VERTIEFUNGSARBEIT – Communication Systems Group, Prof. Dr. Burkhard Stiller

University of Zurich[UZH]

An Overview of Quantum Mechanics with a Focus on Quantum Computation

Leon Philip Sanders
Zürich, CH
Student ID: 07-610-397

Supervisor: Thomas Bocek, Ben Keitch
Date of Submission: April 10, 2014

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Contents

Chapter 1

Introduction

Quantum computation is a relatively new and exciting field of both computer science and physics. It is born out of quantum mechanics. More specifically the great Richard Feynman noted in the beginning of the 1980s [Fey82] that classical computers would not be able to efficiently simulate physical systems perfectly. In this paper a glimpse will be given why that is. The solution he proposed is quantum simulation. Subsequently base on his ideas researchers proposed a computer run by the effects used in said quantum simulation. What a quantum computer is will be the main topic of this paper. In order to try to understand this though it is necessary to get an understanding of the most basic quantum mechanics. A superficial explanation of it will constitute the first part of this paper. Four important laws of quantum mechanics follow. The basic building of a quantum computer, the qubit is described next. Lastly single qubit gates will be described and a short introduction into quantum computational complexity will be given.
Taking a step back first and looking at classical computers it seems that sooner or later Moore's law will come to an end. There is a physical limit to the number of transistors on a chip. Moore himself stated that the limit would be reached within 20 years[Moo05]. Therefore it is worth thinking about new approaches to computation in general. In 1994 Peter Shor gave the field motivation: He showed how a quantum algorithm could outperform a classical one in the field of factorization [NC10].

Chapter 2

Related Work

2.1 Quantum Mechanics

In order to get a deeper introduction into the field, one may consult one of the standard works on quantum mechanics e.g. Sakurai [ST94] in which a comprehensive and detailed overview of the topic is given. In order to understand quantum mechanics in general a good understanding of complex numbers is essential: Yanofsky [YM08] gives a refresher on this topic in chapter 2. Unitary matrices which play an important role in quantum mechanics are treated there as well. Alistair M Rae et al [Rae04] gives an overview of some more advanced parts of the quantum theory . Of course these subfields have many more works associated to them but this essay is more concerned with an overview.

2.2 Quantum Computation

Even though the study of quantum computation is relatively young there is quite a lot of study connected to it. A thorough introduction into the topic aimed at physics students and computer scientists alike is given by Nielsen and Chuang [NC10]. This work is regarded as the standard work of quantum computation. The book goes into details of topics not directly related to this thesis but the first two chapters are a very thorough introduction. The founding document of quantum computation by Richard Feynman should be noted as well [Fey82]. There are texts aimed more at the uninitiated in regards to physics such as [YM08] and [RP11] which help the reader gain a basic understanding.

Chapter 3

Quantum Physics

3.1 Brief History of the Field of Quantum Mechanics

Quantum physics is the study of light, atoms and particles as well as their interactions. Physics in the 17th century was dominated by newtonian mechanics. For various reasons newton believed in the idea of light consisting of particles. Other researchers, among them Descartes and then Huygens showed evidence for light acting as a wave on the basis of refraction [Pai92]. This was not widely accepted until 1801 when Thomas Young showed with his double-slit-experiment that light indeed acted in a way that can only be explained by light behaving as a wave.

In 1900 Lord Rayleigh and Sir James Jeans discovered that there was a law governing blackbody-radiation, the electromagnetic radiation from an idealized physical body absorbing all radiation at all frequencies at a given temperature. Even though it worked well for low frequencies, it revealed an important error in classical physics: When the wavelength approaches zero the law predicts an infinite amount of energy to be emitted [Pai79].

This discrepancy was called the ultraviolet catastrophe. Max Planck explained this in the same year by assuming the energies of an electrons oscillations have to be proportional to multiples of the frequency.

$$E = n \cdot h \cdot f$$

where E is the energy of said electron, n is an arbitrary integer representing the number of electrons and f is its frequency [She01]. As a result of this equation Planck could derive experimentally that the factor $\hbar$ is constant at $6.626 \times 10^-34 J \cdot s$ [She01]. Therefore energy only comes in discrete packets or quanta. Planck himself was reluctant to accept these quanta as they interfered with the interpretation of classical physics and its Continuum Theory [Hei79]. Another problem of physics in the early twentieth century was the photoelectric effect: Ultraviolet light can cause electrons to be ejected from a metal surface. In contrast to the predictions of classical physics this effect was shown to be dependent on the frequency of the light instead of its amplitude [Pai79]. Einstein modified Planck's formula so that the radiation itself would consist of packets of energy: $E = h \cdot f$. These

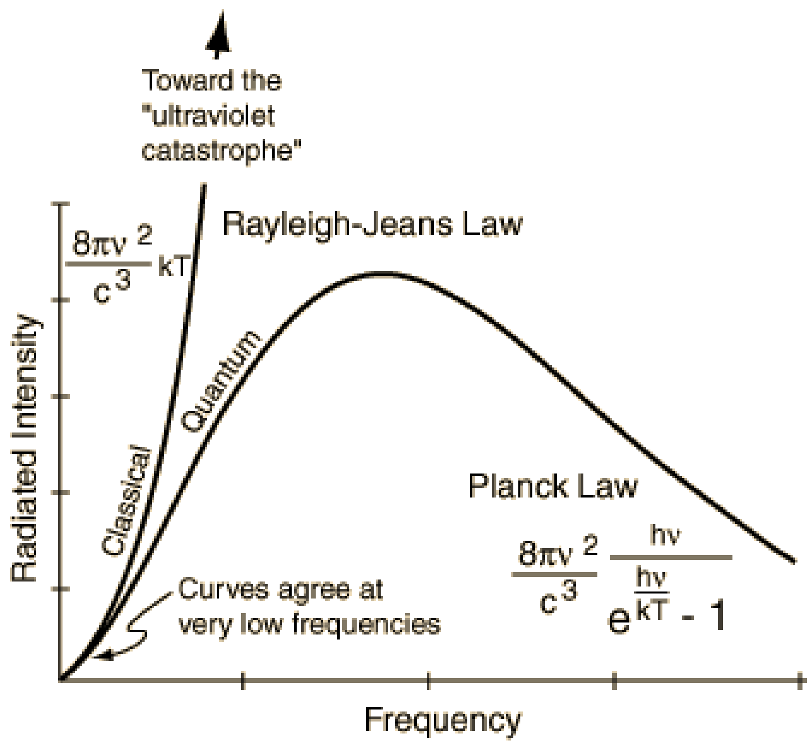

Figure 3.1: It is shown how the classical prediction of electromagnetic radiation fails for high frequencies [oPG00]

packets are now known as photons whereas Einstein called them "Lichtquanta" [Pai79]. The value of h obtained was close to that of Planck's. Later on Einstein could show that even the oscillations of atoms around their equlibrium positions in crystals are quantized [She01].
The frequency times the wavelength specifies the speed of a wave. If now $\lambda f = c$ for an electromagnetic wave, then in conjunction with above equation it follows that $E = \frac{h \cdot c}{\lambda}$ [She01]. Using Einstein's relativity result $E = mc^2$ it was found that $\lambda = \frac{h}{mc}$ [She01]. It is important to note that the mass here is not the restmass of the photon, which is zero, but its relativistic mass meaning the energy contained in the photon. Hence a photon has a wavelength and discrete packets of energy (hf). Therefore a photon behaves both as a wave and as a particle which is now called the Wave-Particle duality. de Broglie reasoned that this may also be true for matter and it was experimentally proven by bombarding metals with electrons instead of photons [Pai92]. Heisenberg could show that this leads to an interesting consequence: He proposed that it was not possible to simultaneously know the exact position and momentum of a particle. If one visualizes a particle it is impossible to know its momentum without measuring its position at two distinct times over a period of time [Hei79]. Werner Heisenberg expressed this relationship with the uncertainty principle $\Delta x \cdot \Delta p \geq \frac{\hbar}{2}$ where $\hbar = \frac{h}{2\pi}$ (in accordance with [She01]). Heisenberg continued to work on the problem, at the same time as Erwin Schödinger and the two of them developed the basis for modern quantum mechanics. They developed equations to describe how the quantum state of a physical system changes over time. The partial differential Schrödinger-equation is the basis for what is now known as the wave function

[She01].

Shortly thereafter Max Born proposed that the predictions of this wave function can be understood as probability amplitudes [Pai79]:

$$\int_X |\psi(x)|^2 d\mu(x) < \infty; \tag{3.1}$$

or in normalized form:

$$\int_{-\infty}^{\infty} |\psi(x)|^2 d\mu(x) = 1;$$

The factor $|\psi(x)|^2$ describes a probability density and is the tool which makes it possible for physicists to work with the properties of quantum mechanics and complex probabilities and ultimately is the basis for a quantum computer [Hun75].

The problem with trying to understand quantum mechanics through the historical perspective is that is is inherently confusing. Some of its founders thought the theory counter-intuitive. Planck was reluctant to accept his own findings, Einstein did not fully accept it until his death and called it "black magic calculus"[NC10], Schrödinger thought up the Gedankenexperiment with his cat to mock the paradoxes of quantum theory and Richard Feynman is famously quoted as saying "I can safely assume that nobody understands quantum theory" [She01].

3.2 Understanding Quantum Mechanics through Programming

So if the predictions of quantum mechanics (QM) confused even some of the founders of the theory and Richard Feynman was still uncomfortable with it in the 80s probably another approach should be taken than trying to understand it by taking a historical route. In order to do so, some experiments carried out in the early 80s by Alain Aspect will be abstracted. He could prove that photons that do not have any physical connection can still influence each other [AGR+82] which is now called entanglement.

In Section 3.2 there is an experiment shown in which a half-silvered mirror called a beam-splitter (BS) is used, which only allows 50% of photons to be transmitted while the other 50% get reflected [Ala08]. Additionally there is a light source (S) and two detectors (D1/D2), which detect incoming photons. In accordance with Feynman there will be an initial state of the depicted experiment as well as configurations meaning a particular path of a photon through the system [Fey82]. For example, this initial state could be that a photon is heading to the beam splitter.

Computer scientists feel at home with programs so the above experiment will be represented as one (see Figure 3.2). These complex numbers associated with the configurations

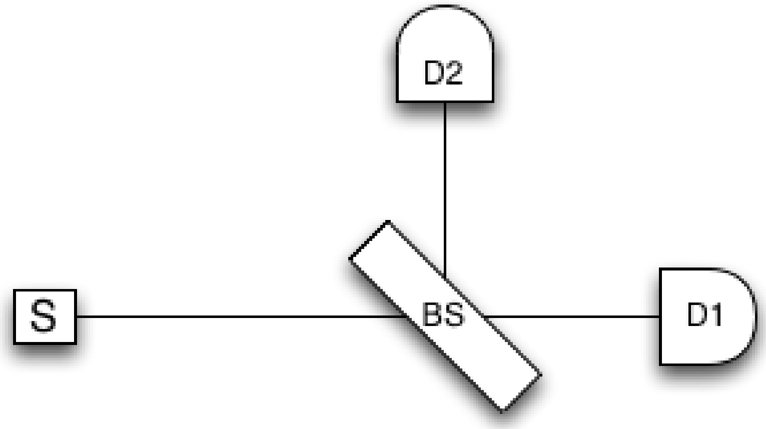

Figure 3.2: The most basic interferometric experiment is shown here with just one beam-splitter.

are not classical probabilities they are probability amplitudes. This means that the complex number describes a (complex) wave and what one studies is its time-evolution [YM08]. For a slightly more through explanation the reader is directed to the beginning of chapter 5 where the Dirac notation is described. The only way one can access the probability of an observable by squared modulus (see Equation 3.1). The possible configurations in the above experiment are:

0. A photon going from S to BS
1. A photon going from BS to D1
2. A photon going from BS to D2

Configuration 0 has a normalized complex amplitude of $\frac{(-1+0i)}{\sqrt{2}}$. This value is arbitrary the only constraint for now is that it is not zero. The amplitudes of configurations 1 and 2 are not set yet they get assigned while the program runs. Mathematically what the BS does is multiplying the incoming configuration by 1 if the photon goes straight or by i if the photon turns 90 degrees. It was mentioned that the amplitudes have to normalized. This means that the squared modulus summed of all the factors must add up to 1 (see Law 1 of important laws of quantum mechanics below). This is basically to say that photon observed is bound to be at a certain place but cannot be in multiple ones. Just as the summed probabilities of a dice have to add up to one this is true for quantum systems as well. One can see this in the program illustrating the experiment the amplitudes of the above configurations.

```python
"""Interferometric experiments illustrated through programming"""
import cmath

class Mirror(object):
    """A reflective mirror"""
    def reflect(self, input):
        try:
            return input*(0+1j/cmath.sqrt(2))
        except TypeError:
            print("Complex Amplitude expected!")


class BeamSplitter(Mirror):
    """A Beam-Splitter is a half-silvered mirror"""
    def let_through(self, input):
        try:
            return input*(1/cmath.sqrt(2)+0j)
        except TypeError:
            print("Complex Amplitude expected!")
```

When the first experiment is run with the following parameters:

```python
def main():
    """The function running the Experiments"""
    #Experiment 1
    exp1 = Experiment()
    bs = BeamSplitter()

    #arbitrary initial value
    config0 = (-1 + 0j)
    exp1.add_config(config0)

    #Photon goes straight: config1
    config1 = bs.reflect(config0)
    exp1.add_config(config1)
    #Photon takes a right angle: config2
    config2 = bs.let_through(config0)
    exp1.add_config(config2)
    print("Exp1:")
    exp1.print_configs()

 #...

if __name__ == "__main__":
```

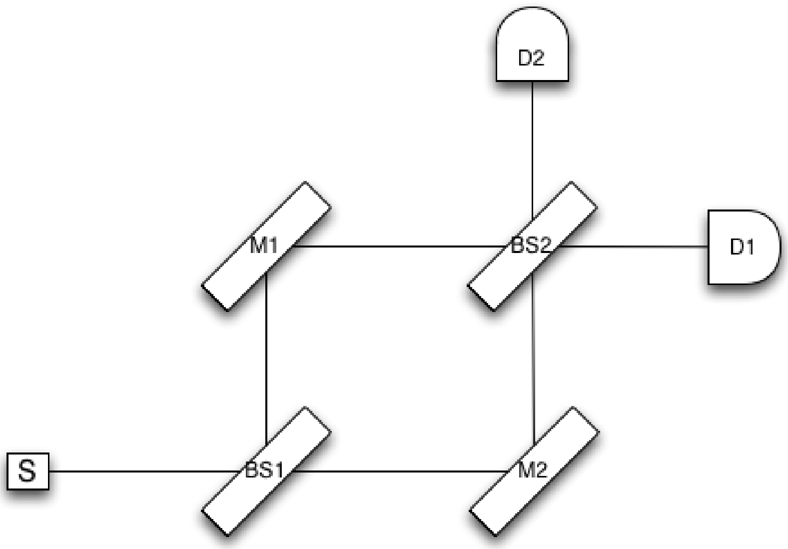

Figure 3.3: The most basic interferometric experiment is shown here with just one beam-splitter.

```
    main()


        ...
```

it produces this output:

```
Exp1:
Configuration 0 has Amplitude (-1+0j) and a squared modulus of: 1.0
Configuration 1 has Amplitude (-0-1j)/sqrt{2} and a squared modulus of: 0.5
Configuration 2 has Amplitude (-1+0j)/sqrt{2} and a squared modulus of: 0.5
```

There is no way of measuring directly what the exact amplitudes of a given configuration are. Fortunately physicists have devised an instrument to measure it using the squared modulus (see Equation 3.1).

Figure 3.3 shows another sightly more complicated experiment: There are full mirrors (M1/M2) involved now. For our purposes mirrors work similarly to BS but reflect all the photons. Therefore the amplitude flowing from an incoming photon to an outgoing one at a 90° angle is multiplied by i. It is important to note that the paths the photon can take have to be exactly the same (P1: $BS1 \rightarrow M1 \rightarrow BS2 \rightarrow D1$ == P2: $BS1 \rightarrow M2 \rightarrow BS2 \rightarrow D2$). Consequently there is a new set of possible configurations:

0. A photon going from S to BS1

1. A photon going from BS1 to M1
2. A photon going from BS1 to M2
3. A photon going from M1 to BS2
4. A photon going from M2 to BS2
5. A photon going from BS2 to D1
6. A photon going from BS2 to D2

This is the representation of the experiment in programmatic form:

```
...

 #Experiment 2
 exp2 = Experiment()
 m = Mirror()

 #First threee configurations stay the same as in Experiment 1
 exp2.add_config(config0)
 exp2.add_config(config1)
 exp2.add_config(config2)

 #The incoming configurations are relfected by the mirrors
 config3 = m.reflect(config1)
 exp2.add_config(config3)
 config4 = m.reflect(config2)
 exp2.add_config(config4)

 #The amplitude flows going corresponding
 #to configuration 5 and 6 are added up
 config5 = bs.reflect(config4) + bs.let_through(config3)
 exp2.add_config(config5)
 config6 = bs.reflect(config3) + bs.let_through(config4)
 exp2.add_config(config6)


 print("Exp2:")
 exp2.print_configs()

 #...
```

The first three configurations have the same amplitudes as the one's above. Configuration 3 and 4 are multiplied by i as can be seen in the corresponding program. Configurations 5 and 6 are very interesting though: The incoming amplitudes (3, 4) are multiplied by the BS as in 3.2. The total amplitude flowing to configuration 6 is $\frac{(2+0i)}{\sqrt{2}}$. The total amplitude flowing to configuration 5 is 0 because the two possible configurations are just added up as can be seen in the corresponding program. In contrast to classical systems in quantum mechanics the path the photon takes does not matter as much as the outcome. Therefore no photon will ever reach D2. This effect is called destructive interference and is due to the phase of the paths canceling each other out.

Therefore the output of the program must be:

```
Exp2:
Configuration 0 has Amplitude (-1+0j) and a squared modulus of: 1.0
Configuration 1 has Amplitude (-0-1j) and a squared modulus of: 0.5
Configuration 2 has Amplitude (-1+0j) and a squared modulus of: 0.5
Configuration 3 has Amplitude (1-0j) and a squared modulus of: 0.5
Configuration 4 has Amplitude (-0-1j) and a squared modulus of: 0.5
Configuration 5 has Amplitude (2+0j) and a squared modulus of: 1.0
Configuration 6 has Amplitude 0j and a squared modulus of: 0.0
```

The full program is enclosed in the appendix. Here the "quantum gap" [Nie08] as Michael Nielsen calls it really is apparent. This outcome is just not explainable with an intuition stemming from classical physics. They are just the way nature behaves as has been shown experimentally. And that is what the little programs illustrating the experiments are: Reality at the level of photons according to our current model of physics.

These predictions should not be confused with probabilities. With classical probabilities one can create sets of similar outcomes such as the even numbers on a dice. Whether one calculates the sum of probabilities for those even numbers on their own and then adds them up or make a set of them does not change the ensuing probability because with classical probabilities $F(x + y) = F(x) + F(y)$. In quantum mechanics because one talks about probability amplitudes and because one uses the squared modulus this is not the case. $|x + y|^2 \neq |x|^2 + |y|^2$. Hence it is possible that amplitudes cancel each other out as in the experiment shown. This effect is called interference in classical wave mechanics as well as quantum mechanics[YM08]. This is one of the less intuitive aspects of quantum mechanics as in our everyday intuition one woudl expect reality to be made up of individually real parts. In contrast the waves in QM can interact with each other.

In a quantum system its evolution depends among others on the second derivative of the amplitude distribution [Hun75]. The laws of physics describe how amplitude distributions develop into new amplitude distributions. A decent approximation from the viewpoint of a computer scientist may be Conway's Game of Life (CGoL): The future state of a cell depends on it's neighboring cells. QM has an inherent property of non-locality. One

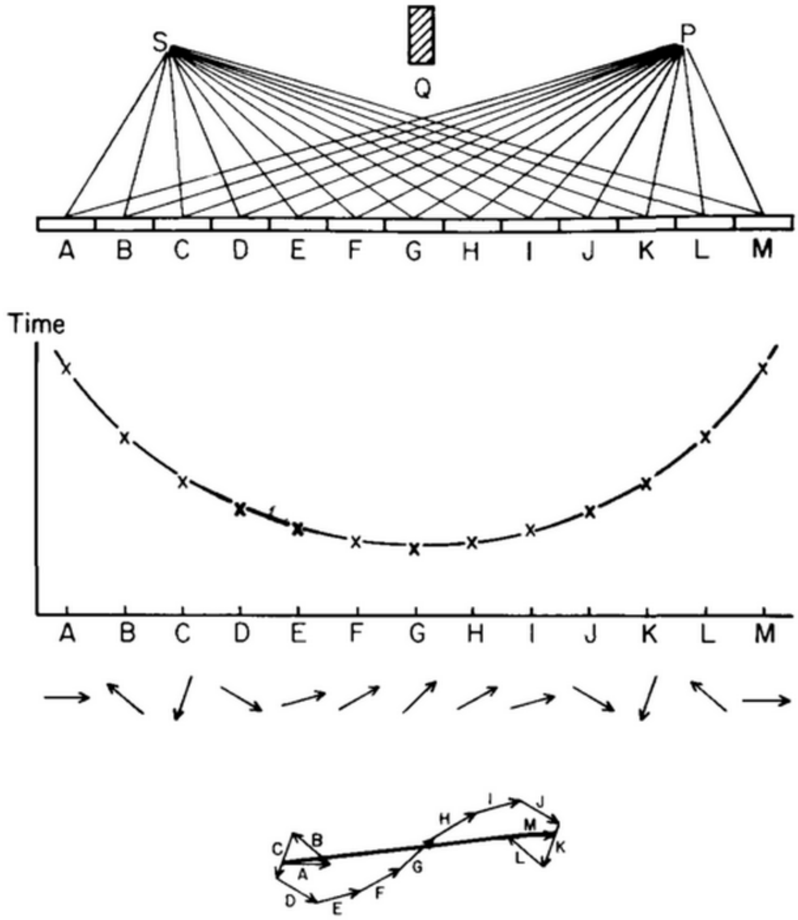

Figure 3.4: The figure shows the abstraction of a mirror and photons hitting it [Fey06]

can determine the change within a configuration using it's infinitesimal neighborhood as one did above in the little programs, because one only used what happened right before, to determine what would happen to the photons next. Just as in CGoL one cannot ask what the future state of a particle will be without taking into account its neighbors. The difference though is that this neighbor does not have to be close at all. Two particles on the opposite side of the universe may influence each other [YM08].

3.2.1 Feynman Path Integrals

In his book QED: The Strange Theory of Light and Matter [Fey06] Richard Feynman describes how a mirror is an example of the interesting fact that amplitudes which flow to the same configuration are added up: One would think that the angle of incidence of a photon equals it's angle of reflection. He describes though that in order to find the amplitude of a photon going from S to P in figure 3.4 one has to add up all the possible paths this could happen as can be seen at the top of figure 3.4. The graph in the middle shows the time it takes for the photon to go from S to P. This way of visualizing QM is called "Feynman Path Integrals". The differences between adjacent paths in the middle of the graph, i.e around G are smaller than on the edges. This is to say that the gap in

time between the paths via A and B is bigger than the one between the paths of via F and G respectively. If one adds up the times it is apparent that most of the amplitude comes from the middle of the mirror [Fey06]. At the bottom of 3.4 one can see how at the edges the photons mostly cancel each other out. Amplitudes are still flowing from all these configurations. But if one calculate the total amplitude it is pretty much the same for both the middle part of the mirror and the whole mirror. That is the reason it seems that the angle of incidence is the same as the angle of reflection.
Knowing this one can actually make a mirror which does not reflect at the angle of incidence. It is possible to scrape away the parts of the mirror that are out of phase and cancel each other out. For example one could keep A and scrape B. This is called a diffraction grating [Fey06]. A mirror changed in this way does not reflect in a normal way but produces little rainbows of color.

3.2.2 Important Laws of Quantum Mechanics

In order to make up for the incompleteness of the above explanation the following important postulates of Quantum Mechanics will be used, out of which the reader may only begin to understand Nr. 2 and 3 based on the explanations above. It is important to note though that these relatively simple postulates constitute the basis for all of quantum mechanics and its strangeness [BBBV97].

Law 1 Quantum mechanics is a linear theory: one can create linear superpositions of wave functions, provided one keep the probability amplitudes normalized. [BBBV97][YM08]

Law 2 The quantum measurement postulate can be described as the wave function collapsing to the basis state corresponding to the outcome of the experiment. [Fey06]

Law 3 One cannot discover the full quantum state of a system, only the squared probability amplitudes $|\alpha|^2$. The α are the projections of the system onto the basis states and are complex-valued. [NC10]

Law 4 One cannot clone an unknown quantum state. [BBBV97][She01]

These laws have far-reaching consequences. To describe them all would exceed the scope of this text manifold. A basic example of the first Law can be found in the description of the ket notation in the following section. Every quantum state can be written as a superposition of the base state. Its identity is given by the complex weights. The measurement we took in the little programs to derive probabilities collapsed the complex amplitude of the configuration as is postulated by Law 2 and 3. Law 4 has not been dealt with at all and has been included for completeness as it is important for a more complex description of quantum computation and especially quantum cryptography [YM08].

Chapter 4

Quantum Computation

Quantum Computation is fundamentally different from classical computing. Classical bits are either 0 or 1. In quantum computing one works with quantum bits. An arbitrary quantum state can be written as $|\psi\rangle$. This notation is called Dirac Ket Notation [YM08]. One may imagine that we have a vector for every point in space x_0 to x_{n-1}. Because if every point is covered individually we could write vector x_0 as a column vector $[1, 0, ..., 0]^T$ and x_{n-1} as $[0, 0, ..., 1]^T$. These vectors must form a basis [BBBV97]. These vectors can also be written as $|x_0\rangle$ to $|x_{n-1}\rangle$. Considering a quantum universe though one must acount for the experimentally proven complex weights and interference shown above. This is why complex amplitudes c_0 to c_{n-1} are needed in front of every vector. Therefore

$$|\psi\rangle = c_0|x_0\rangle + ... + c_{n-1}|x_{n-1}\rangle$$

The Dirac notation is just an easier way of describing this. Because of the wave nature of particles (see de Broglie) one may think of the above equation as n waves, $|x_0\rangle$ to $|x_{n-1}\rangle$, that all contribute with intensity c_i to $|\psi\rangle$. This is the reason we speak of the wave function. This infinite set of vectors is called a Hilbert Space. One such set of vectors may be thought of as a configuration which was used above. The configurations in the two illustrating programs were abstracted and one would have to have many more variables to represent an actual configuration because not every particle in the experiment was accounted for. The above equation represents the wave function used in quantum mechanics.

4.1 Quantum Bits

Quantum bits or qubits are the basic building blocks of a quantum computer. A quantum bit represents a 2-state quantum system [BBBV97] just as a classical bit can have two states. The difference is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$. Every qubit (often denoted $|\psi\rangle$) is in a superposition of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{4.1}$$

The qubit may be represneted in the following way as a matrix:

$$|\phi\rangle = \alpha \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

In programmatic form:

```
import numpy as np
def qubit(alpha, beta):
    return alpha * np.array([[0],[1]]) +
           beta * np.array([[1],[0]], 'D')
```

α and β represent complex numbers as are used in the illustrating experiments and code snippets above. One can think of them as vectors in a two-dimensional vector space. The exact value of α and β cannot be measured (see Law 2). According to 4.1 above a qubit has to collapse into either state 0 or 1. Due to Law 3 this happens with probability $|\alpha|^2$ or $|\beta|^2$ [NC10]. $|\alpha|^2 + |\beta|^2 = 1$ according to the first law of QM. This means that the aforementioned vector is normalized to 1 and therefore one deals with the unit vector in a 2D complex vector space.

The indirect correspondence mentioned above is what makes quantum computation so difficult to fathom: As [NC10] says our intuition tells us that for example a coin can be either heads or tails (assuming a perfect coin that cannot land on its edge). In quantum computation a qubit is in all states between and including 0 and 1 until it is measured.

This leads us to an interesting question: How much information is contained within one qubit? Obviously it seems it would be an infinite amount. A qubit though will always collapse to either 0 or 1. So how much information is contained until one measures? Naturally this is the wrong question as without measuring one do not know but it hints at one of the aspects potentially making quantum computation so powerful: Nature keeps track of all continuous variables describing the state such as α and β until one measure it [DiV95].

One important aspect of the measurement that should be noted is that as soon as it is made the qubit is changed: If a 0 is measured the qubit will thereafter be in state $|0\rangle$ [NC10]. One can therefore not obtain any additional information about the qubit by repeating the measurement. Why this collapse occurs nobody knows.

Although this may seem counter-intuitive or strange, qubits are real. Their properties have been validated by a number of experiments [DiV95][MMK$^+$95][NC10].

Until now this essay just dealt with a single qubit. Of course to have at least some computational power one would need quantum computers with more than one qubit. Quantum computers today often have eight to 16 qubits [YM08]. In quantum computation though there is a lot more that can be done with a single qubit than with a single bit: A quantum gate can already be formed out of a single qubit. These will be described below but unfortunately the description of multiple qubits as well as the one of the corresponding gates would exceed the focus of this paper.

4.2 Gates and Circuits

4.2.1 Single qubit Quantum Gates and Quantum Circuits

A classical computer is built out of wires and logic gates. As an example the only non-trivial logic gate the NOT gate is defined by its truth table. What would a quantum NOT gate look like? Such a gate would have to take $|0\rangle$ and change it to $|1\rangle$ and the other way around. Such a gate though would not tell us anything about the superpositions of the state. It would just interchange them:

$$\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|1\rangle + \beta|0\rangle \tag{4.2}$$

Therefore a quantum NOT gate would act linearly on the given qubit [NC10]. This is due to Law 1 above and is well motivated empirically. Often quantum gates are represented as matrices. As $|\alpha|^2 + |\beta|^2$ must be 1, and this must also be true for the state after using the gate, there are restrictions on the kind of matrices that can be used: The only matrices that may be used are unitary [NC10] i.e. the dot product of the gate and its adjoint equals the identity matrix. One can easily verify that the NOT gate fulfills this requirement. Interestingly this is the only constraint. Any unitary matrix is a valid quantum gate. Even more interesting for a computer scientist may be that they are reversible as well:

$$V = UU^*V = IV$$

The representation of the quantum NOT looks like this:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

We can write the quantum state as a column vector such as:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Hence the output of the quantum NOT gate will be:

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Or in programmatic form:

If the NOT-gate is applied to the qubit it switches the amplitudes. As said before it is reversible as all quantum gates are. The equalities below show that the NOT-gate multiplied by its self-adjoint and in turn multiplied by a qubit again equals the qubit, hence the reversibility of the NOT-gate is shown.

```
#arbitrary complex amplitudes
(t_plus_one, t_minus_one) = np.dot(qubit(alpha, beta), X)
(not_equality, t_equality) = \
        np.dot(dot(X, self_adjoint(X)), qubit(alpha, beta))
(all(qubit(beta, alpha) == np.dot(qubit(alpha, beta), X)),
 all(qubit(alpha, beta) == \
        np.dot(dot(X, self_adjoint(X)), qubit(alpha, beta))))
```

Output:

```
In[603]: (not_equality, t_equality)
Out[603]: (True, True)
```

Evidently single qubit gates can be represented by two-by-two matrices. Other important single qubit gates are the Z-gate and the Hadamard Gate. Bridging the gap to the programmatic description of the first program discussed above one could use this description for a qubit:

```
config0 = qubit(0,-1+0j)
```

The mirror(M) and the beam splitter (B) may be represented as the following matrices:

$$B = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$M = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

they have to be multiplied by

$$\frac{1}{\sqrt{2}}$$

because of the requirement that the sums add up to one as mentioned above in the first law of QM.

4.2.2 Quantum computational complexity

The computational advantage of quantum computers is not apparent right away. The complexity of an algorithm is measured by how many operations (or how much time) it takes it to solve a problem of with a rising number of inputs. A computational problem

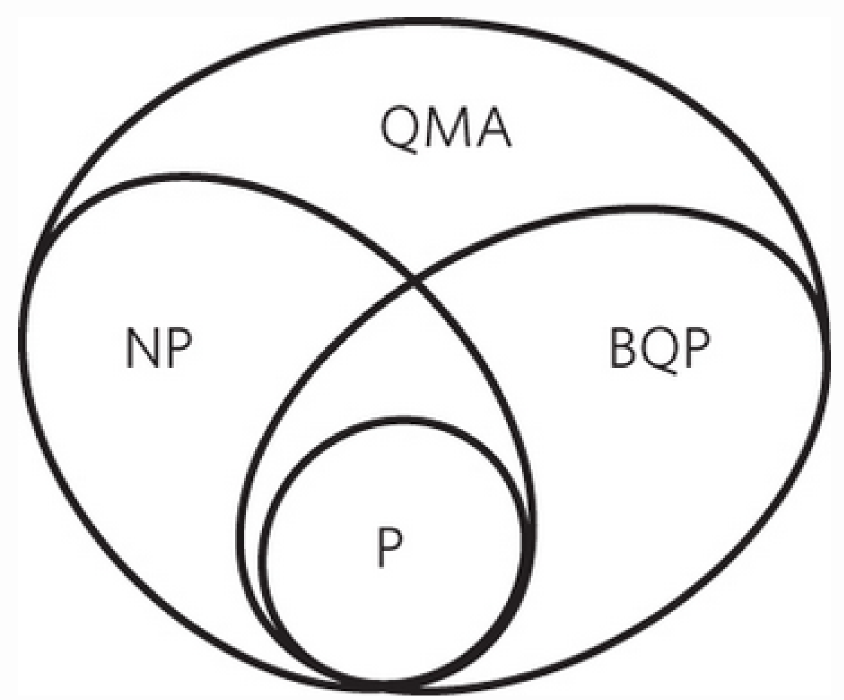

Figure 4.1: The relationship between classical and quantum complexity classes[SV09]

is regarded as efficiently solvable if it scales polynomially with its input size i.e $O(n^k)$ for input size n[NC10]. In accordance to classical computational complexity theory there is a set of problems which are easily solvable for a quantum computer. They are in BQP which stands for bounded error quantum polynomial time and their classical analogue is P [SV09]. The problems easy to check for a quantum computer are in QMA (Quantum Merlin-Arthur) analogous to NP. Of course there is also $QMA - hard$, which represents problems that are not efficiently solvable for a quantum computer. Some problems which are thought to be in NP for classical computers such as factoring have been shown to be in BQP [SV09]. Therefore there are problems most probably not efficiently solvable on a classical computer which are tractable by a quantum computer. Shor's factoring algorithm, whose discussion exceeds the scope of this text is one algorithm that has been shown to be able to use this speedup [NC10].

The following figure illustrates the relation between quantum computational complexity classes and classical complexity:

Chapter 5

Future Work and Conclusions

The above has been a very shallow overview of a very deep, fascinating and complicated topic. Unfortunately the extent to which this matters can be treated within this short thesis does not cover some important aspects of quantum mechanics: The Heisenberg Uncertainty Principle which would be important to understand entanglement. A deeper explanation of decoherence which was hinted at above, Schrödinger's Wave equation and its predictions as well as Bell's EPR Theorem, Hilbert Spaces, Hamiltonians, the behavior of multiple qubits, the universal quantum computer, quantum algorithms, quantum information theory, quantum error correction and quantum cryptography. In general there is too much to say to fit within this thesis. The author hopes that he could nevertheless give an approachable introduction into a field which is complex but very interesting. The effect of interference of subatomic particles was shown by the help of short programs. Some connections between computer science and quantum computation as well as the most basic of quantum gates could also be shown here. In the future it would be interesting to develop this knowledge further and use such gates in similar programs as have been illustrating the experiments, and maybe more complicated ones.

Bibliography

[AGR+82] Alain Aspect, Philippe Grangier, Gérard Roger, et al. Experimental realization of einstein-podolsky-rosen-bohm gedankenexperiment: a new violation of bell's inequalities. *Physical review letters*, 49(2):91–94, 1982.

[Ala08] Aspect Alain. Wave particle duality for a single photon. `http://replay-progressive.ethz.ch/h264-low.http/10.3930/ETHZ/AV-059d632c-3705-4454-aaf6-c3bd82fc0ddc/20131211_HPVG4_ZPC_Vuletic_grabber-dl.m4v`, 2008. Accessed: 2014-04-02.

[Bar99] Julian Barbour. *The end of time: The next revolution in physics*. Oxford University Press, 1999.

[BBBV97] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[DiV95] David P DiVincenzo. Quantum computation. *Science*, 270(5234):255–261, 1995.

[Fey82] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[Fey06] Richard Phillips Feynman. *QED: The strange theory of light and matter*. Princeton University Press, 2006.

[Hei79] Werner Heisenberg. *Quantentheorie und Philosophie: Vorlesungen und Aufsätze*, volume 9948. Reclam, 1979.

[Hun75] Geoffrey Hunter. Conditional probability amplitudes in wave mechanics. *International Journal of Quantum Chemistry*, 9(2):237–242, 1975.

[MMK+95] Chris Monroe, DM Meekhof, BE King, WM Itano, and DJ Wineland. Demonstration of a fundamental quantum logic gate. *Physical Review Letters*, 75(25):4714, 1995.

[Moo05] Gordon E Moore. Excerpts from a conversation with gordon moore: Moore's law. *URL ftp://download. intel. com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore. pdf*, 2005.

[NC10] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[Nie08] Michael Nielsen. Quantum computing for everyone. `http://michaelnielsen.org/blog/quantum-computing-for-everyone/`, 2008. Accessed: 2014-03-22.

[oPG00] Dpt. of Physics and Astronomy GSU. Toward the ”ultraviolet catastrophe”. `http://hyperphysics.phy-astr.gsu.edu/hbase/imgmod/bb7b.gif`, 2000. Accessed: 2014-04-02.

[Pai79] Abraham Pais. Einstein and the quantum theory. *Reviews of Modern Physics*, 51(4):863, 1979.

[Pai92] Abraham Pais. Niels bohr's times: In physics, philosophy and polity. *Physics Today*, 45:65, 1992.

[Rae04] Alastair IM Rae. *Quantum physics: illusion or reality?* Cambridge University Press, 2004.

[RP11] Eleanor G Rieffel and Wolfgang H Polak. *Quantum computing: A gentle introduction*. MIT Press, 2011.

[She01] C David Sherrill. A brief review of elementary quantum chemistry. *Georgia Institute of Technology, School of Chemistry and Biochemistry*, 2001.

[ST94] Jun John Sakurai and San Fu Tuan. *Modern quantum mechanics*, volume 104. Addison-Wesley Reading (Mass.), 1994.

[SV09] Norbert Schuch and Frank Verstraete. Computational complexity of interacting electrons and fundamental limitations of density functional theory. *Nature Physics*, 5(10):732–735, 2009.

[YM08] Noson S Yanofsky and Mirco A Mannucci. *Quantum computing for computer scientists*, volume 20. Cambridge University Press Cambridge, 2008.

Appendix A

Experiment Program

```
"""Interferometric experiments illustrated through programming"""
import cmath

class Mirror(object):
    """A reflective mirror"""
    def reflect(self, input):
        try:
            return input*(0+1j/cmath.sqrt(2))
        except TypeError:
            print("Complex Amplitude expected!")


class BeamSplitter(Mirror):
    """A Beam-Splitter is a half-silvered mirror"""
    def let_through(self, input):
        try:
            return input*(1/cmath.sqrt(2)+0j)
        except TypeError:
            print("Complex Amplitude expected!")


class Experiment(object):
    """Experiment-Stub"""
    def __init__(self):
        super(Experiment, self).__init__()
        self.config_list = []

    def print_configs(self):
        """show resulting configurations"""
        for i in range(len(self.config_list)):
            print("Configuration " + str(i) + \
```

```
                " has Amplitude " + str(self.config_list[i])) +\
                " and a squared modulus of: " +\
                str(abs(self.config_list[i]**2))

    def add_config(self, cfg):
        """add a configuration to the experiment"""
        self.config_list.append(cfg)


def main():
    """The function running the Experiments"""
    #Experiment 1
    exp1 = Experiment()
    bs = BeamSplitter()

    #arbitrary initial value
    config0 = (-1 + 0j)
    exp1.add_config(config0)

    #Photon goes straight: config1
    config1 = bs.reflect(config0)
    exp1.add_config(config1)
    #Photon takes a right angle: config2
    config2 = bs.let_through(config0)
    exp1.add_config(config2)
    print("Exp1:")
    exp1.print_configs()


    #Experiment 2
    exp2 = Experiment()
    m = Mirror()

    #First threee configurations stay the same as in Experiment 1
    exp2.add_config(config0)
    exp2.add_config(config1)
    exp2.add_config(config2)

    #The incoming configurations are relfected by the mirrors
    config3 = m.reflect(config1)
    exp2.add_config(config3)
    config4 = m.reflect(config2)
    exp2.add_config(config4)

    #The amplitude flows going corresponding
    #to configuration 5 and 6 are added up
    config5 = bs.reflect(config4) + bs.let_through(config3)
```

```
    exp2.add_config(config5)
    config6 = bs.reflect(config3) + bs.let_through(config4)
    exp2.add_config(config6)


    print("Exp2:")
    exp2.print_configs()


    #Experiment 3
    #exp3 = Experiment()


if __name__ == "__main__":
    main()
```