\ \sin 2\theta_0 & \cos 2\theta_0\end{bmatrix}\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix} = \begin{bmatrix}\cos\theta\cos 2\theta_0 - \sin\theta\sin 2\theta_0 \\ \sin\theta\cos 2\theta_0 + \cos\theta\sin 2\theta_0\end{bmatrix} = \begin{bmatrix}\cos(\theta + 2\theta_0) \\ \sin(\theta + 2\theta_0)\end{bmatrix}$$

The transformation matrix has the form of a rotation over an angle $2\theta_0$. This is no coincidence; figure 4.6 shows the starting configuration of Grover's algorithm. $|\Psi\rangle$ is the starting state and operator G_1 reflects the vector around the $|\alpha\rangle$ axis. Next, operator G_2 reflects the last vector around the $|\Psi\rangle$ axis. From geometry we know that the composition of two reflections is a rotation.

We now have the necessary equations to find the number of Grover iterations necessary to maximize the probability of measuring a good item. Before the first iteration, the n-qubit register starts in the uniform superposition, $\cos\theta_0|\alpha\rangle + \sin\theta_0|\beta\rangle$. So far, we found that every Grover iteration rotates the vector by an angle $2\theta_0$; the n-qubit register after l iterations thus becomes,

$$\cos((2l+1)\theta_0)|\alpha\rangle + \sin((2l+1)\theta_0)|\beta\rangle.$$

It is now straightforward to derive how many Grover iterations we need in order to maximize the probability of measuring the $|\beta\rangle$. We want the probability of measuring $|\beta\rangle$ to be as close to 1 as possible. This means,

$$\begin{aligned}\sin^2((2l+1)\theta_0) &\sim 1\\ (2l+1)\theta_0 &\sim \frac{\pi}{2}\\ l &= \Theta\left(\frac{1}{\theta_0}\right).\end{aligned}$$

Recall the definition of θ_0 in equation (4.3): $\sin\theta_0 = \frac{1}{\sqrt{2^n}}$; because $\sin\theta_0$ is small,$\theta_0 \sim \frac{1}{\sqrt{2^n}}$. Finally we find that we need $l = \Theta(\sqrt{2^n})$ Grover iterations to find the item we are looking for. Because every iteration performs exactly one query, the query complexity of Grover's algorithm is $\Theta(\sqrt{2^n})$

Quantum search or Grover search can be applied in a variety of ways. My current understanding is that a useful implementation will be a thing of the future. Nonetheless, the techniques applied above are interesting from a theoretical point of view and find their way into new quantum algorithms based on random walks. We will not go into the details why this square root speedup is tight but several beautiful arguments have been given, showing that up to a constant factor this algorithm is the best we can get. Next we will show a generalization of Grover's algorithm with which we can speed up several other classical and quantum algorithms.

4.1.4 Interference & entanglement in Grover's search

It is clear from the argument about the two-dimensional subspace that interference is involved. The rotation in the two-dimensional subspace is the result of constructive interference of the probability amplitude for the good states and destructive interference for the bad states. But what about entanglement? The algorithm starts in an unentangled standard basis state. Because single qubit gates cannot introduce entanglement, the uniform superposition created by the Hadamard gates, is unentangled. It seems that the phase kickback and inversion about the average operators introduce some entanglement. Our following argument proves that entanglement is necessary for Grover's search using qubits.

Lemma 4.2. *Grover's algorithm on a search space with only one good item entangles all the qubits.*

Proof. First note that Grover's search never introduces complex factors in the probability amplitudes. This will drastically simplify the following argument. Next, we define $|\Psi\rangle$ to be the uniform superposition,

$$|\Psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}} |x\rangle,$$

the actual starting state of Grover's algorithm. Let $|\Phi_i\rangle$ be the state after the first phase kickback operation for the search instance where $|i\rangle$ is the good item we are looking for,

$$|\Phi_i\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}} |x\rangle - \frac{2}{\sqrt{2^n}}|i\rangle.$$

Suppose p qubits are unentangled from the other $n-p=q$ qubits in some arbitrary state $|\Phi_i\rangle$. This means the quantum state $|\Phi_i\rangle$ can be written as,

$$|\Phi_i\rangle = \left(a_0|0\rangle^{\otimes p} + \cdots a_{2^p-1}|1\rangle^{\otimes p}\right)\left(b_0|0\rangle^{\otimes q} + \cdots b_{2^q-1}|1\rangle^{\otimes q}\right).$$

We define p^+ the number of positive probability amplitudes and p^- the number of negative probability amplitudes in the set $\{a_0, \cdots, a_{2^p-1}\}$. We define q^+, q^- similarly. Moreover we know that $p^+ + p^- = 2^p$ and $q^+ + q^- = 2^q$. If we expand the tensor product we know that the number of positive probability amplitudes is $m = p^+q^+ + p^-q^-$ and the number of negative probability amplitudes is $n = p^+q^- + p^-q^+$. We also know that the probability amplitude of the item we are looking for is the *only* which is different in sign from *all* the other probability amplitudes after one of the phase kickback operators. Therefore, m or n must be 1. It is not hard to see that this is impossible; say $m = 1$, then at least one of p^+, q^+, p^-, q^- must be zero. But then some standard basis elements have a nonzero probability amplitude. However, we know that all standard basis elements we are not looking for, have the same amplitude. This is a contradiction and we can conclude that for no value of p, the state is a tensor product. Therefore all qubits are entangled. $\square$

4.2 Amplitude Amplification

Amplitude amplification is a technique that was developed in [10] and employs similar techniques as Grover's search to speed up other quantum algorithms. Suppose we have an algorithm, classical or quantum, which finds a good solution with a probability a. Running the algorithm a second time increases the probability of success. How many times do we expect to repeat the algorithm before we find a good solution? Call T the stochastic variable that counts the number of times we must run the algorithm before we find a good solution. We want to calculate $E[T]$,

$$\begin{aligned} E[T] &= \sum_{t=1}^{\infty} t(1-a)^{t-1}a &&= a\sum_{t=1}^{\infty} t(1-a)^{t-1} \\ &= a\frac{D}{D(1-a)}\sum_{t=1}^{\infty}(1-a)^t &&= a\frac{D}{D(1-a)}\frac{1}{1-(1-a)} \\ &= a\frac{1}{(1-(1-a))^2} &&= \frac{1}{a}. \end{aligned}$$

Thus, we expect to run an algorithm with a success probability of a, $\frac{1}{a}$ times before it succeeds. Quantum amplitude amplification reduces the expected number of runs down to $\frac{1}{\sqrt{a}}$. There is one condition for amplitude amplification to work though: the underlying algorithm must be reversible. Therefore the underlying

- deterministic algorithm must be made reversible,
- probabilistic algorithm must be made reversible using the techniques from section 4.2.2,
- quantum algorithm cannot use any measurements.

4.2.1 Algorithm & Analysis

Consider a boolean function $f : X \to \{0, 1\}$ that partitions a set X in good and bad elements, with $x \in X$ good if $f(x) = 1$ and $f(x) = 0$ if bad. Consider a quantum algorithm $\mathcal{A}$ that uses no measurements and on input $|0\rangle$ outputs a superposition $\sum_{x\in X} a_x|x\rangle$ of elements in X. Let a denote the probability that a good element is produced when $\mathcal{A}|0\rangle$ is measured. First the algorithm prepares the starting state $\mathcal{A}|0\rangle$; next, as in Grover's search algorithm, quantum amplitude amplification applies an operator, Q iteratively. Again, operator Q consists of two parts: the first part Q_1 flips the phase of the good elements. The second part Q_2 performs another reflection, $Q_2 = \mathcal{A}S_0\mathcal{A}^{-1}$ with S_0 an operator that flips the phase of all elements except $|0\rangle$.

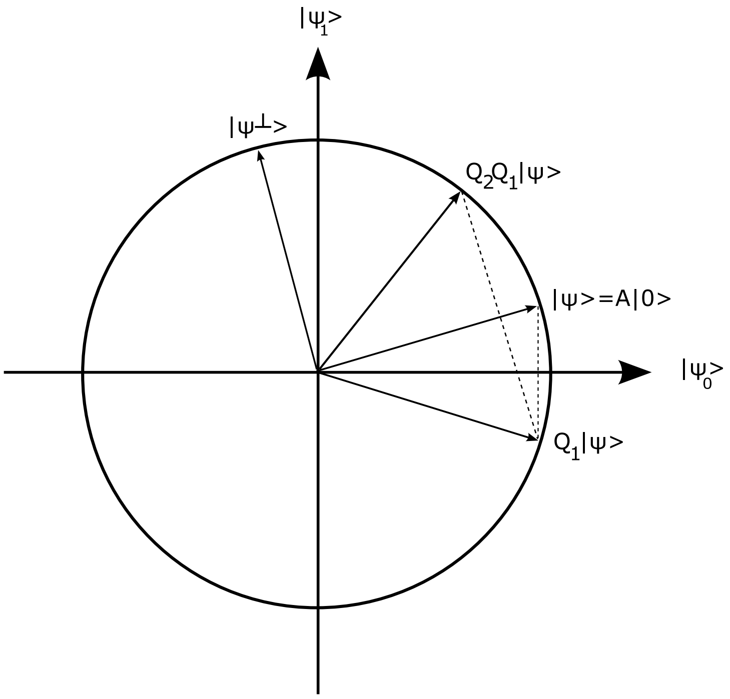


Figure 4.7: Amplitude amplification two-dimensional subspace.

Let us analyze the algorithm and prove that we only need to apply Q, $\frac{1}{\sqrt{a}}$ times. Let us name $|\Psi\rangle = \mathcal{A}|0\rangle$; we know that it consists of good and bad elements. We define $|\Psi\rangle = \cos\theta_0|\Psi_0\rangle + \sin\theta_0|\Psi_1\rangle$, with $|\Psi_0\rangle$ a normalized superposition of the bad elements in $|\Psi\rangle$ and $|\Psi_1\rangle$ a normalized superposition of the good elements in $|\Psi\rangle$. From our assumption about the success probability of $\mathcal{A}$ we know that $\sin^2\theta_0 = a$.

As with Grover's algorithm, without loss of generality, we can restrict the analysis of operator Q to the two-dimensional subspace spanned by $|\Psi_0\rangle, |\Psi_1\rangle$. It is clear from the definition of $|\Psi_0\rangle, |\Psi_1\rangle$ that the starting state $|\Psi\rangle$ is in the two-dimensional subspace. We know that Q_1 only flips the phase of the good elements and leaves all other vector invariant. It follows that the two-dimensional subspace is stable under Q_1.

The effect of Q_2 on the starting state is equal to the identity; on all other states orthogonal to the starting state, more in particular the state in the two-dimensional subspace which is orthogonal to the starting state, $\left|\Psi^\perp\right\rangle$, it flips the phase. $\left|\Psi^\perp\right\rangle$ can be written as $-\sin\theta_0|\Psi_0\rangle + \cos\theta_0|\Psi_1\rangle$. Because $|\Psi\rangle, \left|\Psi^\perp\right\rangle$ form a basis for the subspace, and Q_2 transforms them into the subspace again, the latter is stable under Q_2. It follows that a vector starting in the two-dimensional subspace remains in the two-dimensional subspace throughout the amplitude amplification iterations. Note also, that we can rewrite Q_2 limited to the two-dimensional subspace as $I - 2\left|\Psi^\perp\right\rangle\left\langle\Psi^\perp\right| = |\Psi\rangle\langle\Psi| - \left|\Psi^\perp\right\rangle\left\langle\Psi^\perp\right|$.

The calculation of the effect of operator Q on a superposition $\cos\theta|\Psi_0\rangle + \sin\theta|\Psi_1\rangle$ of good and bad states, is very similar to the one we did for Grover's search.

$$\begin{aligned}
Q\left(\cos\theta|\Psi_0\rangle + \sin\theta|\Psi_1\rangle\right) &= Q_2Q_1\left(\cos\theta|\Psi_0\rangle + \sin\theta|\Psi_1\rangle\right) \\
&= Q_2\left(\cos\theta|\Psi_0\rangle - \sin\theta|\Psi_1\rangle\right) \\
&= \left(|\Psi\rangle\langle\Psi| - \left|\Psi^\perp\right\rangle\left\langle\Psi^\perp\right|\right)\left(\cos\theta|\Psi_0\rangle - \sin\theta|\Psi_1\rangle\right) \\
&= (\cos\theta\cos 2\theta_0 - \sin\theta\sin 2\theta_0)|\Psi_0\rangle + (\sin\theta\cos 2\theta_0 + \cos\theta\sin 2\theta_0)|\Psi_1\rangle.
\end{aligned}$$

The whole operation can also be written in matrix form,

$$\begin{bmatrix}\cos 2\theta_0 & -\sin 2\theta_0 \\ \sin 2\theta_0 & \cos 2\theta_0\end{bmatrix}\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix} = \begin{bmatrix}\cos\theta\cos 2\theta_0 - \sin\theta\sin 2\theta_0 \\ \sin\theta\cos 2\theta_0 + \cos\theta\sin 2\theta_0\end{bmatrix} = \begin{bmatrix}\cos(\theta + 2\theta_0) \\ \sin(\theta + 2\theta_0)\end{bmatrix}$$

As with Grover's search the transformation matrix rotates the state vector over an angle of $2\theta_0$. Before the first iteration, the n-qubit register starts in the uniform superposition, $|\Psi\rangle = \cos\theta_0|\Psi_0\rangle + \sin\theta_0|\Psi_1\rangle$. The n-qubit register after l iterations thus becomes,

$$\cos((2l+1)\theta_0)|\Psi_0\rangle + \sin((2l+1)\theta_0)|\Psi_1\rangle.$$

Using the same argumentation as for Grover's search algorithm, in order for $\sin^2((2l+1)\theta_0)$ to be close to 1, $l = \Theta\left(\frac{1}{\theta_0}\right)$ or $l = \Theta\left(\frac{1}{\sqrt{a}}\right)$.

Example 4.2. The similarity between amplitude amplification and Grover's search algorithm demands looking into. As a matter of fact, we can interpret Grover's search algorithm as a special case of amplitude amplification.

We first interpret the set of elements X as the indices used in Grover's search. We have now embedded the Grover search space in the amplitude amplification algorithm. Our next step is to identify the oracle as the function that partitions the elements of X in good and bad ones. Finally, the Hadamard gates that create a uniform superposition can be viewed as an algorithm $\mathcal{A}$ that finds a good element with a success probability of $\frac{1}{2^n}$. This shows how Grover's search algorithm is a generalization of amplitude amplification.

In addition to this analysis, we can also easily prove the generalized version of Grover's search algorithm. Suppose we know that there are k good elements in a search space. The Hadamard gates then create a uniform superposition which when measured returns a good element with probability $\Theta\left(\frac{k}{2^n}\right)$. Applying the amplitude amplification algorithm, we find a good element with high probability using $\Theta\left(\sqrt{\frac{k}{2^n}}\right)$ queries. An algorithm exists to find a good item when we don't know k but it is beyond the scope of this work [10]. ⊠

4.2.2 Amplitude Amplification of a Random Walk

The most archetypal representative of a language in NPC is the SAT problem. Given a boolean propositional formula, does there exists a satisfying assignment for it. No polynomial time algorithm has been found to solve this decision problem and it seems safe to assume that no algorithm will be found, ever. Nonetheless, because of its wide applicability, it is useful to search for the most efficient algorithms that solve the problem. We will consider a probabilistic algorithm developed by [36] based on the concept of a random walk [1].

Input: A SAT instance.
Output: True if the instance is satisfiable, false otherwise.
RandomWalkSAT(S)
(1) Set n the number of variables in S.
(2) Guess a random string x of length n.
(3) **repeat** 3n times:
(4)   **if** All clauses satisfied
(5)   **return** True
(6)   **else**
(7)   Find an unsatisfied clause, c.
(8)   Satisfy clause c by uniformly picking a literal l and making it true by changing the value of x.
(9)   **return** False

It can be shown that this algorithm succeeds with a probability of at least $\frac{1}{1.329}^n$. Repeating the algorithm 1.329^n times gives an algorithm that runs in time $O(1.329^n \mathsf{poly})$ and succeeds with a constant probability. It is not entirely trivial to just apply amplitude amplification to this algorithm. Because of the guessing in step 2 and step 7, RandomWalkSAT isn't a reversible algorithm. However, we will show how to obtain reversibility while maintaining the correctness of the algorithm.

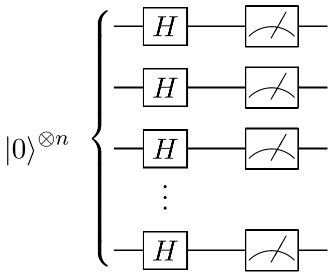


Figure 4.8: Naive quantum circuit for probabilistic choices.

The easiest quantum circuit that guesses a random n bit string as in step 2 of RandomWalkSAT is the circuit shown in figure 4.8. Unfortunately a measurement is used which violates our reversibility condition for amplitude amplification.

Figure 4.9 shows a circuit that resolves this issue. The circuit uses n ancilla qubits in addition to the n qubit register we want to uniformly sample. The effect of the circuit is,

$$|0\rangle^{\otimes n}|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n} I^{\otimes n}} \frac{1}{2^n} \sum_{x \in \{0,1\}} |x\rangle|0\rangle^{\otimes n} \tag{4.5}$$

$$\xrightarrow{CNOT^{\otimes n}_{i \to i+n}} \frac{1}{2^n} \sum_{x \in \{0,1\}} |x\rangle|x\rangle. \tag{4.6}$$

The last equation shows that every component of the n-qubit register is entangled with a similar component in the n ancilla qubits. We now require that nothing is to be done to the ancilla qubits for the rest of the algorithm. Only the n-qubit register and other ancilla qubits can be modified. The entanglement between the first and ancilla qubits ensures that no

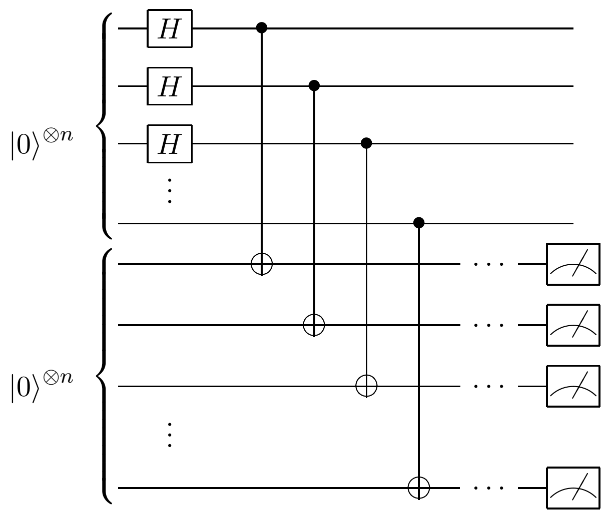


Figure 4.9: Entangling quantum circuit for probabilistic choices.

interference effects can occur. Recall that interference is an effect on the probability amplitudes of different components. Take two arbitrary components, $|x\rangle|x\rangle, |y\rangle|y\rangle$; a unitary evolution on the n-qubit register might transform the first state $|x\rangle|x\rangle$ into $|y\rangle|x\rangle$ but no interference with $|y\rangle|y\rangle$ can occur due to the ancilla qubits.

Finally, at the end of the amplitude amplification algorithm we measure the n ancilla qubits in the standard basis which induces a collapse. Suppose that the outcome of the measurement on the ancilla qubits is $|t_1 t_2 \cdots t_n\rangle$, then we know that the first component and our random string has been sampled as $t_1 t_2 \cdots t_n$. All other random choices in our algorithm can be transformed to this reversible form as long as we keep the ancilla qubits for different probabilistic choices distinct. At the end of the amplitude amplification we can measure all ancilla qubits at once, collapsing the whole superposition to a superposition that reveals our probabilistic choices. Note also, that if we wanted to perform a unitary operation U depending on the measurement outcome, we can just perform a controlled-U operation. [28] show how any unitary operation can be transformed into a controlled-unitary operation. The entanglement ensures that no interference between the states depending on two possible measurement outcomes occurs.

This method is rather expensive in terms of ancilla qubits but it shows an important technique that will prove useful in our perspective on the role of interference and entanglement in quantum computing.

Theorem 4.1. *Using ancilla qubits, all measurements can be postponed to the end of the algorithm.*

The construction from theorem 4.1 allows the amplitude amplification of the random walk to work. RandomWalkSAT needs a register for n, a register for x, $3n$ registers for the c's and $3n$ registers for the l's. However, we modify RandomWalkSAT and add quantum gates which entangle the register for x, the registers for the c's and the registers for the l's with an equal number of ancilla qubits. It is clear that these extra quantum gates make RandomWalkSAT invertible. Now let $\mathcal{A}$ be the modified RandomWalkSAT circuit. Next, we describe how to implement Q_1. This operator should flip the phase of all satisfying assignments. We know that checking whether a given assignment satisfies a formula can be done in polynomial time. Therefore, we let Q_1 be the polynomially sized quantum circuit that checks whether register x is a satisfying assignment; we do not take the ancilla qubits into account. This concludes the description of the amplitude amplification algorithm for the classical random walk. However, amplitude amplification as we describe it assumes that we know the success probability of $\mathcal{A}$. In our case, this depends on the number of satisfying assignments which we do not know beforehand. We refer to [10] for a more advanced amplitude amplification algorithm that works without knowing the success probability.

4.3 The Quantum Fourier Transform & Shor's Factorization Algorithm

The quantum Fourier transform is the basis for several algorithms which are in some sense related: eigenvalue estimation, Shor's factoring algorithm and the discrete log problem. As we discussed in the prelude, factoring is a very old problem that has resisted an efficient solution on classical computers. Let FACTORINGD be the following problem: given an integer n and an integer k; does n have a factor larger than k? FACTORINGD is the decision problem corresponding to the problem of finding a (prime) factor of a given number. FACTORINGD is believed to be in NPI; the class of problems that is probably hard but not complete for NP and believed to sit somewhere in between P and NPC. The discovery of Shor's algorithm puts FACTORINGD in the class BQP. It was conceivable at the time of the discovery in 1994, that quantum computers would be able to solve NPC problems. Research performed during the last ten years has tempered the optimism and it is believed that NPC problems are out of reach even of quantum computers. Nonetheless there is still a handful of problems in NPI which are still believed to be solvable on quantum computers: one primary candidate is GRAPH ISOMORPHISM on which promising research has been performed.

4.3.1 The Quantum Fourier Transform

Back to the quantum Fourier transform. In order for us to focus on the role of interference and entanglement in the algorithm we will skip some parts of the analysis referring to [28] for more details. The quantum Fourier transform is a technique that implements a discrete Fourier transform in an efficient way. The discrete Fourier transform takes as input a vector of N complex number, $[x_0, x_1, \cdots, x_{N-1}]$ and outputs N transformed complex numbers, $[y_0, y_1, \cdots, y_{N-1}]$ with,

$$y_k = \sum_{j=0}^{N-1} e^{\frac{2\pi ikj}{N}} x_j.$$

The quantum Fourier transform is very similar, only it is a linear operator which on the basis states applies a transformation,

$$|k\rangle = \sum_{j=0}^{N-1} e^{\frac{2\pi ikj}{N}} |j\rangle,$$

where we assume that $N = 2^n$ to make our analysis more insightful. It will be convenient to adopt the notation $j = j_1 j_2 \cdots j_n = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n$ and for the binary fraction $0.j_l j_{l+1} \cdots j_m = j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$. We refer to [28] where it is shown with some algebra that the quantum Fourier transform can be written as,

$$|j_1 \cdots j_n\rangle = \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)}{2^{n/2}}. \tag{4.7}$$

Figure 4.10 shows a relatively simple circuit which given an n-qubit input register, prepares the superposition in equation 4.7. The circuit introduces a new quantum gate, the controlled R_k gate, which is represented by the unitary matrix,

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}.$$

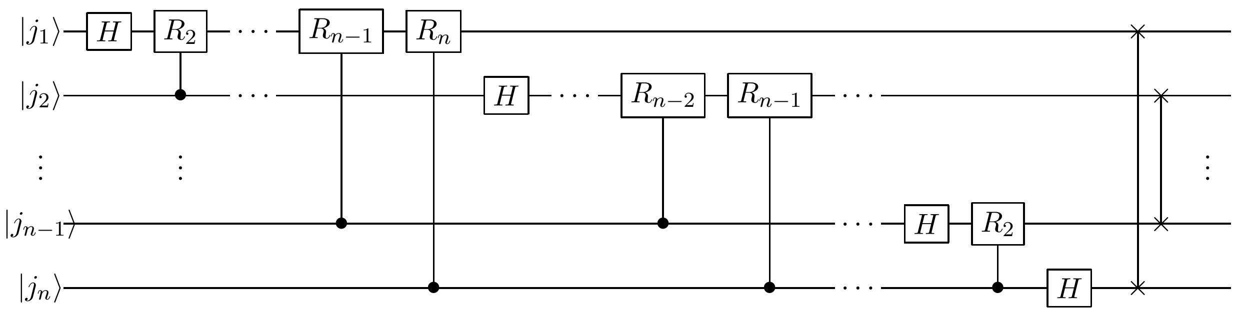


Figure 4.10: Circuit for quantum Fourier transform.

We will work through an example to show what happens when the state $|j_1 j_2 \dots j_n\rangle$ is an input to the circuit shown in figure 4.10. By applying the Hadamard gate to the first qubit, the state becomes,

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right) |j_2 \dots j_n\rangle,$$

since $e^{2\pi i 0.j_1} = -1$ when $j_1 = 1$ and $+1$ otherwise. Applying the controlled R_2 gate produces the state,

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right) |j_2 \dots j_n\rangle.$$

When we continue to apply the controlled R_k gates, the qubits are in state,

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right) |j_2 \dots j_n\rangle.$$

Next we can perform a similar procedure on the second qubit, first the Hadamard gate on the second qubit produces the state,

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right) |j_3 \dots j_n\rangle.$$

We continue applying the controlled R_k gates and find the state,

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right) |j_3 \dots j_n\rangle.$$

Finally, we find that the state of the qubits just before the swap gates is,

$$\frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right) \dots \left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right).$$

The swap gates at the end of the circuit switch the qubit positions in order to produce the quantum Fourier transform in equation (4.7). The circuit clearly shows that the wire starting at qubit k has 1 Hadamard gate and $n-k$ controlled R_x gates. Summing over all qubits in the

n qubit register we find that the circuit needs only $\sum_{i=1}^{n}\sum_{k=1}^{n-i} 1 = \Theta(n^2)$ quantum gates to compute the quantum Fourier transform. The fastest classical discrete Fourier transform algorithms such as the *fast Fourier transform* need $\Theta(n2^n)$ gates. This clearly shows an exponential decrease in circuit size.

Unfortunately, this does not mean that we can just use a quantum computer to compute any Fourier transformation. The Fourier coefficients are hidden in the probability amplitude of the state and there is no straightforward way to discover them. Nonetheless, for a certain class of problems it is possible to unleash the power of the quantum Fourier transform and force an exponential decrease in circuit size. We next briefly introduce one representative of this class of algorithms, Shor's factoring algorithm.

4.3.2 Shor's Factoring Algorithm

Shor's factoring algorithm consists of two parts: a quantum mechanical and a classical part. We refer to [28] for the classical part which reduces the problem of factoring to finding the order of an element in the group of integers modulo some number. In this section we show how to solve this *order finding* part using a quantum computer.

The order-finding problem is the following. Given two numbers N, x, with no common factors, the order of x modulo N is the least nonnegative integer r such that $x^r \equiv 1 \mod N$. Order-finding is believed to be a hard problem on a classical computer yet figure 4.11 shows how it can be performed using a quantum circuit.

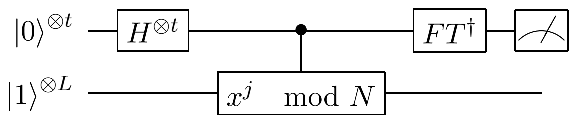


Figure 4.11: Quantum circuit for order finding.

Let U be the quantum gate which on a state $|y\rangle$ operates as follows: $U|y\rangle = |xy \mod N\rangle$. Then the $x^j \mod N$-gate is actually a set of gates that include several U gates. The eigenvectors and eigenvalues of U will prove to important shortly,

$$U|u_s\rangle = \exp\left[\frac{2\pi i s}{r}\right]|u_s\rangle \qquad \text{with} \qquad |u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\exp\left[\frac{-2\pi i s k}{r}\right]\left|x^k \mod N\right\rangle.$$

The size of the quantum registers t, L can be adjusted to reflect the required success probability. The first step of the order-finding circuit creates a superposition over all possible numbers from 0 to $2^t - 1$,

$$\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle|1\rangle.$$

The next step calculates x to the order specified by the t-qubit register, modulo N and puts it in the L-qubit register. We thus get a superposition of the form,

$$\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle\left|x^j \mod N\right\rangle.$$

This last state is again an example of what some researchers have called quantum parallelism. We create a superposition of all possible powers and then calculate some function depending on all the components of the superposition. The output of the function is stored in an extra register which ends up entangled with the input to the function. As we have argued in the previous chapter, this sort of parallelism exists for classical probabilistic computation too. In the next step, we apply the inverse quantum Fourier transform which introduces interference effects and produces the state,

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|s/r\rangle|u_s\rangle,$$

with $|u_s\rangle$ an eigenstate of the U operator and r the order of x modulo N. It is not trivial to show why this is so but the important point is that we now have a superposition of two entangled registers with register one containing some information about r. Measuring both registers produces a state from which we can extract r using continued fractions.

We have skipped some important and non-trivial proofs. Nonetheless, and more importantly, we have seen the different states that occur during the quantum order finding algorithm and the quantum Fourier transform. It is clear that the t-qubit and L-qubit register from the order finding algorithm are entangled. In contrast to the technique we introduced in section 4.2.2 on the amplitude amplification of a random walk, some elements of the L-qubit register will be in the same state because of the periodicity of the exponentiation operation. For example, suppose we want to factor $N = 15$ and we need to find the order of $x = 7$. After the modular exponentiation the state of the quantum system will be,

$$\frac{1}{\sqrt{2^t}}\left[|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + \cdots\right]. \tag{4.8}$$

From equation (4.8) it is clear that because of the periodicity, some components have the same L-qubit state. Therefore, the inverse quantum Fourier transform introduces interference effects between the qubits of the first register for which the second register has the same value.

Unfortunately, it is not easy to point out the source of the speedup of this quantum algorithm because both entanglement and interference are present. In our next section we take a step back and introduce a way of reasoning independent of the quantum algorithm under review to clarify the role of both phenomena for quantum computing.

Chapter 5

The Role of Interference and Entanglement in Quantum Computing

In this chapter we will develop our own original perspective on the role of interference and entanglement in quantum computing. Considering the computation paths representation, it should be clear by now that from a computer science perspective, destructive interference is a powerful primitive. Probabilistic computers are able to spawn new computation paths but once they are spawned, their probability mass is bound to exist until the end of the computation. Destructive interference in quantum computers allows computation paths to extinguish when a computation path with a positive and negative probability amplitude interfere with one another. From a physics point of view, entanglement seems a rather strange and powerful primitive which even questions the validity of Einstein's theory of relativity. This is also how in general the field of quantum computation is divided. On one hand there are the computer scientists who believe that interference causes quantum mechanical speedup while on the other hand, there are physicists who believe that entanglement is more important. This new perspective on the problem does not claim to settle the question once and for all but is an attempt to unify and generalize some concepts that have been suggested in the literature and provide some insight in the problem.

5.1 The role of measurements

When we run a quantum algorithm, we are interested in the final probability distribution over different states. Our definitions of acceptance in languages such as $\mathsf{BPP}, \mathsf{BQP}$ are uniquely characterized by the probability distribution of the final states. Therefore, it would be convenient if we would only have to consider measurement outcomes at the end of the computation. This would allow us to analyze the whole quantum algorithm in the quantum system's Hilbert space and then consider the probability distribution at the end of the computation. However, at first sight, one could design a quantum algorithm with several intermediate measurements. The outcomes of the intermediate measurements could for example influence the choice of future quantum evolutions. Analyzing such a quantum computation requires us to switch between the Hilbert space where the quantum evolutions occur, and another mathematical space where classical decision are made based on measurement outcomes.

Our first goal is to simplify things and show that it is possible to transform every quantum algorithm to a unitary quantum evolution with one measurement as the final step of the computation. We have already discussed how this transformation can be done in section 4.2.2. For deterministic as well as probabilistic algorithms, the technique we developed showed how we can always introduce some entanglement between the actual system and some ancilla qubits in order to postpone the measurement to the end of the computation. Note that we also assume that we perform only local measurements at the end of a quantum algorithm; we do not allow measurements in arbitrary global bases for the same reasons that we only allow local gates. Therefore, every local measurement can be transformed into a measurement in the standard basis by introducing local unitary transformation just before the measurement. Henceforth we can conclude that we the analysis of any quantum algorithm now reduces to an analysis in the Hilbert space of the quantum system and the interpretation of one single standard basis measurement at the end.

The measurement in a standard basis now induces a probability distribution over all possible measurement outcomes. It is conceivable that certain dependencies, originating from entanglement between qubits, exist. However, nothing prohibits probabilistic algorithms to end up in the same probability distribution as the quantum algorithm. This is not in contradiction with the CHSH-inequality from section 3.2.1: this inequality shows how the average values of four different measurements are related in a local-realistic and quantum setting. It does not tell us about any special properties of the probability distribution for one particular measurement.

We therefore suggest that there is little effect on the probability distribution *due to* entanglement. Nonetheless, this does not mean that those states are not entangled, or that entanglement plays no role in quantum computing. We will come back to this issue in the next two sections.

5.2 The role of Hilbert Spaces

Consider the analysis of Grover's algorithm: we only require 2^n orthogonal states to encode our search space. Our analysis never assumes that these states are encoded in qubits. What we will show in this section is that there exists an isomorphism between a 2^n dimensional Hilbert space and a Hilbert space that is 2^n dimensional because it is a tensor product of n two-dimensional Hilbert spaces. Moreover, we will show how to construct the isomorphism for any quantum algorithm.

As a toy example, we demonstrate our ideas on a four dimensional Hilbert space. We have always assumed that a system that needs four different states would be represented with two bits or qubits. Say we want a superposition of the first and fourth state, $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. If we had a physical (quantum) system that has four distinguishable states, we could have $|1\rangle, |2\rangle, |3\rangle, |4\rangle$ as basis states and represent the superposition as $\frac{|1\rangle+|4\rangle}{\sqrt{2}}$. Of course, because this is a whole new physical system, implementing the unitary transformations on this system will be different from implementing them on a qubit system. Scalability will be an issue as adding one qubit doubles the size of the Hilbert space. One would expect that doubling the size of the Hilbert space requires a doubling of the number of distinguishable states for a physical system composed of a single system. In other words, if the number of qubits increases linearly, the number of distinguishable states grows exponentially and this might introduce some scalability issues. Nevertheless, what is interesting in this purely mathematical construction is that we only consider one physical system. Therefore no entanglement exists as the physical system has no system to entangle with!

Let the non-tensor product Hilbert space be called a *simple* Hilbert space. We now generalize the previous construction and construct the isomorphism between larger Hilbert spaces.

First we define the simple Hilbert space itself; the dimension of the simple Hilbert space is equal to the dimension of the tensor product Hilbert space, for simplicity we assume that it is 2^n. We assume a standard basis in the simple Hilbert space, $|x\rangle$ with $x \in [0 \cdots 2^n - 1]$. This induces a trivial isomorphism between the states of both Hilbert spaces: $|x_1 x_2 \cdots x_n\rangle \rightarrow |x_1 2^{n-1} + x_2 2^{n-2} + \cdots + x_n\rangle$. We then compute the Kronecker product of the matrix representation of the different gates that are applied at the same time (using the identity if there is no gate) and let a unitary transformation on the simple Hilbert space correspond to this matrix.

Lemma 5.1. *Every quantum algorithm can be embedded in a simple Hilbert space where no entanglement is involved.*

The representation of the simple Hilbert space by a Hilbert space that is a tensor product of different qubit subsystems is just one out of many possible Hilbert space decompositions. Note, that our construction is sufficiently general that all quantum algorithms can be represented in a simple Hilbert space. Our construction has one important consequence: because the simple Hilbert space corresponds to one physical system, entanglement is nonexistent. It is not even defined for a non-tensor product Hilbert space!

We now argue that because algorithms are a mathematical construct and only need a mathematical space and an interpretation, from the mathematical point of view, the concept of entanglement is not necessary. Only interference effects remain in the 'naked' simple Hilbert space. The interference effects influence the dynamics and might shorten the time to produce a certain probability distribution.

Unfortunately *we* do not live in Hilbert space and we must find a way to efficiently represent the Hilbert space in physical reality. Scalability will be a big concern and our next section will show how entanglement enters the picture again.

5.3 Physical consideration for scalable quantum computation.

Our next argument is somewhat far from the computer scientist's world, but nevertheless the idea is intuitive. Suppose we apply the previous argument on a classical computer: instead of using bits we use quarts, information elements that have four possible values. We could imagine ourselves designing a computer that operated with quarts: adding two quarts would result in something like $02 + 03 = 11$. Where bits are usually implemented as a voltage difference between say 0 and 2 volts, one could imagine implementing quarts using voltages $0, 2, 4, 6$. However, if we push this principle even further and want to implement the computer using one information carrier that has 2^n different states. This exponential number of different states can easily be implemented using only a linear number of bits, however a single physical system with that much different states will need voltages between 0 and $2^n - 2$. One way out of this is to demand that the voltages always stay between 0 and 2 but should crowd exponentially close together. In terms of classical electromagnetic theory this does not introduce a problem, yet, when our voltage differences become sufficiently small one enters the realm of quantum mechanics. There the Heisenberg uncertainty principle from section 2.2.1 and more in particular equation (2.6) applies. This implies that evolving a computer between two distinguishable states would require an exponential amount of time [35].

Let us return to quantum computers [12]. Every physical system has a number of degrees of freedom which in our case correspond to the number of subsystems of the system. At any moment in time, the state of a degree of freedom or subsystem is uniquely described by a pair of canonical coordinates such as for example, position and momentum. The physical resources required to implement the physical system are the ranges of the positions Δq and momenta Δp used by the computation. The measure of the resources used, is defined as the *phase space area* or *action* $A = \Delta p \Delta q$. From physics, we know that the connection between the physical resources and the Hilbert space comes from the fact that every quantum state occupies an area in phase space of size h, Planck's constant. Moreover, orthogonal (distinguishable) states correspond to non-overlapping states. This means that a physical system with action A can accommodate A/h orthogonal states.

If we apply the argument above to our quantum computer we find the following. We required that the Hilbert space featured 2^n orthogonal states, therefore, $A/h = 2^n$. If we use only one physical system to provide the whole action A, it is clear that A will grow exponentially with n. The only solution is to use multiple degrees of freedom or multiple physical subsystems. Suppose we have t identical physical systems with action A; the dimension of the corresponding Hilbert space will be $\frac{A}{h}^t$. Setting this dimension equal to 2^n means that we need $t = \Theta(n)$ different subsystems.

This argument shows that entanglement enters the picture only because we need to save on resources. As far as the analysis is concerned, there is no mathematical idea behind entanglement and it is a by-product of embedding the Hilbert space in a physical system.

5.4 Conclusion

The analysis we performed above is a synthesis of different ideas that exist in the literature [34, 12]. Our argument first showed that we only need to consider quantum computation with measurements at the end. This makes analysis easier as we need only to concern ourselves with the unitary evolutions that drive the computation and the final probability distribution. We need not take into account for example, measurement outcomes in between upon which new unitary evolutions might be conditioned. The latter would force us to move between what happens in the quantum mechanical Hilbert space and what happens in another space where the classical parts of the computation perform their work. Next, we showed how entanglement is totally absent in simple Hilbert spaces. Nonetheless, we can model any possible quantum algorithm in these simple Hilbert spaces and derive properties such as their time, space or query complexity. Finally we found that in order to embed the mathematical concept of quantum computation on a physical system, we need to introduce entanglement to save on resources.

Note that our conclusion does not contradict our findings for Grover and Shor's algorithm. In lemma 4.2 and section 4.3.2 we showed that entanglement is necessary for both algorithms. Our discussions clearly stated that there is entanglement between the *qubits* so we implicitly assumed that we implemented our algorithms on a physical system composed of two-dimensional subsystems. This shows the importance of the construction of simple Hilbert spaces in lemma 5.1.

Thus our argument suggests that the more powerful dynamics due to interference are more important from the perspective of computational power and algorithms. Entanglement is what allows scalability and thus would be something to be careful about when designing implementations for quantum computers.

One must be cautious though as our argument does not work the other way around. We cannot simply design an algorithm in a simple Hilbert space and then run it on a quantum computer. The reason is that we only allow evolutions from the universal gate set to implement a quantum algorithm. Therefore, not every matrix transformation in the simple Hilbert space corresponds to a physically realizable evolution.

Some researchers have quantitatively calculated the evolution of entanglement during a quantum computation [23, 29]. Their research focuses on calculating or simulating a particular algorithm and measure how entangled a certain state of the computation is. It seems to me that with our conclusion in mind, this research is of little interest in the search for the power of quantum computation.

Although the problem we started from is interesting from a foundational point of view, our own solution to it did not involve significant research but is more a synthesis of known results. The suggestion of where interference and entanglement enter the picture is a *qualitative* answer to the question what role interference and entanglement play in quantum computing. Computer science has yet to find where the power of quantum computers sits in the complexity hierarchy. A solution to this problem is equivalent to a *quantitative* answer on the role of interference for quantum computing. Small steps toward its resolution have been made but no definitive insight has been discovered. Nonetheless, we introduce some concepts in our next chapter.

Chapter 6

The Road Ahead

Our last chapter gave a qualitative answer to what role interference and entanglement play in quantum computing. Computer scientists are ultimately interested in a quantitative characterization of the power of quantum computation. In this chapter we introduce some concepts and discoveries about the place of quantum computing in the hierarchy of complexity classes. Unfortunately, the difficulty of this task prohibits spectacular progress in the short time the question has existed. Therefore, we give an overview of the state of the art while mentioning open problems and some suggestions for further research.

6.1 Introduction to Quantum Complexity

We will be mostly concerned with the computational power of the two classes $\mathsf{EQP}, \mathsf{BQP}$. Because of their importance we repeat their usual definitions here[1].

Definition 6.1 (EQP or exact quantum polynomial time [11]). EQP *is the set of languages which are accepted without error by a uniform family of polynomial sized quantum circuits.*

Definition 6.2 (BQP or bounder-error quantum polynomial time [11]). BQP *is the set of languages which are accepted by a uniform family of polynomial-size quantum circuits, with at most 1/3 probability of error.*

From this definitions it is already clear that $\mathsf{EQP} \subseteq \mathsf{BQP}$ and as we pointed out in claim 3.1, since reversible computation is a special case of quantum computation, $\mathsf{P} \subseteq \mathsf{EQP}$.

Theorem 6.1. $\mathsf{BPP} \subseteq \mathsf{BQP}$.

Proof. We will apply a technique similar to the one we used in section 4.2.2 on the probability amplification of the random walk. The idea is that for every coin toss we have to make, we introduce an ancilla qubit which we put in state $1/\sqrt{2}(|0\rangle + |1\rangle)$. We then perform the rest of the algorithm conditioned on the ancilla qubit. This introduces entanglement between the rest of the qubits and the ancilla qubit. At the end of the computation, we measure the ancilla qubit in the standard basis, and the rest of the qubits collapse into a state that would have occurred if the coin toss were equal to the measurement outcome. If the probabilistic algorithm erred on less then 1/3 of its computation paths, then the quantum simulation will err on the exact same computation paths. This proves that the probability that the quantum computation errs is less than 1/3. $\square$

[1] Our definition from section 3.4 is highly intuitive but not the usual definition of BQP.

Grover's algorithm is interesting in that it proves that quantum computers are more powerful than classical computers for oracle problems. Unfortunately, the speedup is relatively moderate; moreover, it has been shown in [7] that every bounded-error quantum algorithm making T queries can be simulated with a deterministic algorithm making only $O(T^6)$ queries, and every exact quantum algorithm making T queries can be solved with a deterministic algorithm making only $O(T^4)$ queries. In other words, the oracle setting does not allow quantum computers to cross the barrier of superpolynomial speedups. My research has not yielded many results about EQP so we limit our characterization here to that of BQP.

It is not hard to show that FACTORINGD is in NP: given n and k, a nondeterministic computer can just guess a factor $Q > k$, divide n by Q and check whether the remainder is 0.

Theorem 6.2. *If* $\mathsf{P} \neq \mathsf{NP}$ *then there are languages in* NPI *that are not in* NPC *[13].*

It is generally assumed that FACTORINGD is one of these problems in NPI. Shor's algorithm puts FACTORINGD in BQP and evidence has been gathered showing that other NPI problems (e.g., GRAPH ISOMORPHISM) might also be solvable using quantum computers in polynomial time. Therefore we can conclude that the power of BQP at least partly overlaps with NPI.

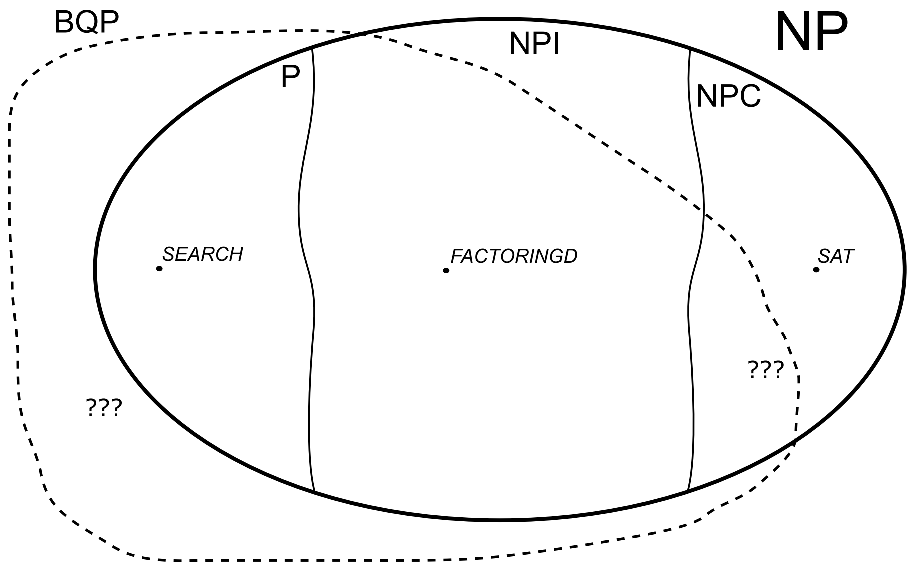


Figure 6.1: Quantum Complexity Theory Overview.

Figure 6.1 represents the results above. Note that it is unknown how nondeterminism relates to quantum computation, in other words, $\mathsf{BQP} \subseteq \mathsf{NP}$? Completeness implies that if there is one algorithm in BQP that is in NPC then $\mathsf{NP} \subseteq \mathsf{BQP}$; in figure 6.1 this means that the dotted line would encompass the whole set NP. Figure 6.1 bounds BQP 'from below', the discussion up next bounds the power of BQP from higher up in the complexity hierarchy.

6.2 Naive Quantum Simulations

Simulating one model of computation with another is the most straightforward way to prove inclusions between complexity classes. $\mathsf{P} \subseteq \mathsf{NP}$ can be shown by simulating a deterministic computer on a nondeterministic computer. This simulation is trivial as the former is a special case of the latter. We will next show what resources are necessary for simulating a quantum computer on a deterministic computer which will prove a first upper bound on BQP.

In order to make our arguments more readable, we make three assumptions. First of all, we will assume that the input state is a standard basis state. This is just a convenience which does not lose any generality. Secondly, using the result from [11] that states that complex factors do not increase the power of quantum computers, we assume that our algorithms only involve real numbers. Finally, our last assumption relates to a problem that pops up always when one simulates continuous variables on a discrete computer: there is some discretization error involved. We will therefore assume that all numbers involved in describing a quantum circuit or input state are rational numbers. During a computation[2], nonrational numbers often sneak in and much research has been done on the propagation of rounding errors during a simulation. We will mostly skip its discussion, casually pointing out where its effect might have stronger consequences.

Let us consider a naive algorithm that simulates a quantum circuit and decides whether the instance is in BQP or not, QSimI. The two most important components of the algorithm are a vector that stores all probability amplitudes and a subroutine that calculates the Kronecker product of the matrix corresponding to the quantum gate on certain qubits and the identity matrix on all other qubits.

Input: An input state $|\Psi\rangle$ and a description of a quantum circuit C.
Output: True if the instance is in BQP, false otherwise.
QSimI($|\Psi\rangle, C$)
Transform the state $|\Psi\rangle$ in a probability amplitude vector v.
foreach Gate X in circuit C
  Calculate the matrix transformation A_X corresponding to quantum gate X.
  Perform the matrix multiplication $v = A_X v$.
Compute the probability distribution $p = v^T v$.
if The probability mass of the accepting state is larger than 2/3 in p
  return True
else
  return False

This algorithm fails to be efficient in two important ways. First of all, writing down every amplitude takes an exponential amount of space. There are 2^n basis states and each one can have its own probability amplitude. There is only one restriction that states that vector v in our algorithm must be normalized; this reduces the number of independent probability amplitudes to only $2^n - 1$. Moreover, every gate corresponds to a 2^n by 2^n matrix which demands an exponential amount of time for every matrix update. Therefore, this algorithm runs in exponential time and space and allows us to state that BQP $\subset$ EXPSPACE.

Luckily, by only computing the probability amplitudes of the necessary states, we can get rid of the exponential space requirement. Remember the acceptance probability of an algorithm for a BQP language from section 3.4,

$$\sum_{c_1, c_2, \ldots, c_{t(|x|)-1}} T(c_a, c_1) T(c_1, c_2) \ldots T(c_{t(|x|)-1}, c_b) \geq 2/3. \tag{6.1}$$

[2]A simple Hadamard applied to a basis state already introduces nonrational coefficients!

It is not hard to show that every component of the sum requires only polynomial time and space to compute. Because $t(n)$ is a polynomial by definition, every component of the sum is a product of a polynomial number of real numbers. Every number $T(c_i, c_j)$ can be computed efficiently: if the gate corresponding to the number influences qubits i and j, $T(c_i, c_j)$ is equal to the corresponding entry in the gate's local transformation matrix; if the gate does not influences qubits i and j, $T(c_i, c_j) = 0$ if $i \neq j$ and $T(c_i, c_j) = 1$ if $i = j$. QSimII refines the simulation of QSimI.

Input: An input state $|\Psi\rangle$ and a description of a quantum circuit C.
Output: True if the instance is in BQP, false otherwise.
QSimII($|\Psi\rangle, C$)
Set $M = 0$.
Compute the starting and accepting states c_a, c_b.
foreach Path $c_a, c_1, \cdots, c_{t(|x|)-1}, c_b$
  $M = M + T(c_a, c_1)T(c_1, c_2) \ldots T(c_{t(|x|)-1}, c_b)$
if $M \geq 2/3$
  return True
else
  return False

There are an exponential number of paths from c_a to c_b in the computation tree but the important thing is that the algorithm only requires a polynomial amount of space. This allows us to refine our previous statement about BQP and conclude that BQP $\subset$ PSPACE. Without proof we note that we can even refine our bound on BQP to BQP $\subseteq$ PP.

More results are known about BQP but they are of less importance than the above. Nonetheless, in order to tighten the bounds on the power of BQP, there exists an approach which might prove fruitful. Instead of analyzing universal quantum circuits, we can limit ourselves to certain subsets that make analysis easier. The next two sections show that certain subsets have properties that make them easier to analyze. We then hope that we can extend the subsets, while keeping the nice properties to come to a characterization of universal quantum circuits and its corresponding bounded-error class, BQP.

6.3 Stabilizer Circuits

We have always described our quantum states by writing down the amplitudes of the basis states. This approach worked well, but from the set of all possible formalisms to represent quantum states, there is one in particular that has an elegant mathematical structure: the stabilizer formalism [28, 4, 15]. Unfortunately, the formalism's computational power is limited to that of the class $\oplus$L, which suggest that it is not even capable of universal classical computation.

6.3.1 Concept

The stabilizer formalism takes a radical departure from the notion to describe states with probability amplitudes. Instead, it describes a quantum state with a number of operators that leave the state invariant. If we are careful in choosing these operators, a unique quantum state corresponds to the list of operators. Moreover, if we let the state evolve through a circuit, we have to update the operators that leave it invariant instead of a vector of probability amplitudes.

6.3.2 Stabilizer States

When an operator leaves a state invariant it means that the state is a eigenstate of the operator with eigenvalue 1. Unfortunately, an operator can have many eigenvectors corresponding to eigenvalue 1. Thus, if we require that a quantum state is invariant under different operators, we might limit the number of possible eigenvectors the state can be in. Just how many operators and how they must be different will become clear soon.

First of all it is not clear how this could be an interesting way to describe state vectors. Every operator on an n-qubit system can be represented by a $2^n \times 2^n$ matrix. Even writing down one of these requires exponentially more space than writing down all the probability amplitudes. The solution to this problem is to limit ourselves to operators from the group $\mathcal{P}_n$; the group of tensor products of Pauli operators on n qubits together with the multiplicative constants $\pm 1, \pm i$. There are only 4 Pauli operators on one qubit, which require only 2 bits to specify. A tensor product of n Pauli operators together with a multiplicative constant thus requires only (2n + 2) bits to specify. This is a dramatic decrease in space to represent operators but still allows a rich mathematical structure.

Let us develop the theory with a simple example. Suppose we have a quantum computer in state,

$$|\Theta_+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}},$$

using the notation we introduced in section 3.2. We define operator $A_i B_j$ to mean Pauli operator A applied to qubit i and Pauli operator B applied to qubit j where we discard the identity operator for clarity. Simple linear algebra[3] then shows that $X_1 X_2 |\Theta_+\rangle = |\Theta_+\rangle$ and $Z_1 Z_2 |\Theta_+\rangle = |\Theta_+\rangle$. We say that $|\Theta_+\rangle$ is *stabilized* by the operators $X_1 X_2$ and $Z_1 Z_2$. A little less obvious is that $|\Theta_+\rangle$ is the *only* state that is stabilised by these operators.

Definition 6.3. *If S is a subset of $\mathcal{P}_n$ and let V_S be the set of all quantum states that are stabilized by S, then S is said to be the stabilizer of V_S.*

Definition 6.4. *If S is a subgroup of $\mathcal{P}_n$ and let V_S be a one-dimensional vector space, we define the single basis vector in V_S a stabilizer state.*

It is not hard to see that V_S is a vector space. Every linear combination of two elements in V_S is also stabilized by S. Our next property lies at the heart of the mathematical structure of the stabilizer formalism. It makes analyzing stabilizer states much easier.

Theorem 6.3. *Every stabilizer is a subgroup of $\mathcal{P}_n$.*

Proof. The group operation is matrix multiplication: if M, N stabilize a state, it is trivial to show that MN and NM also stabilize the state. Matrix multiplication is associative and the identity matrix I stabilizes every state. Finally, $\forall A \in \mathcal{P}_n : A^2 = I \vee A^4 = I$ thus every element in S has an inverse. This shows that a stabilizer is a subgroup of $\mathcal{P}_n$. □

Furthermore, a subgroup of $\mathcal{P}_n$ can be described by a set of generators. A result from group theory states that every group of k elements has a generator of at most $\log k$ elements. Therefore every subgroup of $\mathcal{P}_n$ with $|\mathcal{P}_n| = 4^{n+1}$ can be described with at most $2(n+1)$ generators. As pointed out previously, every generator needs only $2n+2$ bits: therefore we need $O(n^2)$ bits to describe a state in the stabilizer formalism.

[3] It is actually more convenient to think of operator X as an operator that switches $|0\rangle$ to $|1\rangle$ and vice versa and operator Z as an operator that introduces a phase of -1 in front of $|1\rangle$. It is then more intuitive to see that $X_1 X_2, Z_1 Z_2$ stabilize $|\Theta_+\rangle$.

Example 6.1. Suppose we want to describe a generalized Bell state,

$$\frac{|0000\rangle + |1111\rangle}{\sqrt{2}}.$$

It is clear that switching all $|0\rangle$'s and $|1\rangle$'s at once leaves the state invariant: $X_1X_2X_3X_4$ is definitely in the stabilizer. Switching the phase of two $|1\rangle$'s at once also leaves the state invariant: Z_1Z_2, Z_1Z_3, Z_1Z_4, Z_2Z_3, Z_2Z_4 and Z_3Z_4 are all in the stabilizer. Note that because $Z^2 = I$, we cannot combine any two of the Z_iZ_j stabilizers into $X_1X_2X_3X_4$. On the other hand, we know that because there are four qubits involved, we need four independent generators. Therefore, we know that $\langle X_1X_2X_3X_4, Z_1Z_2, Z_1Z_3, Z_1Z_4\rangle$ uniquely determines the generalized Bell state. ⊠

Example 6.2. A lot of our algorithms started in the $|0\rangle^{\otimes n}$ state. Recall that the Pauli Z operator leaves $|0\rangle$ invariant and switches the phase of a $|1\rangle$. Therefore $Z_1, Z_2, \cdots, Z_n$ are all independent operators stabilizing the state $|0\rangle^{\otimes n}$; thus $\langle Z_1, Z_2, \cdots, Z_n\rangle$ is the stabilizer of the all zero state. ⊠

Not just any subgroup of $\mathcal{P}_n$ can be used as a stabilizer for a non-trivial vector space[4]. Two necessary, yet not sufficient conditions are that

- the elements of S commute,
- $-I$ is not an element of S.

Suppose $M, N \in S$ anti-commute. Take any element $|\Psi\rangle \in V_S$: $|\Psi\rangle = MN|\Psi\rangle = -NM|\Psi\rangle = -|\Psi\rangle$ which implies that $|\Psi\rangle$ is the zero vector, a contradiction. Furthermore, suppose $-I \in S$: $|\Psi\rangle = -I|\Psi\rangle = -|\Psi\rangle$ again $|\Psi\rangle$ would have to be the zero vector, a contradiction. Suppose a Pauli operator A with a phase of $\pm i$ were in the stabilizer, then $A^2 = -I$ would be in the stabilizer which is prohibited.

Observation 6.1. *No Pauli operator with a phase of $\pm i$ can be in the stabilizer.*

We will now derive some properties in order to show what structure a generator must have in order for it to qualify as a stabilizer. First of all, we want the generators $g_1, g_2, \cdots, g_l$ of a stabilizer S to be independent. This means that removing an element g_i makes the group generated smaller,

$$S = \langle g_1, g_2, \cdots, g_l\rangle \neq \langle g_1, g_2, \cdots, g_{i-1}, g_{i+1}, \cdots, g_l\rangle.$$

There exists a useful tool for checking whether a set of Pauli operators is an independent set of generators: the *check matrix*. The check matrix for a set of l generators on n qubits is a $l \times 2n$ matrix. Each row of the check matrix corresponds to a generator; we let $r(g_i)$ be the row corresponding to generator g_i,

$$r(g_i) = [x_{i1}x_{i2}\cdots x_{in}z_{i1}z_{i2}\cdots z_{in}].$$

Suppose $g_i = A_1A_2\cdots A_n$ then for each element A_j:

- if $A_j = I$: $x_{ij} = z_{ij} = 0$,
- if $A_j = X$: $x_{ij} = 1, z_{ij} = 0$,

[4]The zero vector is stable under the action of any subset of $\mathcal{P}_n$ and we therefore call it a trivial vector space.

- if $A_j = Y$: $x_{ij} = 1, z_{ij} = 1$,
- if $A_j = Z$: $x_{ij} = 0, z_{ij} = 1$.

Let A be the Pauli operator in g_i acting on qubit l. Then element l and $l+n$ uniquely determine the Pauli operator. Therefore, up to a global phase factor, row $r(g_i)$ uniquely characterizes a generator.

Example 6.3. The check matrix corresponding to the generator,

$$\langle XZZXI, IXZZX, ZXIXZ, ZZZZZ, XXXXX \rangle,$$

is,

$$\left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

$\boxtimes$

Next we define a $2n \times 2n$ matrix Λ by,

$$\Lambda = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix},$$

where the I matrices on the off-diagonals are $n \times n$. The following two lemma's show a connection between the check matrix and the independence of the generators. Note that in the check matrix representation we perform addition modulo 2, $\oplus$.

Lemma 6.1. *g_i, g_j commute ($[g_i, g_j] = 0$) if and only if $r(g_i)\Lambda r(g_j)^T = 0$.*

Proof. First of all note that for any two different Pauli operators, A, B: $AB = -BA$; for two equal Pauli operators A we have $A^2 = I$. Next, let $g_i = A_1 A_2 \cdots A_n$ and $g_j = B_1 B_2 \cdots B_n$. Therefore,

$$\begin{array}{rcll} [g_i, g_j] & = & 0 & \Leftrightarrow \\ (A_1B_1)(A_2B_2)\cdots(A_nB_n) - (B_1A_1)(B_2A_2)\cdots(B_nA_n) & = & 0 & \Leftrightarrow \\ (A_1B_1)(A_2B_2)\cdots(A_nB_n) - (-1)^t(A_1B_1)(A_2B_2)\cdots(A_nB_n) & = & 0, & \end{array}$$

where t is the number of posititions where $A_i \neq B_i$. Therefore, $[g_i, g_j] = 0$ if and only if $t \equiv 0 \mod 2$. In other words, the two Pauli operators commute if and only if they differ in an even number of positions. We will now show that $r(g_i)\Lambda r(g_j)^T$ will exactly count $t \mod 2$. First we must expand the matrix product with the dual vector and vector,

$$r(g_i)\Lambda r(g_j)^T = x_{i1}z_{j1} \oplus x_{i2}z_{j2} \oplus \cdots \oplus x_{in}z_{jn} \oplus x_{j1}z_{i1} \oplus x_{j2}z_{i2} \oplus \cdots \oplus x_{jn}z_{in}. \tag{6.2}$$

We now want to show that this expression is equal to $0 \mod 2$. Suppose g_i has a Pauli I operator for qubit l, then $x_{il} = 0, z_{il} = 0$. Because I commutes with any other Pauli operator, whatever the Pauli operator for qubit l in g_j, the contribution to t should be 0. It is not hard to see that this is exactly what happens in equation 6.2: both x_{il} and z_{il} are multiplied with some other values and binary added to the sum but because they are 0 do not contribute. A similar reasoning applies if the Pauli operator in g_i for qubit l is X, Y or Z: if the corresponding Pauli operator in g_j is equal, the contribution is $0 \mod 2$ and if the corresponding Pauli operator is different, the contribution is $1 \mod 2$. $\square$

Lemma 6.2 ([28])**.** *Let $S = \langle g_1, g_2, \cdots, g_l \rangle$ be such that $-I$ is not an element of S. The generators are independent if and only if the rows of the corresponding check matrix are independent.*

Proof. We first note that $\forall i : g_i^2 = I$. If not, $-I$ would be in S and S would be the trivial stabilizer. Next, if we forget the phase factor, binary addition of two rows of the check matrix corresponds to the group operation on $\mathcal{P}_n$: $r(g_i) \oplus r(g_j) = r(g_i g_j)$.

The rows of the check matrix are linearly dependent if $\exists j : a_j \neq 0$ such that $\sum_i a_i r(g_i) = 0$. Using the binary addition property, this condition occurs if and only if $\prod_i g_i^{a_i}$ is equal to the identity, up to an overall multiplicative factor. Because $-I \notin S$, the factor must be 1 and the last condition corresponds to $g_j = \prod_{i \neq j} g_i^{a_i}$. Therefore, the generators are linearly dependent if and only if $\exists j, a_j \neq 0 : \sum_i a_i r(g_i) = 0$. Taking the contrapositive of this statement proves the lemma. □

Finally we want to prove how the number of generators for a stabilizer S influences the size of V_S.

Lemma 6.3 ([28])**.** *Let $S = \langle g_1, g_2, \cdots, g_l \rangle$ be independent generators such that $-I$ is not an element of S. Fix $i \in [1, l]$. There exists $g \in \mathcal{P}_n$ such that $g g_i g^\dagger = -g_i$ and $\forall j \neq i : g g_j g^\dagger = g_j$.*

Proof. Let G be the check matrix corresponding to $g_1, g_2, \cdots, g_l$. By lemma 6.2 the rows of G are independent, there exists a vector x such that $G\Lambda x = e_i$ with e_i an l-dimensional vector with a 1 in position i and 0 everywhere else. Let g be the operator corresponding to $r(g) = x^T$. By definition we have $r(g_j)\Lambda r(g)^T = 0$ for $j \neq i$ and $r(g_i)\Lambda r(g)^T = 1$. This implies that $g g_i g^\dagger = -g_i$ and $\forall j \neq i : g g_j g^\dagger = g_j$. □

Lemma 6.4 ([28])**.** *Let $S = \langle g_1, g_2, \cdots, g_{n-k} \rangle$ be generated by $n-k$ independent and commuting elements from $\mathcal{P}_n$, and such that $-I \notin S$. Then V_S is a 2^k dimensional vector space.*

Proof. Let $x = [x_1 \cdots x_{n-k}] \in \mathbb{F}_2^{n-k}$. We define,

$$P_S^x = \frac{\prod_{j=1}^{n-k} \left(I + (-1)_j^x g_j \right)}{2^{n-k}}.$$

Because $(I+g_j)/2$ is the projector onto the $+1$ eigenspace of g_j, $P_S^{(0,\cdots,0)}$ is the projector onto V_S. By lemma 6.3, for each x, there is a corresponding g_x such that $g_x P_S^{(0,\cdots,0)} g_x^\dagger = P_S^x$. Therefore, the dimension of every P_S^x is equal to the dimension of V_S. Furthermore, two different projectors are orthogonal,

$$\begin{aligned}
P_S^x P_S^y &= \frac{\prod_{j=1}^{n-k} (I + (-1)^{x_j} g_j)}{2^{n-k}} \frac{\prod_{j=1}^{n-k} (I + (-1)^{y_j} g_j)}{2^{n-k}}, \\
&= \frac{\prod_{j=1,i=1}^{n-k} (I + (-1)^{x_j} g_j)(I + (-1)^{y_i} g_i)}{2^{2(n-k)}}, \\
&= \frac{\prod_{j=1,i=1}^{n-k} (I + (-1)^{x_j} g_j + (-1)^{y_i} g_i + (-1)^{x_j + y_i} g_j g_i)}{2^{2(n-k)}}.
\end{aligned}$$

In this last equation, there will be a term with $i = j$ for which,

$$I + (-1)^{x_i} g_i + (-1)^{y_i} g_i + (-1)^{x_i + y_i} g_i g_i = I + (-1)^{x_i} g_i + (-1)^{1 - x_i} g_i - I = 0.$$

Finally, we find that,

$$I = \sum_x P_S^x.$$

The left-hand side is a projector onto a 2^n-dimensional space, while the right-hand side is a sum over 2^{n-k} orthogonal projectors of the same dimension as V_S. Therefore, the dimension of V_S must be 2^k. This concludes our proof. □

Lemma 6.4 implies that a stabilizer on n qubits requires only n generators to be uniquely determined. The generators we used in our examples confirm this property. We have now completely characterized stabilizer states. Our next step is to show how unitary evolution affects the stabilizer.

6.3.3 Stabilizer Evolution

Suppose the state of a quantum computer is described by the stabilizer $\langle g_1, \cdots, g_n \rangle$. This means that for all i, there exists a state $|\Psi\rangle$ such that $g_i|\Psi\rangle = |\Psi\rangle$. From the state vector approach, we know that a unitary operator U transforms a state $|\Psi\rangle$ into state $U|\Psi\rangle$. Simple algebra shows that,

$$Ug_i|\Psi\rangle = Ug_iU^\dagger U|\Psi\rangle,$$

so after the unitary transformation, the operator $Ug_iU^\dagger$ acts on state $U|\Psi\rangle$ just as g_i did on $|\Psi\rangle$. Therefore, applying a unitary operation to a stabilizer state transforms a generator $\langle g_1, \cdots, g_n \rangle$ into $\langle Ug_1U^\dagger, \cdots, Ug_nU^\dagger \rangle$.

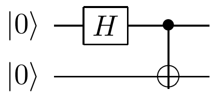
We must be cautious though; we limited the set of operators to elements of $\mathcal{P}_n$ in order to reduce the number of bits needed to describe the stabilizer. It is also clear that because there are a finite number of elements in $\mathcal{P}_n$, there are definitely only a finite number of stabilizers. Nonetheless, universal quantum circuits can prepare an infinite number of quantum states. The solution to this issue will be to limit the unitary evolutions in the stabilizer formalism to operators that leave $\mathcal{P}_n$ invariant. These operators, called the normalizer of $\mathcal{P}_n$, form a group too, the Clifford group.

The stabilizer formalism would not be attractive if the unitary operators in the Clifford group were rather arbitrary. The elegance of the stabilizer formalism comes from the fact that the Clifford group is generated by the Hadamard, Phase and CNOT gate. As we pointed out previously, only the $\pi/8$ or Toffoli gates are missing for universal quantum computation. How the absence of the latter two gates influences computational power will be explained in section 6.3.7.

Let us now compute how the generators of the Clifford group transform elements of the stabilizer. Using matrix multiplication one can show,

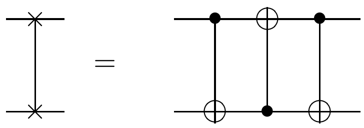
$$\begin{aligned}
H \quad &: \quad X \to Z \\
& \quad\;\; Z \to X \\
P \quad &: \quad X \to Y \\
& \quad\;\; Z \to Z \\
CNOT \quad &: \quad X \otimes I \to X \otimes X \\
& \quad\;\; I \otimes X \to I \otimes X \\
& \quad\;\; Z \otimes I \to Z \otimes I \\
& \quad\;\; I \otimes Z \to Z \otimes Z.
\end{aligned}$$

There is definitely an element of symmetry in these equations. Furthermore, note that it is also possible to implement all the Pauli gates too: $X = HP^2H$, $Y = PHP^2HP$, $Z = P^2$. Let us illustrate these techniques using some examples.



Example 6.4. Let's resist the urge to compute the effect of this circuit using the amplitude representations and instead calculate the evolution of the stabilizer. We know from a previous example that the stabilizer starts out as $\langle Z_1, Z_2 \rangle$. Applying a Hadamard gate to the first qubit transforms $Z_1 \to X_1$; the stabilizer becomes: $\langle X_1, Z_2 \rangle$. The second step applies a CNOT (qubit one is control, qubit two is target) transforming $X_1 \to X_1 X_2$ and $Z_2 \to Z_1 Z_2$; the stabilizer ends up as $\langle X_1 X_2, Z_1 Z_2 \rangle$ which we showed is the stabilizer of the Bell state. ⊠

Example 6.5. Suppose we want to know which operation the circuit above performs. However,



we do not want to assume a certain input state. Because $\mathcal{P}_2$ is generated by X_1, X_2, Z_1, Z_2, it is sufficient to check how the circuit transforms this stabilizer to know how it works on an arbitrary Pauli operator.

$$\begin{array}{rcl}
X_1 & = & X \otimes I \xrightarrow{CNOT1\to2} X \otimes X \xrightarrow{CNOT2\to1} I \otimes X \xrightarrow{CNOT1\to2} I \otimes X = X_2 \\
X_2 & = & I \otimes X \xrightarrow{CNOT1\to2} I \otimes X \xrightarrow{CNOT2\to1} X \otimes X \xrightarrow{CNOT1\to2} X \otimes I = X_1 \\
Z_1 & = & Z \otimes I \xrightarrow{CNOT1\to2} Z \otimes I \xrightarrow{CNOT2\to1} Z \otimes Z \xrightarrow{CNOT1\to2} I \otimes Z = Z_2 \\
Z_2 & = & I \otimes Z \xrightarrow{CNOT1\to2} Z \otimes Z \xrightarrow{CNOT2\to1} Z \otimes I \xrightarrow{CNOT1\to2} Z \otimes I = Z_1.
\end{array}$$

The circuit exchanges $X_1 \leftrightarrow X_2$ and $Z_1 \leftrightarrow Z_2$: it swaps the first and second qubit. ⊠

6.3.4 Stabilizer Measurements

We know how to describe stabilizer states and have derived a convenient way to update the stabilizer when a limited class of unitary operators is applied. To make the formalism complete, even measuring stabilizer states in the standard basis - using Pauli $\pm Z_i$ operators - can be done efficiently. Note first that the elements of the group $\mathcal{P}_n$ either commute or anticommute and because they are Hermitian, they can be regarded as observables. Suppose now that we wish to measure a state $|\Psi\rangle$ described by the stabilizer $\langle g_1, \cdots, g_n \rangle$ with observable $g \in \{\pm Z_i\} \subseteq \mathcal{P}_n$. There are two possibilities,

- g commutes with every element of the stabilizer,
- g anticommutes with one element of the stabilizer. Without loss of generality, we can assume that G anticommutes with only one element of the stabilizer, say g_1, if this weren't the case and G anticommutes with say g_1, g_j, than we can easily replace g_j in the generator by $g_1 g_j$. Now g anticommutes with g_1 and commutes with $g_1 g_j$, while the stabilizer remains identical.

The first case is computationally the most intensive. We know that for all g_i in the generator, $g_i g|\Psi\rangle = gg_i|\Psi\rangle = g|\Psi\rangle$. Therefore $g|\Psi\rangle \in V_S$ and thus must be a multiple of $|\Psi\rangle$. Because $g^2 = I$, $g|\Psi\rangle = \pm|\Psi\rangle$ whence either g or $-g$ must be in the stabilizer. If g is in the stabilizer, $g|\Psi\rangle = |\Psi\rangle$ and thus the measurement yields $+1$ with probability 1. If $-g$ were in the stabilizer, $-g|\Psi\rangle = |\Psi\rangle$ and the measurement would yield -1 with probability 1. Checking whether g or $-g$ is in the stabilizer is the most involved operation in the whole stabilizer formalism. In section 6.3.6 we will explain the most efficient algorithm known to date.

When g anticommutes with some element from the generator the following method shows us how to compute the measurement outcome. The eigenvalues of g are ± 1 and correspond to projectors $\frac{I \pm g}{2}$. The measurement outcome probabilities are given by,

$$\text{Pr[Measuring } g \text{ yields } +1] = \langle\Psi|\frac{I+g}{2}|\Psi\rangle,$$
$$\text{Pr[Measuring } g \text{ yields } -1] = \langle\Psi|\frac{I-g}{2}|\Psi\rangle.$$

Using the property that g anticommutes with g_i we find,

$$\begin{aligned}
\text{Pr[Measuring } g \text{ yields } +1] &= \langle\Psi|\frac{I+g}{2}|\Psi\rangle \\
&= \langle\Psi|\frac{I+g}{2}A_1|\Psi\rangle \\
&= \langle\Psi|A_1\frac{I-g}{2}|\Psi\rangle \\
&= \langle\Psi|\frac{I-g}{2}|\Psi\rangle \\
&= \text{Pr[Measuring } g \text{ yields } -1].
\end{aligned}$$

Therefore, Pr[Measuring g yields $+1$] = Pr[Measuring g yields -1] and equal to 1/2. If the measurement outcome was $+1$, the new state is $\sqrt{1/2}\frac{I+g}{2}|\Psi\rangle = \frac{I+g}{\sqrt{2}}|\Psi\rangle$. It is clear that $g, g_2, \cdots, g_n$ all stabilize $|\Psi\rangle$; moreover because g did not commute with one element of the previous generator, it was not in the stabilizer. Therefore it is independent from $g_2, \cdots, g_n$ and thus the new generator becomes $\langle g, g_2, \cdots, g_n\rangle$. A similar argument applies when the measurement outcome is -1 yielding the stabilizer $\langle -g, g_2, \cdots, g_n\rangle$.

6.3.5 Gottesman-Knill Theorem

The results from the previous sections combined can be summarized in the remarkable *Gottesman-Knill* theorem:

Theorem 6.4 (Gottesman-Knill)**.** *Suppose a quantum computation is performed which involves only the following elements: state preparation in the computational basis, Hadamard-gates, phase-gates, CNOT-gates, Pauli-gates and measurements of observables in the Pauli group. Such a computation may be efficiently simulated on a classical computer.*

6.3.6 Classical Simulation

In this section we will explain a classical algorithm to efficiently simulate stabilizer circuits that was developed by Scott Aaronson and Daniel Gottesman in [4]. This algorithm will be important to finally be able to characterize the computational power of stabilizer circuits.

The algorithm extends the concept of the check matrix and defines a *tableau*. A tableau not only lists the generators for a stabilizer state, but also the *destabilizer* and phase information. The destabilizer is defined as a group of generators which together with the stabilizer generators, generate the entire Pauli group $\mathcal{P}_n$.

$$\left(\begin{array}{ccc|ccc|c} x_{11} & \cdots & x_{1n} & z_{11} & \cdots & z_{1n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \cdots & x_{nn} & z_{n1} & \cdots & z_{nn} & r_n \\ \hline x_{(n+1)1} & \cdots & x_{(n+1)n} & z_{(n+1)1} & \cdots & z_{(n+1)n} & r_{n+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(2n)1} & \cdots & x_{(2n)n} & z_{(2n)1} & \cdots & z_{(2n)n} & r_{2n} \end{array}\right)$$

Rows 1 to n of the tableau represent the destabilizer generators $g_1, \cdots, g_n$ and rows $n+1$ to $2n$ represent the stabilizer generators $g_{n+1}, \cdots, g_{2n}$. The same notation as the check matrix is implied: let $g_i = A_1 A_2 \cdots A_n$; for every A_j,

- if $A_j = I$: $x_{ij} = z_{ij} = 0$,
- if $A_j = X$: $x_{ij} = 1, z_{ij} = 0$,
- if $A_j = Y$: $x_{ij} = 1, z_{ij} = 1$,
- if $A_j = Z$: $x_{ij} = 0, z_{ij} = 1$.

Note that the tableau has one column extra in comparison to the check matrix. This column will store phase information: $r_i = 0$ implies a positive phase whilst $r_i = 1$ implies a negative phase, imaginary phases are prohibited by observation 6.1. An example will clarify the notation.

Example 6.6. Most of the time, our algorithms start in the state $|0\rangle^{\otimes n}$. The example from section 6.3.2 already showed that the stabilizer for this state is generated by $\langle Z_1, Z_2, \cdots, Z_n\rangle$. We know that $ZX = Y$ and $Z^2 = I$. Therefore, $\langle X, Z\rangle$ generate all the Pauli operators. Therefore, we only need $\langle X_1, X_2, \cdots, X_n\rangle$ as the destabilizer in addition to $\langle Z_1, Z_2, \cdots, Z_n\rangle$ as the stabilizer to generate $\mathcal{P}_n$. Henceforth, the standard initial tableau becomes,

$$\left(\begin{array}{c|c|c} \mathrm{I} & 0 & 0 \\ \hline 0 & \mathrm{I} & 0 \end{array}\right)$$

⊠

This representation allows convenient simulation of the Clifford gates: CNOT, Hadamard and Phase. For example, suppose we want to simulate a Hadamard gate on qubit l. From section 6.3.3 we know that for every generator, it transforms X_l into Z_l, Z_l into X_l which implies that it leaves I_l, Y_l invariant. In the tableau representation, $x_{il} = 1, z_{il} = 0$ corresponds to X_l in the generator i. Transforming X_l to Z_l in generator i then corresponds to setting $x_{il} = 0, z_{il} = 1$. Performing this swtich for all $i \in [1, 2n]$ corresponds to switching column l with column $l+n$. Moreover, if I_l or Y_l would be present in the generator element, then $x_{il} = 0, z_{il} = 0$ or $x_{il} = 1, z_{il} = 1$ and simulating the Hadamard gate leaves them invariant.

A similar argument would show that simulating a phase gate on qubit l would correspond to binary adding column l to column $l+n$ and storing the result in column $l+n$. A CNOT gate with control qubit l and target qubit k corresponds to adding column l to column k, storing the result in column k and adding column $k+n$ to column $l+n$, storing the result in column $l+n$.

Before we describe the part of the algorithm that simulates the measurement we must describe a subroutine for another tableau manipulation procedure: RowSum. RowSum corresponds to the Pauli group operation. Suppose we want to multiply two generators corresponding to rows h, k and store the result in h. The hardest part of the group operation is to keep track of the phase information. We will shortly see how this should be done. On the other hand, it is relatively easy to calculate the new tableau entries for row h. Say row h and row k have a one in column j. This means that both have a Pauli X operator for qubit j. Because $X \cdot X = I$, the new entry should become 0. A similar reasoning applies to the part of the tableau that stores information on the Pauli Z operators and we find that binary addition performs the trick.

Back to the hard part. Let $g(x_1, z_1, x_2, z_2)$ be a function that returns the exponent to which i is raised when the matrices represented by $x_1 z_1$ and $x_2 z_2$ are multiplied:

1. $x_1 = z_1 = 0$: return 0,
2. $x_1 = z_1 = 1$: return $z_2 - x_2$,
3. $x_1 = 1, z_1 = 0$: return $z_2(2x_2 - 1)$,
4. $x_1 = 0, z_1 = 1$: return $x_2(1 - 2z_2)$.

Let us illustrate case 3 when $x_1 = 1, z_1 = 0$. This implies that the first Pauli matrix is X. There are four different possibilities for the second Pauli matrix:

- I, then the product is $X \cdot I = X = i^0 X$; therefore we must raise i to the exponent 0,
- X, then the product is $X \cdot X = I = i^0 I$; therefore we must raise i to the exponent 0,
- Y, then the product is $X \cdot Y = iZ = i^1 Z$; therefore we must raise i to the exponent 1,
- Z, then the product is $X \cdot Z = -iY = i^{-1} Y$; therefore we must raise i to the exponent -1.

It is now easy to check that $z_2(2x_2 - 1)$ returns the right exponent:

- if the second matrix was I, then $x_2 = 0, z_2 = 0$ and $z_2(2x_2 - 1) = 0$,
- if the second matrix was X, then $x_2 = 1, z_2 = 0$ and $z_2(2x_2 - 1) = 0$,
- if the second matrix was Y, then $x_2 = 1, z_2 = 1$ and $z_2(2x_2 - 1) = 1$,
- if the second matrix was Z, then $x_2 = 0, z_2 = 1$ and $z_2(2x_2 - 1) = -1$.

The algorithm RowSum(h, k) then becomes,

RowSum(h, k)
(1) **if** $2r_h + 2r_k + \sum_{j=1}^{n} g(x_{kj}, z_{kj}, x_{hj}, z_{hj}) \equiv 0 \mod 4$
(2) Set $r_h = 0$.
(3) **else if** $2r_h + 2r_k + \sum_{j=1}^{n} g(x_{kj}, z_{kj}, x_{hj}, z_{hj}) \equiv 2 \mod 4$
(4) Set $r_h = 1$.
(5) **foreach** $j \in [1, n]$
(6) Set $x_{hj} = x_{hj} \oplus x_{kj}$.
(7) Set $z_{hj} = z_{hj} \oplus z_{kj}$.

The phase is calculated in the first four lines of the algorithm: $2r_h + 2r_i$ corresponds to the previous exponents of i, $\sum_{j=1}^{n} g(x_{ij}, z_{ij}, x_{hj}, z_{hj})$ corresponds to the extra exponents that arise because of the group operation. Remembering observation 6.1 we know that no Pauli operator with a phase of $\pm i$ is in the stabilizer. Because a stabilizer is a subgroup of $\mathcal{P}_n$, applying the group operation (or RowSum) returns another element in the stabilizer. It follows that the total sum from step 1 or 3 can never equal $1, 3 \mod 4$.

We are now ready to give an algorithm that simulates the stabilizer circuit, SimStabilizer.

Input: A description of the stabilizer circuit that accepts $|0\rangle^{\otimes n}$ as input.
Output: The final tableau and possible measurement outcomes.
SimStabilizer($[G_1, G_2, \cdots, G_{p(n)})$
(1) **foreach** Clifford gate G_i
(2)   **if** G_i is a CNOT gate from control a to target b
(3)   **foreach** $j \in [1, 2n]$
(4)   Set $r_j = r_j \oplus x_{ja} z_{jb} (x_{ib} \oplus z_{ja} \oplus 1)$.
(5)   Set $x_{jb} = x_{jb} \oplus x_{ja}$.
(6)   Set $z_{ja} = z_{ja} \oplus z_{jb}$.
(7)   **else if** G_i is a Hadamard gate on qubit a
(8)   **foreach** $j \in [1, 2n]$
(9)   Set $r_j = r_j \oplus x_{ja} z_{ja}$.
(10)   Swap x_{ja}, z_{ja}.
(11)   **else if** G_i is a Phase gate on qubit a
(12)   **foreach** $j \in [1, 2n]$
(13)   Set $r_j = r_j \oplus x_{ja} z_{ja}$.
(14)   Set $z_{ja} = x_{ja} + z_{ja}$.
(15)   **else if** G_i is a measurement of qubit a in the standard basis
(16)   **if** There exists a $p \in [n+1, 2n]$ such that $x_{pa} = 1$
(17)   **foreach** $j \in [1, 2n], j \neq p, x_{ja} = 1$
(18)   RowSum(j, p).
(19)   Set the $p - n$'th row equal to the p'th row.
(20)   Set the p'th row to be identical 0 except $z_{pa} = 1$.
(21)   Set r_p equal to 0 or 1 with equal probability.
(22)   **else**
(23)   Add an extra row (2n+1)'th scratch row to the tableau and set it identical 0.
(24)   **foreach** $j \in [1, n], x_{ja} = 1$
(25)   RowSum$(2n+1, j+n)$.
(26)   Return r_{2n+1}.

Once we interpret the rows of the tableau as representing the generators of the stabilizer we can refer to the discussion above for the operation of RowSum, CNOT, Hadamard and Phase on the tableau.

The first case of the standard basis measurement is the one with a random outcome. When we measure qubit a in the standard basis, this corresponds to measuring the observable Z_a. From section 6.3.4 we remember that we have a random outcome when Z_a anticommutes with some generators. When generator p has an X or Y Pauli operator for qubit a, does the measurement anticommute; line 16 of the algorithm checks whether this is the case. Moreover, we remember from section 6.3.4 that if there are multiple anti-commuting elements in the generator, we can reduce them to one single anti-commuting generator element. This is done in lines 17 and 18 of the algorithm. Next we update the stabilizer and destabilizer and return a uniformly distributed outcome.

The second case of the standard basis measurement corresponds to a deterministic outcome where we must determine whether 0 or 1 is observed. Before we argue why the algorithm is correct, we define the *symplectic inner product* and state without proof some invariants of the algorithm.

Definition 6.5. *We define the symplectic inner product of two rows of the tableau as,*

$$r_h \cdot r_i = x_{h1}z_{i1} \oplus \cdots \oplus x_{hn}z_{in} \oplus x_{i1}z_{h1} \oplus \cdots \oplus x_{in}z_{hn}.$$

Note the similarity to lemma 6.1: the symplectic inner product is 0 if the two operators represented by the rows commute, and 1 if they anticommute.

Proposition 6.1. *The following are invariants of the tableau algorithm [4]*

- $r_1, \ldots, r_n$ *commute.*
- $\forall h \in \{1, \ldots, n\}$*,* r_h *anticommutes with* r_{h+n}
- $\forall i, h \in \{1, \ldots, n\}$ *such that* $i \neq h$*,* r_i *commutes with* r_{h+n}*.*

From the discussion in section 6.3.4, we know that Z_a commutes with all elements of the stabilizer when the measurement has a deterministic outcome. Using the third statement in proposition 6.1 this implies that,

$$\sum_{h=1}^{n} c_h r(h+n) = \pm Z_a, \tag{6.3}$$

with $r(i)$ the Pauli operator corresponding to the i'th row of the tableau. If we can determine the c_h's, by calculating the sum in equation (6.3), we can learn whether the phase of the right-hand side of the equation is positive or negative corresponding to the two possible measurement outcomes. Using linear algebra we can calculate these c_h's,

$$c_i = \sum_{h=1}^{n} c_h (r_i \cdot r_{h+n}) = r_i \cdot \sum_{h=1}^{n} c_h r_{h+n} = r_i \cdot Z_a \mod 2. \tag{6.4}$$

Therefore, if we check whether r_i anticommutes with Z_a – which it does if and only if $x_{ia} = 1$ (line 24) – then we learn the value of c_i and thus whether $\textsc{RowSum}(2n+1, i+n)$ needs to be called.

What about the complexity of this procedure? The space requirements of the algorithm consists of the tableau and a constant number of temporary variables only: the tableau has size $2n \times (2n+1) = O(n^2)$. For each quantum gate, the algorithm needs only a linear number of operations to update the stabilizer. A measurement with random outcome needs a linear number of updates too, the deterministic outcome needs a quadratic number of updates. Our next section shows that it isn't surprising that classical computers are able to simulate stabilizer circuits efficiently.

6.3.7 Computational Power

This is our final section on stabilizer circuits and we will use all the notions we have discussed previously to come to a characterization of the power of stabilizer circuits. Because complexity theory is mostly concerned with decision problems, our first task is to define a decision version that corresponds to the simulation of stabilizer circuits.

Definition 6.6 (GOTTESMAN-KNILL). *Given a stabilizer circuit, $\mathcal{C}$, will qubit 1 be in state $|1\rangle$ with certainty after $\mathcal{C}$ is applied to the initial state $|0\rangle^{\otimes n}$.*

It is clear from SIMSTABILIZER in our previous section that GOTTESMAN-KNILL $\in$ P. Even more is true! In this section, we will show that SIMSTABILIZER enables us to prove an even stronger inclusion: GOTTESMAN-KNILL $\in \oplus$L.

$\oplus$L is usually defined as the class of problems which are decided by a nondeterministic logspace Turing machine that accepts if and only if the total number of accepting paths is odd. Our next problem is a typical $\oplus$L-Complete problem: ODDPATHS.

Definition 6.7 (ODDPATHS). *Given a graph G, let,*

- *M be the adjacency matrix M_G of G,*
- *s a starting node,*
- *e an end node,*
- *k a positive integer.*

ODDPATHS *is the problem that asks whether there exists an odd number of paths of length k between s and e in graph G.*

ODDPATHS is definitely in $\oplus$L. We can build a nondeterministic Turing machine which stores a pointer that initialy points to the starting node s. At each step of the algorithm, the nondeterministic machine chooses a node from the set of all neighbors of the node the pointer is currently pointing to. It then updates the pointer to point to the chosen node. The Turing machine performs this step k times and accepts if the pointer points to e at the end, else it rejects. Because we only need to store a pointer to a node and a number to count up to k, this Turing machine clearly runs in logspace. Almost by definition, there are an odd number of paths of length k from s to e if and only if there are an odd number of accepting computations. Using the reachability method we can prove that ODDPATHS is a $\oplus$L-Complete problem.

Definition 6.8 (ITMATPROD). *Given a series of matrices $A^{(1)}, A^{(2)}, \cdots, A^{(n)}$ over $\mathbb{F}_2^n$ and integers $0 \leq i, j \leq n$. Is $[A^{(1)}A^{(2)}\cdots A^{(n)}]_{ij} = 1$?*

This problem is again a $\oplus$L-Complete problem. First of all, it is in $\oplus$L. From the definition of the matrix product we know that,

$$[A^{(1)}A^{(2)}\cdots A^{(n)}]_{ij} = \sum_{a,b,\cdots,z} A^{(1)}_{ia}A^{(2)}_{ab}\cdots A^{(n)}_{zj}.$$

A nondeterministic Turing machine can just guess a sequence $a, b, \cdots, z$ and multiply all the matrix elements in the sequence. If the product is one, it accepts, else it rejects. Because the sum is in $\mathbb{F}_2$, only if there are an odd number of accepting paths does the Turing machine accept. Because we only need to store an intermediate multiplication, clearly this procedure can be done in logspace. Furthermore, there exists an easy reduction from OddPaths to ItMatProd. If the tuple $\langle M, s, e, k\rangle$ describes an OddPaths instance, we convert it into an ItMatProd instance by setting every $A^{(i)} = M$ with $1 \leq i \leq k$ and ask whether the element (s, e) is 1. What is interesting about ItMatProd is that it is equivalent to a problem that is intimately connected to stabilizer circuits. Consider the problem of simulating a polynomial sized CNOT circuit:

Definition 6.9 (CNOT-NOT)**.** *Given a circuit with only CNOT-gates and NOT-gates, $\mathcal{C}$, will qubit 1 be in state $|1\rangle$ with certainty after $\mathcal{C}$ is applied to the initial state $|0\rangle^{\otimes n}$?*

Note first that this problem is a subproblem of ItMatProd because the matrix representation of a CNOT-gate or NOT-gate is a matrix over $\mathbb{F}_2^n$. Therefore, calculating whether qubit 1 is in state $|1\rangle$ can be seen as solving ItMatProd with CNOT-gates and NOT-gates substituted for the matrices $A^{(i)}$. There is one small issue that must be solved though. ItMatProd expects the matrices $A^{(i)}$ as its input. Unfortunately, writing down an $n \times n$ matrix requires a polynomial amount of space. The solution to this problem is easy: every time the Turing machine for ItMatProd accesses an element $A^{(i)}_{kl}$, we insert a small logspace subroutine that calculates $A^{(i)}_{kl}$. This proves that CNOT-NOT$\in \oplus\mathsf{L}$. We briefly sketch how one would prove that CNOT-NOT is complete for $\oplus\mathsf{L}$. There are n^2 different CNOT or NOT gates on n qubits. These matrices are elements of a vector space; moreover, there exists an inner product on this n^2 dimensional vector space of matrices called the *Hilbert-Schmidt* inner product relative to witch these matrices are orthogonal and therefore form a basis. This implies that every matrix over $\mathbb{F}_2^n$ can be written as a linear combination of n^2 CNOT or NOT matrices. Since, calculating the inner product can be performed in logarithmic space, we conclude that every ItMatProd instance can be reduced to CNOT-NOT.

So far we have proved that CNOT-NOT is $\oplus\mathsf{L}$-Complete. Because CNOT-NOT is a subset of Gottesman-Knill, this implies that Gottesman-Knill is $\oplus\mathsf{L}$-Hard. In order to prove that Gottesman-Knill is $\oplus\mathsf{L}$-Complete, we must also prove that Gottesman-Knill$\in \oplus\mathsf{L}$. We refer to the full proof to [4] but we will discuss the idea here. As we discussed in the section on how to simulate CNOT, Hadamard and Phase gates, the whole algorithm boils down to calculating the sum of some variables modulo 2. Therefore, a CNOT-NOT oracle can perform this simulation. However, calculating the phase is a highly nonlinear procedure. Nonetheless, in [4] the authors present a nontrivial algorithm which runs in logspace that gradually computes the phase by calling a CNOT-NOT oracle too. Finally, because CNOT-NOT is in $\oplus\mathsf{L}$ and $\oplus\mathsf{L} = L^{\oplus\mathsf{L}}$ this concludes the proof that Gottesman-Knill$\in \oplus\mathsf{L}$.

This concludes our characterization of the computational power of stabilizer states. It is conjectured that $\oplus\mathsf{L} \subset \mathsf{P}$ which means that stabilizer circuits are probably not even universal for classical computation. Nonetheless, the mathematical elegance of stabilizer circuits and its detailed characterization in terms of computation complexity make it an important theoretical discovery. It is rather remarkable how the absence of one gate makes such a difference in computational power. This is not new however, in section 2.4 we saw how the Hadamard gate is the only gate that is needed to increase the computational power of P to BQP. Our next section limits quantum computation in a different way and steps up a little in the complexity theoretic hierarchy.

6.4 P-Blocked Quantum Algorithms

The biggest problem for simulating quantum algorithms is that, modulo normalization, all of the 2^n components of the superposition can have independent probability amplitudes. When the algorithm needs to update the state after a quantum gate is applied, we might need to access arbitrary probability amplitudes.

A result by Jozsa and Linden, primarily intended to clarify the role of entanglement in quantum computing, limited the scope of the entanglement across the qubits to circumvent the problem above. In [21], the authors introduce a new subset of quantum states: the *p-blocked* states. A state is p-blocked if it can be written as a product state of components with at most p entangled qubits. More in particular, an n qubit state $|\Psi\rangle$ is p-blocked if for some constant p, there exist a partition $B = B_1 \cup B_2 \cup \cdots \cup B_n$ of the qubits with $|B_i| \leq p$ and $|\Psi\rangle = |\Psi_{B_1}\rangle \otimes |\Psi_{B_2}\rangle \otimes \cdots \otimes |\Psi_{B_k}\rangle$ with $|\Psi_{B_i}\rangle$ a superposition consisting of qubits in B_i. These superposition $|\Psi_{B_i}\rangle$ can be written as,

$$|\Psi_{B_i}\rangle = \sum_{x \in \{0,1\}^{|B_i|}} a_x |x\rangle. \tag{6.5}$$

The states $|\Psi_{B_i}\rangle$ can be described by at most 2^p probability amplitudes which is a constant in the number of qubits. We can now state the result by Jozsa and Linden:

Claim 6.1. *If a quantum computation runs in polynomial time with the property that at each stage of the computation, the state of the computation is p-blocked, an efficient classical simulation exists.*

The algorithm proposed by Jozsa and Linden is very similar to QSIMI but because of the p-blocked condition, it is more efficient in space and time resources. Because every component of the partitioning needs at most a constant number of rational numbers to describe the probability amplitudes, and there are at most n elements in the partitioning, the space requirements for the algorithm are $O(n)$. What about the time requirements to calculate the updates?

Let us call the state of the quantum computation at step t, $|\alpha_t\rangle$ and its corresponding qubit partitioning $B^t = B_1^t \cup B_2^t \cup \cdots \cup B_n^t$. First of all, we assume that we use a universal gate set with single qubit gates and CNOT. Therefore, there are two possible unitary transformations possible at stage t:

- The unitary transformation at stage t accesses qubits from a single component of the partition. W.l.o.g. we assume that this component of the partitioning is B_1^t.
- The unitary transformation at stage t accesses qubits from two different partitions. W.l.o.g. we assume that these components of the partitioning are B_1^t, B_2^t.

The first case is the easiest one. The probability amplitudes of $|\Psi_{B_2}\rangle, \cdots, |\Psi_{B_k}\rangle$ remain invariant as no qubit of those partition elements is involved under the operation of the quantum gate. The matrix corresponding to the quantum gate on the partition B_1 can be computed and will be at most a 2^p by 2^p matrix, again a matrix of constant size. Next, our simulation calculates the product of the transformation matrix with the probability amplitude vector. This requires 2^p vector products which again can be done in constant time. Summarizing, updating one component of the partition under a transformation can be done in constant time.

What about the second case? It is actually not much different from the first case if we merge the two partitions first. This results in a partition with 2^{2p} probability amplitudes and 2^{2p} by 2^{2p} transformation matrices operating on it. The extra difficulty arises after the transformation matrix is applied to the probability amplitudes. If $|B_1^t| + |B_2^t| < p$, we are done. On the other hand, if $|B_1^t| + |B_2^t| > p$, the premise states that it is possible to partition the qubits $B_1^t \cup B_2^t$ in at least two new sets. Let $P : Q$ be a bipartitioning such that,

$$\begin{aligned}\left(p_1|p_1\rangle + p_2|p_2\rangle + \cdots + p_{|P|}\left|p_{|P|}\right\rangle\right)&\left(q_1|q_1\rangle + q_2|q_2\rangle + \cdots + q_{|Q|}\left|q_{|Q|}\right\rangle\right) \\ &= \left(b_1|p_1q_1\rangle + b_2|p_1q_2\rangle + \cdots + b_{|P||Q|}\left|p_{|P|}q_{|Q|}\right\rangle\right).\end{aligned}$$

Suppose we know which qubits belong to P and which ones to Q. The problem reduces to finding the headers of the rows and columns of the following table,

	p_1	p_2	$\cdots$	$p_{\lvert P\rvert}$
q_1	$b_1 = p_1q_1$	$b_{\lvert Q\rvert+1} = p_2q_1$	$\cdots$	$b_{(\lvert P\rvert-1)\lvert Q\rvert+1} = p_{\lvert P\rvert}q_1$
q_2	$b_2 = p_1q_2$			$\vdots$
$\vdots$				
$q_{\lvert Q\rvert}$	$b_{\lvert Q\rvert} = p_1q_{\lvert Q\rvert}$	$\cdots$		$b_{\lvert P\rvert\lvert Q\rvert} = p_{\lvert P\rvert}q_{\lvert Q\rvert}$

Table 6.1: Probability distribution table.

If we normalize the first column, it is equal to vector $[q_1 \quad q_2 \quad \cdots \quad q_{|Q|}]$ up to a global phase factor. Similarly, normalizing the first row makes it equal to vector $[p_1 \quad p_2 \quad \cdots \quad p_{|P|}]$ up to a global phase. Note that it does not matter which value the global phase factor has because a quantum state is invariant under a global phase transformation.

Unfortunately, we assumed that we knew the correct bipartitioning $P : Q$ which is not the case. Luckily, because there are only a constant number of qubits, there are only a constant number of bipartitionings; we can thus perform an exhaustive search of all possible bipartitionings, calculate amplitude vectors for all of them and then check whether their product correspond to the original amplitudes: FindPartition.

Input: A set of maximally $2p$ qubits B with corresponding amplitudes.
Output: A partitioning such that the state represented by the amplitudes is a tensor product of the state represented by the amplitudes of the partitioning.
FindPartition($[b_1, b_2, \cdots, b_{|P|,|Q|}]$)
foreach Bipartitioning $P : Q$ of B
  $n_p = \sum_{i=1}^{|P|} |b_{1i}|^2$.
  $n_q = \sum_{i=1}^{|Q|} |b_{i1}|^2$.
  $[p_1 \cdots p_n] = [b_{1i} \cdots b_{1n}]/n_p$.
  $[q_1 \cdots q_n] = [b_{i1} \cdots b_{n1}]/n_q$.
  foreach $i \in [1, |P|], j \in [1, |Q|]$
    if $b_{ij} \neq p_iq_j$
      break
  return $\left(P : Q : [p_1 \quad p_2 \quad \cdots \quad p_{|P|}] : [q_1 \quad q_2 \quad \cdots \quad q_{|Q|}]\right)$

We are now left in a situation where computations in a p-blocked state at every step are efficiently simulatable on a classical computer. Another way to interpret this result is that a necessary, though not sufficient, condition for a quantum algorithm to provide an exponential speedup is that its states cannot be p-blocked for at least one step. In other words, using the formal definition of entanglement, multipartite entanglement across the qubits must be unbounded. For example, in [21] the authors show how Shor's factorization algorithm conforms to the latter statement.

6.4.1 Computational Power

What is interesting about p-blocked quantum algorithms is that they are equivalent in power to the complexity class P. First of all, if a problem can be solved with a p-blocked algorithm, our simulation argument above proves that there exists a polynomial time algorithm that solves the problem to. On the other hand, it is not hard to see that every language in P can be decided with a p-blocked algorithm too. Suppose $L \in \mathsf{P}$ and $\mathcal{C}$ is a polynomial sized classical circuit that decides L. We already pointed out that every circuit can be made reversible, so let $\mathcal{C}^r$ be the reversible classical circuit that decides L. Remembering claim 3.1, we know that the circuit $\mathcal{C}^r$ has a quantum equivalent $\mathcal{C}^q$ of polynomial size. The only issue that must be addressed is that the state of $\mathcal{C}^q$ must be p-blocked. The state of the reversible deterministic circuit can never exists in a superposition, therefore if it is in say state $x_1 x_2 \cdots x_n$, the state of the equivalent quantum circuit will be $|x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$. This state is clearly a 1-blocked state and therefore, for any $p > 0$, p-blocked.

6.4.2 Final Words on P-Blocked States

The reason why p-blocked algorithms can be simulated in polynomial time is because p-blocked states can be described by $\mathsf{poly}(n)$ parameters in contrast to the $O(2^n)$ parameters needed for a general quantum state. In other words, p-blocked states are characterized by only polynomial degrees of freedom which in turn are able to generate 2^n probability amplitudes.

Nonetheless, the fact that states with a polynomially sized description correspond to the class of unentangled states depends strongly on our mathematical formalism! Let's give two examples. First of all, suppose we would use simple Hilbert spaces (section 5.2) to describe our algorithms. Only using polynomial degrees of freedom would correspond to some property of the probability distribution implied by the superposition. Polynomial degrees of freedom do not correspond to entanglement in this formalism. A second example are the stabilizer states: they only require a polynomial number of bits to describe quantum states. Here it is even more clear that polynomial degrees of freedom correspond to a class of states totally different from the p-blocked states: the stabilizer states. As we have showed many times, there are stabilizer states which are highly entangled.

Unfortunately, it is not so that all quantum computations which states can be described with a polynomial amount of resources can be simulated efficiently. The updates between the gates might take superpolynomial time to compute.

6.5 Open Problems, Further Research and Conclusion

This chapter introduced us to several concepts that are useful in the study of quantum complexity theory. In this last section we point out several open problems that might lead to interesting insights. Wherever possible, we suggest possible approaches to solve these problems.

Let us first point out a weakness of the stabilizer formalism. We pointed out that because there are only a finite number of generators on n qubits, there must be only a finite number of stabilizer states on n qubits. Contrast this with for example p-blocked states which have a continuous range of possible states if we forget our assumption that amplitudes must be rational. I have tried to extend the theory of stabilizers to include one or more real degrees of freedom. Unfortunately I did not find a representation of continuously valued matrices with a group structure that was stabilized by the Clifford group. On the other hand, I would like to try to apply the theory of Lie-groups and see if it would be a way forward. It would be interesting to see whether, let us call them *continuous stabilizers*, increase the computational power of the formalism.

I have also studied another subset of universal quantum computation that is very similar to the p-blocked formalism of Jozsa & Linden. In [20], Guifre Vidal proposes a representation of slightly entangled quantum states based on the notion of the Schmidt decomposition of quantum states. It is very clear from that representation that again a polynomial number of degree of freedom generate 2^n probability amplitudes.

What was interesting with the stabilizer formalism was that we could design any quantum circuit we wanted as long as we used CNOT, Hadamard or Phase gates. It is therefore a little disappointing that p-blocked and Vidal computation do not imply such a convenient characterization. E.g., it is not clear how one could design a quantum algorithm where each intermediary state is p-blocked. Nonetheless, it would be interesting to see whether such a characterization exists for p-blocked or Vidal states but so far I have found no evidence that it is so.

Related to p-blocked and Vidal states is the issue of polynomial degrees of freedom. One can think of many formalisms which given:

- a polynomial number of variables $\tilde{a} = [a_1 \cdots a_{p(n)}]$,
- an exponential number of functions $\phi_1(\tilde{x}), \cdots \phi_{2^n}(\tilde{x})$ with $\tilde{x} = [x_1 \cdots x_{p(n)}]$,

represent quantum states. Every function ϕ_i returns the probability amplitude of some n-qubit basis state. Another question which might provide interesting insights is what conditions on the ϕ_i are necessary such that we can simulate a unitary evolution in polynomial time. Fix a particular basis state $|x\rangle$. We know that $\phi_x(\tilde{a})$ returns the probability amplitude of $|x\rangle$. Suppose we apply a unitary transformation on the state described by $\tilde{a}$. The new probability amplitude of $|x\rangle$ becomes,

$$\phi_x(\tilde{a'}) = \sum_y U_{yx}\phi_y(\tilde{a}).$$

Updating the state therefore corresponds to solving certain equations. It might be worthwhile to look into different classes of ϕ_x that lead to interesting subsets of general quantum computation.

Finally there exists a number of other subsets of quantum states [31]. One of the most promising subsets are the so called *tree states*. They provide an excellent connection to computational complexity theory but introduce a whole new range of problems to be solved.

Quantum complexity theory is definitely an area where much is to be discovered. On one hand, we can be optimistic: quantum mechanics sometimes simplifies things [33]. On the other hand, there are a lot of parallels between probabilistic and quantum computing. Both allow exponential degrees of freedom in the description of their states yet there is clear evidence that they will be different: it is conjectured that $\mathsf{BPP} = \mathsf{P}$ while FACTORINGD provides evidence that $\mathsf{P} \subset \mathsf{BQP}$. The difficulty of proving the $\mathsf{BPP} = \mathsf{P}$ case might feed the pessimism that a solution will take a while to be discovered. Nonetheless, my research has provided me with some valuable insights in the one difference in dynamics between probabilistic and quantum computation: destructive interference. I hope these insights allow me to make a contribution to the final characterization of BQP's complexity in the future.

Appendix A

The Density Operator

Usually, quantum states are described using the language of state vectors. Unfortunately, this paradigm has some deficiencies and a more powerful tool to describe quantum states exists: the density matrix formalism. This formalism represents a quantum state with an operator called the *density operator*. If the state of a quantum system is precisely known, as in the state vector approach, we say that it is a *pure state*. The density matrix formalism also allows what are called *mixed states* which are not representable by a state vector.

Suppose a quantum system is in a one of a number of states $|\Psi_i\rangle$, each with probability p_i. We call the set $\{p_i, |\Psi_i\rangle\}$ with $\sum p_i = 1$ an *ensemble of pure states*. If there is more than one $p_i > 0$, the state is a mixed state. The density operator or *density matrix* of an ensemble of pure states is defined by the equation,

$$\rho = \sum_i p_i |\Psi_i\rangle\langle\Psi_i|. \tag{A.1}$$

Example 1.1. Suppose we have an ensemble $\{1/2, |00\rangle; 1/2, |11\rangle$. Its density matrix is,

$$1/2|00\rangle\langle 00| + 1/2|11\rangle\langle 11| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Contrast this with the density matrix of the pure quantum state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$,

$$1/2|00\rangle\langle 00| + 1/2|00\rangle\langle 11| + 1/2|11\rangle\langle 00| + 1/2|11\rangle\langle 11| = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

⊠

It turns out that all of quantum mechanics can be represented in the density matrix formalism. Moreover, it is true that the density matrix contains all the information that can be known about a quantum state. This was not the case for pure states which can't represent an ensemble. Nonetheless, for most of quantum computing, state vectors suffice, but is not uncommon in quantum information theory to use the density matrix formalism because of its extra power. We will now show some interesting properties of density matrices and introduce the necessary concepts in order to understand the use of density matrices in this thesis.

Theorem A.1. *Suppose we apply a unitary evolution U on a quantum state described by density matrix ρ. After the evolution the new density matrix $\rho' = U\rho U^\dagger$.*

Proof. Suppose the quantum state is in a pure state $|\Psi_i\rangle$ with probability p_i, then after the evolution the state is $U|\Psi_i\rangle$ with probability p_i. Therefore, the new density matrix becomes,

$$\begin{aligned}\rho' &= \sum_i p_i U|\Psi_i\rangle\langle\Psi_i|U^\dagger \\ &= U\left(\sum_i p_i|\Psi_i\rangle\langle\Psi_i|\right)U^\dagger \\ &= U\rho U^\dagger\end{aligned}$$

□

Suppose we perform a measurement with projectors P_i on an ensemble $\{p_i, |\Psi_i\rangle\}$ and we want to know the probability of measuring m. If the initial state was $|\Psi_i\rangle$, then,

$$\Pr[\text{measurement returns } m \mid \text{state was i}] = \langle\Psi_i|P_m|\Psi_i\rangle = tr\left(P_m|\Psi_i\rangle\langle\Psi_i|\right).$$

By the law of total probability,

$$\begin{aligned}\Pr[\text{measurement returns } m] &= \sum_i \Pr[\text{Measurement returns } m \mid \text{state was i}]p_i \\ &= \sum_i \operatorname{tr}\left(P_m|\Psi_i\rangle\langle\Psi_i|\right)p_i \\ &= \operatorname{tr}\left(P_m\sum_i p_i|\Psi_i\rangle\langle\Psi_i|\right) \\ &= \operatorname{tr}\left(P_m\rho\right).\end{aligned}$$

Example 1.2. Suppose we have an ensemble $\{1/4, |00\rangle; 1/4, |01\rangle; 1/4, |10\rangle; 1/4, |11\rangle$ with density matrix $I/4$. Take an arbitrary observable A with a decomposition in projectors P_i. It is easy to see that,

$$\Pr[i] = \operatorname{tr}(P_i I/4) = 1/4\operatorname{tr}(P_i) = 1/4\operatorname{tr}(|e_i\rangle\langle e_i|) = 1/4\langle e_i|e_i\rangle = 1/4,$$

with e_i the eigenvector corresponding to projector P_i. This state is known as the *maximally mixed state*: *any* measurement returns a uniform distribution of all possible outcomes. ⊠

Simple algebra is sufficient to develop all the properties of density matrices from elementary quantum mechanics. For our purpose, we need one final concept: the *reduced density matrix*. Density matrices are an extremely powerful tool for describing subsystems of composite quantum systems. The concept of a reduced density matrix is indispensable in the analysis of composite quantum systems. Suppose we have physical systems A and B, whose state is described by a density matrix ρ^{AB}. The reduced density matrix for system A is,

$$\rho^A = \operatorname{tr}_B\left(\rho^{AB}\right),$$

where tr_B is a map of operators known as the partial trace. The effect of the partial trace is: $\operatorname{tr}_B\left(|a\rangle\langle a| \otimes |b\rangle\langle b|\right) = |a\rangle\langle a|\operatorname{tr}\left(|b\rangle\langle b|\right) = |a\rangle\langle a|\langle b|b\rangle$. In addition, the partial trace is linear in its argument.

What makes the partial trace important is that it describes the state of a subsystem. If the joint state of quantum system is described by density matrix ρ^{AB}, the state of system A is described by $\mathrm{tr}_B\left(\rho^{AB}\right)$. It is not entirely obvious why this is so, and it is beyond the scope of this text to prove this more formally. An example will show its importance, though.

Example 1.3. Suppose Alice and Bob share an EPR pair, $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, with density matrix

$$1/2|00\rangle\langle 00| + 1/2|00\rangle\langle 11| + 1/2|11\rangle\langle 00| + 1/2|11\rangle\langle 11| = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Alice possesses the first qubit while Bob possesses the second. Alice's reduced density matrix is,

$$\begin{aligned} &\mathrm{tr}_B\left(1/2|00\rangle\langle 00| + 1/2|00\rangle\langle 11| + 1/2|11\rangle\langle 00| + 1/2|11\rangle\langle 11|\right) \\ &= 1/2\left(\mathrm{tr}_B(|00\rangle\langle 00|) + \mathrm{tr}_B(|00\rangle\langle 11|) + \mathrm{tr}_B(|11\rangle\langle 00|) + \mathrm{tr}_B(|11\rangle\langle 11|)\right) \\ &= 1/2\left(|0\rangle\langle 0| + |1\rangle\langle 1|\right) \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

This clearly shows that Alice has a maximally mixed qubit. Every measurement she makes on her qubit will return a uniform distribution of all possible measurement outcomes. It would be as if she would toss a coin to get the measurement outcome; in other words, no measurement returns any information on the state of the EPR pair. ⊠

As a final word on density matrices we want to point out that they have very good analytical properties. For example, density matrices make it convenient to introduce the notion of entropy. Without getting into details, we mention that the entropy of the maximally mixed state is maximal. This means that from the point of view of Alice in our last example, she knows nothing about the joint quantum state.

An extensive coverage of density matrices and their properties can be found in [28].

Appendix B

Nederlandse Samenvatting

De 100 jaar oude theorie van de kwantummechanica heeft de laatste 25 jaar voor een revolutie in ons beeld van computers gezorgd. Sinds de jaren 80 werd het duidelijk dat er een verband bestaat tussen wat een computer kan berekenen en de wetten van de klassieke fysica. Het verband impliceert dat fysische systemen een computer kunnen simuleren en omgekeerd, dat een computer een klassiek fysisch systeem kan simuleren. Het was dan ook logisch dat onderzoekers zich de vraag gingen stellen wat de implicaties zouden zijn als het klassiek fysisch systeem vervangen zou worden door een kwantummechanisch systeem.

Eind jaren 80 en begin jaren 90 werd er een een robuust theoretisch model voor een kwantum computer ontwikkeld maar de grote doorbraak kwam er pas in 1994. In dat jaar ontwikkelde Peter Shor een polynoomtijd algoritme om natuurlijke getallen te factoriseren. Het factorisatieprobleem is niet alleen van praktisch belang in de cryptografie, maar neemt ook een sleutelrol in in de complexiteitstheorie. Men vermoedt namelijk dat er geen efficiënt klassiek algoritme bestaat om een getal in priemfactoren te ontbinden. Het daaropvolgende jaar ontdekte Lov Grover een nieuw kwantum algoritme om een element te zoeken in een niet gesorteerde rij van n elementen met slechts $\Theta(\sqrt{n})$ queries.

Sinds deze ontdekkingen zijn er slechts enkele nieuwe kwantum algoritmes gevonden. De vraag die zich opdringt, is waarom het zo moeilijk blijkt te zijn om nieuwe kwantum algoritmes te vinden. Een mogelijk antwoord zou kunnen zijn dat er misschien helemaal niet veel problemen zijn die een kwantum computer sneller kan oplossen dan een klassieke computer. Om dit probleem verder te onderzoeken stellen we ons in deze thesis de vraag op welke manier een kwantum computer verschilt van een klassieke computer. Twee typische kwantum fenomenen komen hiervoor in aanmerking: interferentie en verstrengeling.

Na een inleiding over kwantum computers en een bespreking van de kwantum fenomenen, bestuderen we of Shor's algoritme en Grover's algoritme gebruik maken van interferentie en verstrengeling. We komen tot de conclusie dat dit inderdaad het geval is als we onze informatie voorstellen met kwantum bits; de kwantum tegenhangers van klassieke bits. Een kwantum algoritme kan tussentijdse metingen bevatten die op hun beurt de volgende stappen van het algoritme kunnen bepalen. De interpretatie van het meetresultaat en de beïnvloeding van de volgende stappen van het algoritme gebeuren met een klassiek algoritme. Hierdoor zal de analyse van een kwantum algoritme rekening moeten houden met evoluties in de Hilbertruimte afgewisseld met een aantal stappen van een klassiek algoritme. We tonen echter aan dat elke meting van een kwantum systeem helemaal tot op het einde van het algoritme uitgesteld kan worden. Deze vereenvoudiging laat ons toe om heel het algoritme te analyseren in de toestandsruimte van het kwantum systeem en op het einde slechts een enkele meting te moeten interpreteren. Omdat we in het algemeen onze informatie voorstellen met verschillende kwantum bits is de Hilbertruimte van het hele kwantum systeem een tensorproduct van verschillende tweedimensionale Hilbertruimtes. We tonen aan dat deze tensorproduct-Hilbertruimte isomorf is met een eenvoudige Hilbertruimte; een Hilbertruimte die geen tensorproduct is van kleinere Hilbertruimtes. We besluiten dus dat *elk* kwantum algoritme kan geanalyseerd worden in deze eenvoudige Hilbertruimtes. In deze context is enkel interferentie van belang; verstrengeling is een ongedefinieerd begrip.

Aan de andere kant tonen we aan dat als we een Hilbertruimte in een fysiek systeem willen inplanten, we verstrengeling nodig hebben om de scaleerbaarheid van de kwantum computer te verzekeren. Dit suggereert dat enkel de fysicus of ingenieur die een kwantum computer bouwt, rekening moet houden met verstrengeling.

Onze conclusie geeft een kwalitatief beeld van de rol van interferentie en verstrengeling voor kwantum computers. Helaas geeft dit weinig inzicht in de kracht van kwantum computers. Daarom sluiten we onze thesis af met een inleiding op de karakterisatie van de kracht van kwantum computers en hoe deze gerelateerd is met de hiërarchie van bekende complexiteitsklassen. Omdat dit probleem van een veel hogere moeilijkheidsgraad is, bieden we hier geen concrete oplossingen aan maar bespreken we bestaande inzichten en suggereren een aantal onderzoeksmogelijkheden voor de toekomst.

Appendix C

Symbols

$A \subset B$ - A is a proper subset of B.
$A \subseteq B$ - A is a subset of B or A is equal to B.
$|\Psi\rangle$ - A vector Ψ.
$\langle\Psi|$ - A dual vector Ψ.
$\langle\Psi|\Phi\rangle$ - The scalar product between $|\Phi\rangle$ and $\langle\Psi|$.
δ_{ab} - The Kronecker delta: if $a = b \Rightarrow \delta_{ab} = 1$, if $a \neq b \Rightarrow \delta_{ab} = 0$.
a^* - The complex conjugate of scalar a.
A^* - The complex conjugate of matrix A.
A^T - The transpose of matrix A.
$A^\dagger$ - The complex conjugate of the transpose of matrix A.
$\mathbb{F}_n$ - The finite field with n elements.
σ_x - Pauli X matrix.
σ_y - Pauli Y matrix.
σ_z - Pauli Z matrix.
$[A, B]$ - The commutator of A and B, $AB - BA$.
$\{A, B\}$ - The anti-commutator of A and B, $AB + BA$.
$\oplus$ - The binary addition operator.
$\otimes$ - The tensor product operator.
$|\Theta_+\rangle$ - Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$.
$|\Theta_-\rangle$ - Bell state $\frac{|00\rangle-|11\rangle}{\sqrt{2}}$.
$|\Phi_+\rangle$ - Bell state $\frac{|01\rangle+|10\rangle}{\sqrt{2}}$.
$|\Phi_-\rangle$ - Bell state $\frac{|01\rangle-|10\rangle}{\sqrt{2}}$.

Bibliography

[1] A. Ambainis: *Quantum Search Algorithms*, SIGACT News Complexity Column, 2004, *arXiv e-print quant-ph/0504012*

[2] arXiv, *http://www.arxiv.org*

[3] L. Adleman, J. DeMarrais and M. Huang: *Quantum computability*, SIAM Journal on Computing, 26(5):1524-1540, 1997

[4] S. Aaronson and D. Gottesman: *Improved Simulation of Stabilizer Circuits*, 2004, *arXiv e-print quant-ph/0406196*

[5] A. Y. Kitaev: *Quantum Computations: Algorithms and Error Correction*, Russ. Math. Surv., 52(6):1191-1249, 1997

[6] A. Barenco, C.H. Bennet, R. Cleve, D.P. DiVincenzo, N. Nargolus, P. Shor, T. Sleator, J. Smolin and H. Weinfurter: *Elementary Gates for Quantum Computation*, Phys. Rev. A, 52:3457-3467, 1995, *arXiv e-print quant-ph/9503016*

[7] R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf: *Quantum Lower Bounds by Polynomials*, IEEE Symposium on Foundations of Computer Science, 352-361, 1998, *arXiv e-print quant-ph/9802049*

[8] D. Bouwmeester, A. Ekert and A. Zeilinger: *The Physics of Quantum Information*, Springer, 2001

[9] M. Boyer, G. Brassard, P. Høyer and A. Tapp: *Tight Bounds on Quantum Searching*, 1996, *arXiv e-print quant-ph/9605034*

[10] G. Brassard, P. Høyer, M. Mosca and A. Tapp: *Quantum Amplitude Amplification and Estimation*, 2000, *arXiv e-print quant-ph/0005055*

[11] E. Bernstein and U. Vazirani: *Quantum Complexity Theory*, SIAM J. Comput., 26(5):1411-1473, 1997, *arXiv e-print quant-ph/9701001*

[12] C. Caves, I.H. Deutsch and R. Blume-Kohout: *Physical-resource demands for scalable quantum computation*, 2003, *arXiv e-print quant-ph/0304083*

[13] C. Papadimitriou: *Computational Complexity*, Addison-Wesley, 1994

[14] D. Aharonov: *A Simple Proof that Toffoli and Hadamard are Quantum Universal*, 2003, *arXiv e-print quant-ph/0301040*

[15] D. Gottesman: *The Heisenberg Representation of Quantum Computers*, 1998, *arXiv e-print quant-ph/9807006*

[16] D. Deutsch and R. Jozsa: *Rapid Solution of Problems by Quantum Computation*, Proceedings of the Royal Society of London, Series A 439:553-558, 1992

[17] D. A. Meyer: *Quantum Games and Quantum Algorithms*, 2000, *arXiv e-print quant-ph/0004092*

[18] D. van Melkebeek: *CS 880 Lecture Notes: Quantum Computing*, 2002, *http://www.cs.wisc.edu/ dieter/Courses/CS880-s2002/*

[19] A. Einstein, B. Podolsky and N. Rosen: *Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?*, Phys. Rev., 47:777-780, (1935)

[20] G. Vidal: *Efficient Classical Simulation of Slightly Entangled Quantum Computations*, Phys. Rev. Lett. 91:147902, 2003, *arXiv e-print quant-ph/0301063*

[21] R. Jozsa and N. Linden: *On the Role of Entanglement in Quantum Computational Speed-Up*, 2002, *arXiv e-print quant-ph/0201143*

[22] J. Preskill: *Physics 219 Lecture Notes: Quantum Computing*, 2001

[23] V. Kendon and W. Munro: *Entanglement and its Role in Shors Algorithm*, 2004, *arXiv e-print quant-ph/0412140*

[24] A.Y. Kitaev, A.H. Shen and M.N. Vyalyi: *Classical and Quantum Computation*, American Mathematical Society, 2002

[25] L. Fortnow: *One Complexity Theorist's View of Quantum Computing*, Theoretical Computer Science, 292(3):597-610, 2003

[26] M. Hirvensalo: *Quantum Computing*, Springer-Verlag, 2001

[27] M. Sipser: *Introduction to the Theory of Computation*, PWS Publishing Company, 1997

[28] M. Nielsen and I.L. Chuang: *Quantum Computation and Quantum Information*, Cambridge University Press, 2000

[29] R. Orús and J. Latorre: *Universality of Entanglement and Quantum Computation Complexity*, 2003, *arXiv e-print quant-ph/0311017*

[30] P. Shor: *Why haven't more quantum algorithms been found?*, Journal of the ACM, 50(1):87-90, 2003

[31] S. Aaronson: *The Complexity Zoo*, *http://www.complexityzoo.com*

[32] S. Aaronson: *Is Quantum Mechanics an Island in Theoryspace*, 2004, *arXiv e-print quant-ph/0401062*

[33] S. Aaronson: *Quantum Computing, Postselection, and Probabilistic Polynomial-Time*, 2004, *arXiv e-print quant-ph/0412187*

[34] S. Lloyd: *Quantum Search Without Entanglement*, Phys. Rev. A, 61:010301, 1999, *arXiv e-print quant-ph/9903057*

[35] S. Lloyd: *Ultimate Physical Limits to Computation*, Nature, 406:1047-1054, 2000

[36] U. Schönig: *A Probabilistic Algorithm for k-SAT and constraint satisfaction problems*, IEEE Symposium on Foundations of Computer Science, 410-414, 1999