ACCESSING THE INFINITE;
NEW POWERS OF THE QUANTUM AUTOMATON

By

BENJAMIN TYLER SPROTT

UNIVERSITY OF WATERLOO

Accessing the Infinite; New Powers of the Quantum Automaton

by

Benjamin Tyler Sprott

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Physics

Waterloo, Ontario, Canada 2004

I hereby declare that I am the sole author of this thesis.
I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Ben Sprott

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Ben Sprott

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Using the inherent group structures on universal and non universal gate sets, the ability of quantum systems to distinguish languages is generalized. Comparisons to classical automata are made and group theoretic methods are used to devise classical automata that also distinguish the same languages. In every case an increasing disjunction between the size of the quantum automaton and the classical automaton is highlighted.

Acknowledgments

I want to thank my parents Jean and Doug, for their moral and spiritual support, as well as my uncle Eric for his financial support over the years.

Of course, I also want to thank Lucien Hardy for taking me on as a student, and who's ideas formed the seed for this work. I thank Doug Park who's untiring help with my pure math was indispensable. I would also like to thank Jeffrey O. Shallit and Ashwin Nayak, here at Waterloo, for their help with my automata theory, and Sarah Rees of Newcastle for her help with my group theory. I would also like to thank Raymond Laflamne and Achim Kempf for acting on my committee.

Finally, I could not have done any of this without my good friends Marcus, Robbi, Pam, Laura, and Jordan. Our friendship alone has made this endeavour worthwhile.

Contents

List of Figures

Chapter 1

Introduction

Quantum systems exist in a continuum of pure states. Pure states are odd in that, while having no entropy, they still contain some form of information; quantum information. In a paper by Lucien Hardy [2], a set of axioms are given which underlie quantum theory. The crucial axiom states that there is a continuous and reversible transformation between any pure states of a quantum system. As was previously known, this implies that there is an uncountable infinite continuum of pure states of any quantum system.

Since the set of all pure states is a continuum there are certainly infinite countable subsets. It is these subsets that are accessed when using a universal gate set for quantum computation. If one could access these states directly they would give even the tiniest quantum computer, the qubit, an infinite advantage over any classical machine. However, one cannot access these states directly, in the sense that one cannot extract information from these states directly. Upon measurement, a qubit gives one of only two possible outputs, not infinitely many. Furthermore, if one imagines the quantum machine broken into states that correspond to correct and incorrect answers, the incorrect answers may happen to lie arbitrarily close to the correct ones. Distinguishing correct answers from incorrect ones then becomes arbitrarily difficult. How, then, can we take advantage of what seems like an infinitely large set of internal states? A means to do this is given here.

While quantum mechanics is well understood in terms of quantizing classical Hamiltonian equations of motion, there is still interest in what makes quantum mechanics different. Its origins are certainly also of great interest. Attempts have been made to form a set of axioms for quantum mechanics which then could be cited as the real "reasons" for quantum behaviour. In [2] a

set of axioms is proposed that posit a special type of statistics that quantum mechanics would obey. In [19] we see an attempt at a purely information theoretic set of axioms. These types of considerations come to bear on quantum computation where new questions and answers can be found.

Typically, the question that is asked about quantum computation is, "How much more powerful is quantum computation than classical computation?" Concrete examples, in the form of faster algorithms, have been created that show that quantum computation carries some greater powers. These algorithms, then, pose questions as to why they are faster. One way to define a line between real and supposed speedups in quantum computation is to answer whether or not the quantum computations can be simulated on a classical computer. Interestingly, while it has been presumed true, entanglement alone is not the single defining feature that lends power to quantum computation. The evolution of a quantum system may be simulated by simply storing the unitary matrices used as the universal gate set. Storing these matrices in a classical computer and keeping track of their products amounts to the simulation of the system. Errors occur by the finite nature of the information storage. It was shown in [11] that this kind of simulation is linear in the number of qubits except in the case of high levels of entanglement. However, this kind of analysis is not sufficient to really probe the differences between the two schemes. For example we see [5] where quantum computations involving arbitrary levels of entanglement can be simulated efficiently. This is achieved using the group structure generated by the Clifford gates. In a personal communication, Gottesmann states that the Clifford group is a finite group and it is not too difficult to calculate its order. In [1] we find a possible example of how a qubit can deterministically solve problems in a much more space efficient way than a deterministic one way finite state automaton(1DFA). A general means of talking about those types of computational tasks is given here. It is shown in [7] that any one way quantum finite state automaton (1QFA) can be efficiently simulated by a 1DFA, but the reverse is not the case.

Recently, a class of computational tasks has been conceived each element of which, while easily performed by one quantum bit, can only be done by an increasingly large set of classical deterministic automata [1]. Further refinements were given in [3], [4]. It is shown here that the system envisioned in [1] is equivalent to what is called a measure once one way quantum finite state automaton (MO-1QFA). The classical automata used in the comparison in [1] were implicitly assumed to be one way deterministic finite automata (1DFA). Upon equating the models in

[1] with MO-1QFA and 1DFA, we find the resulting gap in computational power implied in [1] seemingly at odds with the current research in quantum automata theory. It is shown here that, indeed, the task *does* indicate a newly discovered gap between the power of quantum and classical automata, and is not at odds with the currently accepted viewpoint. Furthermore, the gap in computational power is shown to be due to a previously studied property of the MO-1QFA found in [8].

It has been shown [7] that any one way quantum finite automaton (1QFA) accepts, with bounded error, only a proper subset of the regular languages. However, [1] does not suggest that a 1QFA can recognize any non-regular languages with bounded error. Instead the work relies on the fact that 1QFA can accept non-regular languages with unbounded error. Here we will see the task presented in [1] rewritten as a task of distinguishing languages, both regular and non-regular. Two examples are given in which a qubit can distinguish several classes of language pairs. The direct translation of [1] into 1QFA theory shows, succinctly, the gap in computational power to be constant to linear and potentially constant to infinite. The added power forms a novel way of accessing the infinite set of pure states of a quantum system.

It seems that there is a serious contradiction between [1] and the current research in quantum computation [7], [8]. However this is not the case. In non-deterministic computation, which quantum computation can be, but not always is, it is important to recognize that the accuracy to which you may compute things is tantamount, especially in finite state automata theory. In automata theory, it is not important, how accurately you approximate any unitary. Instead, what is important is the probability that the machine will accept words within a given language and the probability of rejecting those not in the language. This accuracy is generally bounded away from $\frac{1}{2}$ by a small amount ϵ. We find in [8] a model that is in exact agreement with [1] but encounters the same problem of distinguishing elements in a language from those in the set of all possible words. There we see the proof that the measure once 1QFA is equivalent to a group finite automaton and the 1DFA. These proofs are certainly in contradiction with the notion that the set of pure states of a quantum system is infinite. It may seem impossible to utilize the infinite set of pure states in light of the current research, however we show that one may indeed access the infinite.

When we think of quantum computation we imagine quantum gates, written in an algorithm using some sort of universal gate set. There are plenty of choices for these gate sets, and they all

boil down to the same thing. For any gate set, one must chose some elements of SU(n) which, under matrix multiplication, generate some subgroup of SU(n). If the subgroup generated by these matrices is not infinite, then the gate set is not universal. Here we see that, while we are trying to access the continuum of states, we instead settle for a countable infinite subgroup. The group structures generated by our choice of gate set can then be used to make inferences about the types of things the computer can do. For example, if the group that is generated has only ten elements in it, then there are only ten pure states that can be accessed, and simulating this kind of computer would require a classical computer of about ten classical states. At the very least, it is certainly bounded. The work herein uses the group structures generated by various elements in SU(n), to say things about what the quantum automaton can do, and how it differs from a classical automaton.

It is shown here that the structure of quantum algorithms themselves can be used to understand the simulation of quantum machines. One may see each quantum algorithm as a word formed from letters in an alphabet, the letters of the alphabet are identified with the generators of a finitely presented group. Furthermore, it is also shown that equivalence classes in these groups can be defined in terms of formal grammars. The words may be written on a tape and used as an input for a classical machine. With the group structure on a particular universal gate set investigated, a question is then posed as to whether or not a 1DFA can simulate the action of a qubit. The computational task is shown here to be equivalent to a special problem in combinatorial group theory in which, again, two languages are distinguished. The languages are proven to be non-regular. This leads to results that are of interest to group theorists, namely that the qubit may solve a type of word problem for finitely presented groups. The result that a QFA can solve the word problem only with unbounded error is shown and is in agreement with current research [8].

Chapter 2

Distinguishing Classical Fields

Imagine one has a real field defined over some region of space. This means that at every point, one should be able to find, or measure a real number that is due to the field. If the field is classical then it is one in which information can be extracted without disturbing the field itself. To simplify things, let us imagine a real classical field defined along a one dimensional flat space $\mathbb{R}$ as in Figure 2.1. If it were possible to write down the number value of the field at some point, then we might store it in the memory of a computer, or perhaps just keep it in a book. For the moment, let us say that we dont know anything about this field except for one thing. Assume that it has been promised to us that the integral of the field is an integer multiplied by a known real number. This means that we can distinguish the field as being one of two types, namely, those for which their integral is an even number, and those for which their integral is an odd number. How might one tell which class the field belongs to? Here are two different methods one might employ to answer this question. Both methods are deterministic. One is a quantum protocol, while the other is a classical protocol.

2.1 How a Qubit Distinguishes Classical Fields: Integration

Given here is a quantum protocol that will distinguish the even integrals from the odd integrals in a purely deterministic fashion.

A quantum system with n distinguishable states evolves continuously through pure states. A pure state of the system is described by a vector of norm 1 in a complex, n-dimensional vector

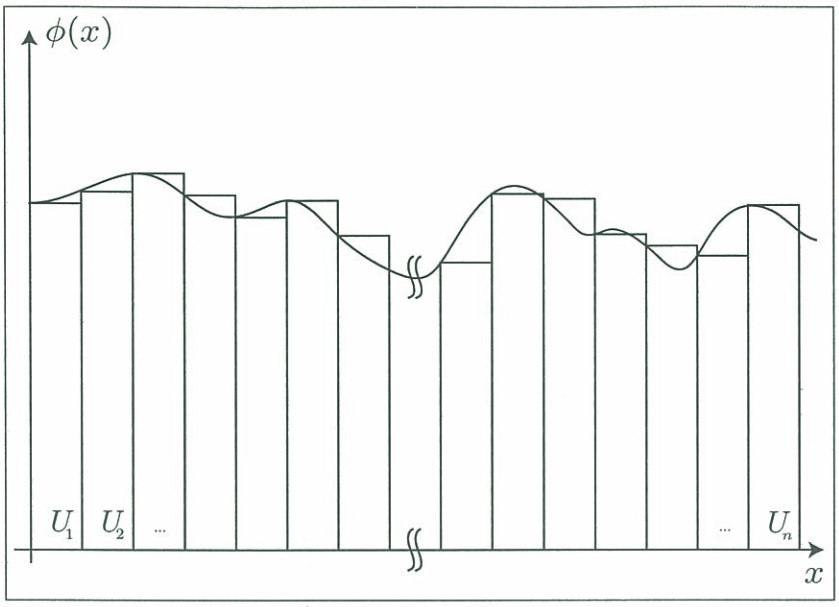

Figure 2.1: A classical field

space. Transformations of these vectors are unitary matrices. One such unitary transformation is one that maps a system in the state $|\psi\rangle = |00\ldots0\rangle$ to the state $|11\ldots11\rangle$, namely the NOT gate applied to each qubit separately. For the moment, let us focus on just a single qubit NOT gate, which is given by

$$U\left(\alpha|0\rangle + \beta|1\rangle\right) = \beta|0\rangle + \alpha|1\rangle$$

Which can alternately be written in two equations

$$\begin{aligned} U|0\rangle &= |1\rangle \\ U|1\rangle &= |0\rangle \end{aligned}$$

One way to imagine the evolution of a qubit is to see it travelling along one dimension, x, over which some classical field $\phi(x)$ is varying see Fig. 2.1. The field is classical, and thus coupling it to the qubit does not affect the field. If the qubit is coupled to the field, it will change state as it traverses the field. The internal state of the system will rotate around in the 2-dimensional Hilbert space. The effect is thus a unitary transformation on $\mathcal{H}_2$. Since the qubit is moving, its

position is a function of time. We can simplify this idea by fixing the position of the particle and varying the field continuously over time. The initial state of the qubit is

$$|\psi(t=0)\rangle = |0\rangle$$

We can choose our field to be a magnetic field whose direction is fixed in the x direction, but whose amplitude is varying with time. The Hamiltonian is

$$\hat{H} = \omega(t) S_x$$
$$\omega(t) = \frac{|e|B(t)}{m_e c}$$
$$S_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The time evolution operator is

$$\begin{aligned} U(t) &= \exp\left(-\frac{i|e|}{\hbar m_e c}\int_0^t S_x B(t')dt'\right) \\ &= \exp\left(-\frac{i|e|S_x}{\hbar m_e c}\int_0^t B(t')dt'\right) \end{aligned}$$

Let $I(t)$ be the integral in the exponent

$$\begin{aligned} I(t) &= \int_0^t B(t')dt' \\ &= \int_0^t \phi(t')dt' \end{aligned}$$

Let the field be such that the integral of the field between points 0 and t is an integer, m, times a real number, β.

$$\int_0^t \phi(t')dt' = m\beta$$

This is the single defining characteristic of the field. Otherwise there is no restriction on the field. It need not be continuous. The time varying field can be expressed in terms of a time varying

magnetic field strength

$$B(t) = \beta\phi(t')$$
$$I(t) = \int_0^t \beta\phi(t')dt'$$
$$= \beta \int_0^t \phi(t')dt'$$

If we are able to set β

$$\beta = \frac{\hbar m_e c m\pi}{2|e|}$$

The time evolution operator equation becomes

$$U(t) = \exp\left(\frac{-im\pi}{2} S_x\right)$$

This operator has a matrix equivalent that we can find using a spectral decomposition. The operator is of the form

$$\begin{aligned} U(t) &= f(S_x) \\ &= \sum_i f(\lambda_i)|e_i\rangle\langle e_i| \end{aligned}$$

Where the λ_i and $|e_i\rangle$ are the eigenvalues and eigenvectors of the operator S_x.

$$U(t) = \frac{1}{2}\begin{pmatrix} e^{-i\frac{m\pi}{2}} + e^{i\frac{m\pi}{2}} & e^{-i\frac{m\pi}{2}} - e^{i\frac{m\pi}{2}} \\ e^{-i\frac{m\pi}{2}} - e^{i\frac{m\pi}{2}} & e^{-i\frac{m\pi}{2}} + e^{i\frac{m\pi}{2}} \end{pmatrix}$$

$$U(t) = \begin{pmatrix} \cos\frac{m\pi}{2} & i\sin\frac{m\pi}{2} \\ i\sin\frac{m\pi}{2} & \cos\frac{m\pi}{2} \end{pmatrix}$$

Though we are promised that the integral will be integer valued, we don't know if m is even or odd. Using this method, we can determine whether m is even or odd, since the operator acting on the qubit is

$$U(t) = \begin{cases} \textit{Identity} & m \text{ even} \\ iS_x & m \text{ odd} \end{cases}$$

The resulting operator maps the qubit from $|0\rangle$ to $|0\rangle$ if m is even, and from $|0\rangle$ to $i|1\rangle$ if m is odd. A measurement of the qubit then determines the parity of m deterministically.

2.2 Questions of Approximation

One may wonder how the system's performance is affected, if, instead of coupling the quantum system with the field, we operated on it based on information gathered from the field. It is fair to assume that we can break up the integral into N sub-integrals. These sub-integrals are given by numbers which can be read, and the length of these numbers gives the accuracy to which we know the field. If we took these numbers and attempted to use them to transform the quantum system, we have two problems. First, the numbers themselves, being of finite length, contain inherent error. Secondly, transforming those numbers into a unitary on the qubit should also contain some error of approximation.

In answer to these questions we can note that any unitary can be approximated to any accuracy, see [10], [6]. This means that, whatever complications that may arise from the approximations can be dealt with by going to a better approximation. We simply need to find the unitary matrices to high accuracy and then approximate them to high accuracy. Furthermore, there is no limit to the accuracy which we can achieve thus making accuracy of approximation a non-issue.

2.3 How a Classical Machine Distinguishes Classical Fields

Is it possible that a classical machine could distinguish the kinds of fields discussed above? Of course, one may imagine a classical physical system with continuous degrees of freedom, such as a rod that is free to pivot about one end. If the rod were in some way coupled to a field then the task might be performed classically. What would this tell us about the difference between classical and quantum information? It might not tell us much for in each case, there is no information that is handled in order to perform either task. One may gather information about the field, and then use it to *instruct* a system as to how to distinguish the two types of fields. Along these lines [1] shows that a classical machine, given such a task, must have a smallest size. The classical task is similar to that of distinguishing fields, except the fields are now given as indexed sequences of numbers. The sums of the sequences are promised to be either odd multiples of K or even multiples of K.

The integral described in the previous section can be broken down into a sum of sub-integrals as in Figure 2.1. Each of these sub-integrals can be seen as individual unitary operators U_i.

These operators, applied sequentially evolve the quantum system in discrete steps. Each of these sub-integrals can be any value in $\mathbb{R}$ which creates difficulties for integrating the field classically. Nevertheless, with this simplification the integration now becomes a sum

$$I = \sum_{n=1}^{N} \phi_n = m\alpha \tag{2.1}$$

While the actual sub-integrals must take on arbitrary value in the real numbers, the problem is further simplified by assuming that the values of each sub-integral is restricted to values of the form

$$\phi_n = \frac{\alpha k_n}{K}$$

where $k_n \in \{0, 1 \ldots, 2K-1\}$. By substituting this into (2.1) we find

$$\begin{aligned} \sum_{n=1}^{N} \phi_n &= \sum_{n=1}^{N} \frac{\alpha k_n}{K} \\ \sum_{n=1}^{N} \frac{\alpha k_n}{K} &= m\alpha \\ \sum_{n=1}^{N} k_n &= mK \end{aligned}$$

We will see that, even with these vast simplifications, there still remains a gap between the quantum and classical machines performing this task. Recall that we just want to know if m is even or odd. Given the following properties of modular arithmetic

$$\begin{aligned} \mathrm{mod_{ab}(ax)} &= \begin{cases} 0 & \text{if } ax < ab \\ ax - nab & \text{for } nab < ax \text{ or } nb < x \end{cases} \\ &= a(x - nb) \\ &= a\mathrm{mod_b(x)} \end{aligned}$$

in order to know if m is even or odd we just need to keep track of the sum mod $2K$.

$$\begin{aligned} \mathrm{mod_{2K}(mK)} &= K\mathrm{mod_2(m)} \\ &= \begin{cases} K & \text{if } m \text{ is odd} \\ 0 & \text{if } m \text{ is even} \end{cases} \end{aligned}$$

Thus we have a well defined mathematical task, namely, given a sequence of numbers k_n chosen from the finite set $Q = \{0, 1 \ldots, 2K-1\}$ such that

$$\sum_{n=1}^{N} k_n = mK$$

we must find the sum of those numbers mod $2K$ and hence decide if m is even or odd.

Ultimately, we are trying to compare classical information to quantum information via a comparison between a classical machine and a quantum machine. To this end let us define the classical machine. The classical machine is some object that has a finite set of distinguishable states. Abstractly, this can be modelled as any finite set, call it S whose members are labelled $l_n \in S = \{0, 1, \ldots, L-1\}$. In order to compare this machine to the qubit, we simply ask, "How large must L be such that it can perform the task of addition mod $2K$?" Intuitively we can imagine the machine starting in the first state, which we label 0 since the sum at the beginning is 0, and then proceeds to go to the k_1^{th} state after being given the first number, k_1. After reading the second number k_2, the machine will transition to the $\mathrm{mod}_{2K}(k_1 + k_2)$ state.

A nice way to visualize this is through graphs as we can see from Figure 2.2. In [1] these maps are defined abstractly as functions from the space $S \times Q$ to S as

$$f^n_{l_{n-1}}(k_n) = l_n \tag{2.2}$$

The space $S \times Q$ is the set of all states of the system, and implies that at any stage, the only available information is the internal state of the machine and the current number in the sequence k_n. This is phrased in [1] as a communication task where the previous n numbers are somehow encoded in Q, the machine. The k_n are seen as local classical information which the parties attempt to encode in the system before passing it on.

We can see that for a classical automaton such as that described here, the transitions that the machine makes can be seen as maps from a finite set to a finite set. In the case of modular addition, the map that is induced by the symbol 1, $f^1_{i-1}(1) = i$, is given in Figure 2.3. The maps in eq. (2.2) may, as in Figures 2.2 and 2.3, be defined in exact agreement with modular addition, so that the resulting state label is the modular sum of the label of the previous state and the given number from Q. It is easy to see that this protocol, or set of maps, will fail if the number of internal states, L is less than $2K$. A machine design specifically for addition mod $2K$ has maps

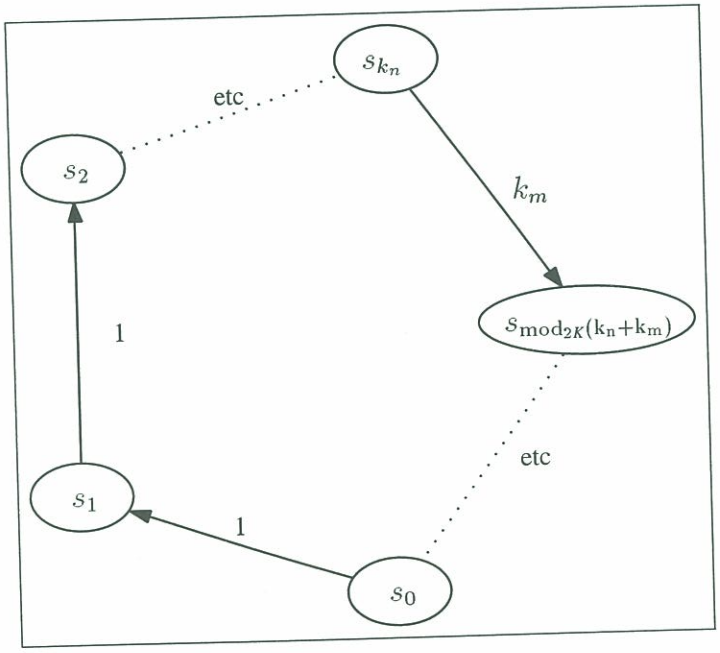

Figure 2.2: A Machine that adds mod2K: the simplest protocol

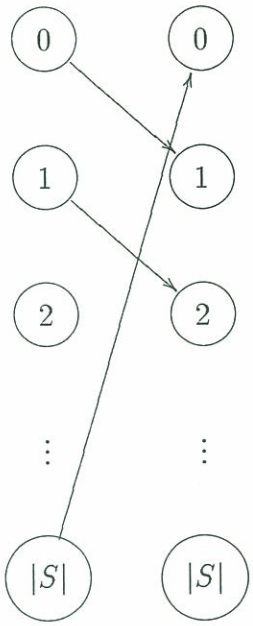

Figure 2.3: The map induced on the modular adder upon input of 1

of the form

$$f^{n}_{l_{n-1}}(k_n) = \mathrm{mod}_{2\mathrm{K}}(\mathrm{l_{n-1}} + \mathrm{k_n}) = \mathrm{l_n}$$

However, there may be other definitions of maps that may distinguish the sums. The definition of those maps is called the protocol, and [1] establishes that regardless of protocol, L must be larger than $2K$. Thus, we can imagine that there are other protocols that do the same job of deciding if m is even or odd. Amazingly, in [1], we find a proof that no protocol will be successful, given that $L < 2K$. The proof relies on the fact that, if the set of internal states is smaller than the set of numbers Q, there will be at least one transition from internal state to internal state that is induced by two numbers, Figure 2.4. This means that, if the machine is found in some state l_n, then there are two distinct sums that are consistent with the internal state l_n. One sum includes a, while the other includes b. This will have the effect that, upon receiving the final number k_N and finding the machine in state l_{N-1}, one has several choices when deciding which sum the machine was encoding.

Let A_n be the set of distinct sums mod $2K$ that are consistent with receiving the machine in state l_n. The first thing we should notice about this set is that it has at most $2K$ elements as its

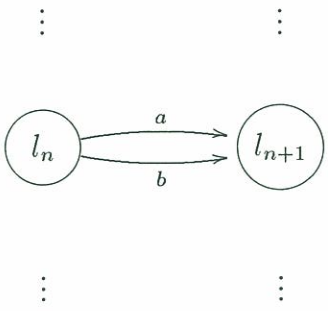

Figure 2.4: The $+1$ map.

elements are the integer valued sums that the machine may be encoding. Furthermore, A_n is a subset of the set of all sums mod $2K$, which clearly has exactly $2K$ elements. The set of all sums mod $2K$ has a modular type of algebra known as $\mathbb{Z}/2K$ and A_n is embedded in it. If we were to add a number to any of the elements of A_n, it would, sort of, push it forward inside the larger set of numbers between 1 and $2K$. The element would be pushed forward until it reached $2K$, and would then find its way back to 1 and so on. We need to show that, regardless of protocol, since there will always be one transition which is induced by two distinct integers, the resulting set of possible sums mod $2K$ must grow with K. Furthermore, we need to show that no matter what subset $A_n \subseteq \mathbb{Z}/2K$ our protocol chooses, this subset must always have elements that differ by K, thus causing an error. In mathematical terms we need to prove the following conjecture

$$\begin{aligned} |A_{n+1}| &\geq |A_n \oplus \{a, b\}| \\ |A_n \oplus \{a, b\}| &\geq |A_n| + 1 \end{aligned} \tag{2.3}$$

and thus $|A_N| \geq N$. The proof of (2.3) is by contradiction. We begin by assuming that a machine with less than $2K$ internal states can successfully perform the task of adding mod $2K$, and thus render (2.3) false. Let $L = 2K - 1$ and we assume that this machine is such that when it is in state l_n for which there are two different integers, a and b, inducing the transition to l_{n+1} it is true that

$$|A_n \oplus \{a, b\}| = |A_n|$$

If we note that

$$|A_n| = |A_n \oplus \{a\}| = |A_n \oplus \{b\}|$$

then we can perform the following algebra

$$\begin{aligned}
A_n \oplus \{a,b\} &= A_n \oplus \{a\} \cup A_n \oplus \{b\} \\
|A_n \oplus \{a,b\}| &= |A_n \oplus \{a\} \cup A_n \oplus \{b\}| \\
|A_n \oplus \{a,b\}| &= |A_n \oplus \{a\}| + |A_n \oplus \{b\}| - |A_n \oplus \{a\} \cap A_n \oplus \{b\}| \\
|A_n \oplus \{a\}| &= |A_n \oplus \{a\}| + |A_n \oplus \{a\}| - |A_n \oplus \{a\} \cap A_n \oplus \{b\}| \\
|A_n \oplus \{a\}| &= |A_n \oplus \{a\} \cap A_n \oplus \{b\}| \\
|A_n \oplus \{b\}| &= |A_n \oplus \{a\} \cap A_n \oplus \{b\}|
\end{aligned}$$

From (2.4) and (2.4) we arrive at the following important consequence

$$\begin{aligned}
A_n \oplus \{a\} &\subseteq A_n \oplus \{b\} \\
A_n \oplus \{b\} &\subseteq A_n \oplus \{a\}
\end{aligned}$$

meaning that

$$A_n \oplus \{a\} = A_n \oplus \{b\}$$

Thus, rotating all the elements forward by a gives the same set as rotating them forward by b. These two sets that result from rotating forward by b and a can be turned back, by subtracting a from each element implying that

$$\begin{aligned}
A_n &= A_n \oplus \{|\, b-a \,|\} \\
A_n &= A_n \oplus \{|\, \Delta_1 \,|\}
\end{aligned}$$

Δ_1 is a period of the set A_n as such it must divide $2K$. One may show that if Δ_1 is not a period of A_n, then A_n must still have a period. We can apply Δ_1 i_1 times, for some i_1, and find that the resulting transformation on A_n is $0 < i_1\Delta_1 \mathrm{mod} 2\mathrm{K} < \Delta_1$. Relabeling $i_1\Delta_1 \mathrm{mod} 2\mathrm{K} = \Delta_2$ then either Δ_2 is a period or we can apply it a series of i_2 times to find the resulting transformation $0 < i_2\Delta_2 \mathrm{mod} 2\mathrm{K} < \Delta_3$ and so on. Eventually, there will come a smallest possible number $\Delta_j = v$ equal to or greater than 1. v is a period of A_n and thus must divide $2K$. Any divisor, v, of $2K$ partitions the set of sums mod $2K$ into $2K/v$ elements and this forms A_n. This set of elements must contain those that differ by $2K$ by simple arithmetic. This means that there exists two distinct sums, consistent with l_n, one indicating an even multiple of K and one indicating an odd multiple of K, and hence causing an error. The original assumption, that a successful

protocol exists for a machine with less than $2K$ states, causes an error, and hence the assumption must be false. With this proven, we have it that the classical machine must have at least $2K$ internal states, and thus its size scales with K.

We have changed the task so considerably that it may well be that a qubit cannot perform this task. It will be shown further on how a qubit performs this task. It should be kept in mind, that this is *not* a task of integration, or even of counting. Instead this is a task of distinguishing sequences whose sums are even multiples of K from those that are odd multiples of K. In Section 6.3 it is shown how a qubit performs this task. This is accomplished by translating this work into automata theory. The alphabet used there is reduced to a single symbol, whereas in [1] an arbitrarily large alphabet is used. This ever increasing alphabet, when being translated, causes conceptual difficulties, and is best phrased in a unary language.

Chapter 3

Quantum Gates

It has been shown that any unitary transformation on n qubits can be approximated to arbitrary accuracy by the H_i, T_i, and the $CNOT_{ij}$ gates. The subscripts refer to the qubits on which the transformation is applied. In the case of the $CNOT_{ij}$ gate, the j^{th} qubit is flipped conditional on the i^{th} qubit. The matrices for these transformations are

$$\begin{aligned} H &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ T &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \end{aligned} \tag{3.1}$$

The matrix for the two qubit $CNOT$ is

$$CNOT_{12} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The matrices on n qubits are of rank 2^n since, in order to build a single qubit gate U for the i^{th} qubit, one multiplies identities and U using tensor products. For example

$$U_i = I \otimes I \otimes \ldots \otimes U \otimes \ldots \otimes I \otimes I$$

Through multiplying these gates together one may approximate any unitary transformation to any accuracy. We give each gate a symbol and thus the set of all gates forms a sort of alphabet.

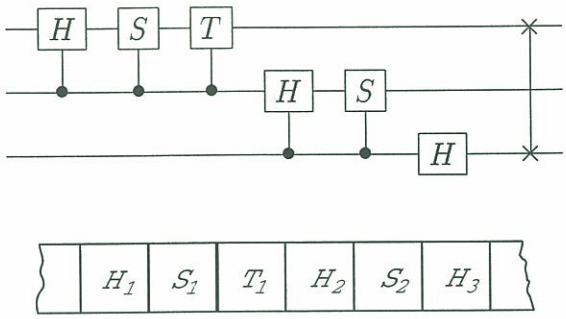

Figure 3.1: Quantum algorithm acting on 3 qubits and the tape describing it

Let us call this alphabet Σ. Algorithms can thus be seen as words formed in this alphabet. Of course the term algorithm is being used improperly. Really, these are just the set of all unitary transformations on the qubit. It seems clear that many different sets of matrices may form universal gate sets. Interestingly it seems that almost any gate set is universal [21], [20]. Each gate set should have differing algebraic properties.

We can see that H does not commute with T for example. How does one go about finding a gate set that has a desired algebraic property? What are the freedoms available for finding dense subsets, and how can those freedoms be manipulated to solve various problems? These questions remain open.

Quantum gates can be labelled and the resulting algorithm can be written on a tape. Figure 3 shows an algorithm and the tape below it. Given a quantum algorithm that maps the evolution with gates, a classical tape can be written. This tape can then form the input to a classical computer which would then be designed to predict the resulting quantum state. It will be shown later on that every quantum algorithm can be elevated to the status of a formal language. These languages are non-regular and, oddly, not even quantum computers can distinguish them from the set of all possible words.

Chapter 4

Automata Theory

4.1 Languages and Alphabets

The task in [1] has a well phrased analog in finite state automata theory. In automata theory the main objects of interest are the mathematical models for the machine, M, which represent the computer, and the languages, L, they recognize. A language is any prescribed set of words in an alphabet Σ of symbols. An alphabet is simply any set of symbols, e.g. $\Sigma = \{a, b, c, \ldots\}$ or $\Sigma = \{0, 1, x, \ldots\}$. The words of the language obviously have a beginning and an end, typically the right and left ends respectively. The alphabet is often augmented to include symbols $\$, ¢$ that indicate the beginning and end of a word, respectively. The expanded alphabet is given by $\Gamma = \Sigma \cup \{\$, ¢\}$. The empty character is given as ϵ. Languages, both finite and infinite, can be given by an expression. An example of such an expression on the alphabet $\Sigma = \{0, 1\}$ is

$$L = \{x \in \Sigma^* | x = 0^*1^*\}$$

or, the set of all words consisting of a string of zeros followed by a string of ones. The simplest language is the set of all possible words in the given alphabet. This set is denoted by Σ^* and contains the empty string. Σ^* can be seen as a free product on the alphabet or the set of all possible combinations of the letters of the alphabet. The complexity goes up from there towards languages that cannot be recognized by any finite state machine. The machine accesses words by reading a tape which consists of a sequence of letters. The machine reads the letters one at a time. After reading each symbol, the machine changes its internal state and moves one space along the

tape. In general the tape may move left, right, or remain stationary after each computation.

4.1.1 Language Recognition: Deterministic, Bounded and Unbounded Error

Finite automata are designed to perform one task and this is to recognize some language. Language recognition is a decision problem, where, given a set of words, known as the language, a machine must return the same response for all words in the set. Naturally, that response is just some state of the machine, and that state is called the final or *accepting* state. If after every symbol in a word is read, the machine is found in the final state q_f then it accepts the word as an element of a language. If it is found in any other state, it is considered to have rejected the word. Each machine is designed to accept only a single language. Thus, if our language is the set of all even length strings of 0s, then we have $\Sigma = \{0\}$ and

$$L = \{0^{2n} | n \in \mathbb{Z}\}$$

The machine accepting this language can have as little as two internal states and is given in Figure 4.1 After the machine reads any word in the language it always returns to Q_0. Reading odd length words, those in $\Sigma^* - L$ the machine always ends up in Q_1. Upon reading the end of tape symbol ¢, the machine moves to the Q_{acc} state. The double circle around Q_{acc} indicates that it is the accepting or final state.

A machine may accept a language in a non-deterministic fashion. It may be that for some of the trials the automaton accepts a word and for others it may reject it. If an automaton accepts every word in a language half of the time, and also accepts every word in $\Sigma^* - L$, half of the time, then there is no way to distinguish words in the language from those outside the language. A margin ensures that the computer accepts words in a language with probability greater than half, $p > \frac{1}{2} + \epsilon$. If an automaton accepts words in a language L with probability $p > \frac{1}{2} + \epsilon$ and rejects words in $\Sigma^* - L$ with the same probability, then the automaton is said to accept L with bounded error. The error is bounded away from $\frac{1}{2}$ by ϵ. Since there is a margin, then statistics may be gathered to decide which words are in a language. This fact is why a 1QFA is actually less powerful than a 1DFA. If $\epsilon = 0$ then there is no bound on the probability that the machine may reject words in $\Sigma^* - L$ and we say that the automaton accepts L with unbounded error.

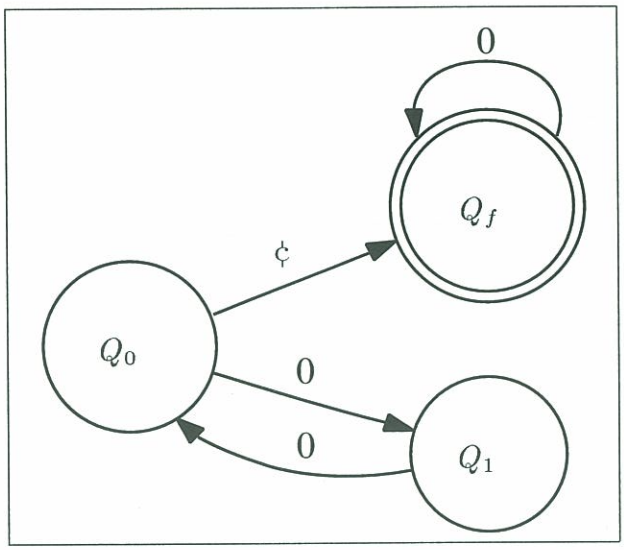

Figure 4.1: A machine accepting even length words

4.2 Classical Computation Models

4.2.1 The 1DFA

The model of classical computation considered here is the one way deterministic finite state automaton (1DFA). After each computation, these machines always move one character to the left, rather than having the freedom to read a tape forwards or backwards. This machine is similar to a Turing machine except it cannot rewrite the symbols on the tape and the number of internal states is finite. Also, a Turing machine has the freedom to read the tape in both directions. The set of internal states is some finite set, call it $Q = \{q_0, q_1 \ldots, q_f\}$. The state of the machine at any time can be given by the internal state and the symbol on the tape, (q, σ). The space of states is then $Q \times \Sigma$, and the transitions of the machine are given by a set of functions from this space onto Q

$$\delta : Q \times \Sigma \to Q$$

Each map can be written as a function

$$\delta(q, \sigma) = q'$$

meaning that if the machine is in state q and the symbol σ is read, then the machine changes to state q'. A DFA can be compactly written as

$$M(Q, \Sigma, \delta, q_0, q_f)$$

M is a machine with a set of internal states Q, a set of maps between those states, δ, an initial state q_0 and a final state q_f.

Another way to define this is through a set of maps from the larger set $Q \times \Sigma \times Q$ to the set $\{0, 1\}$. For example, the map

$$\delta(q, \sigma, q') = 1$$

Indicates that, upon reading the symbol σ, a machine in state q changes to state q'. As we will see, this type of definition allows us to assign probabilities and then quantum probability amplitudes to various transitions of our choice simply by changing $\{0, 1\}$ to $\mathbb{R}$ or $\mathbb{C}$.

Any Language accepted by a 1DFA is a regular language. This will be described in greater detail below.

4.2.2 Non-Deterministic Finite Automata

For a non-deterministic automaton, there may be transitions between elements that occur even when no input is offered to the machine.

$$\delta(q, \epsilon, q') = 1$$

In a probabilistic FA, some maps may have an associated probability not equal to one

$$\delta(q, \sigma, q') = a \text{ for } 0 \leq a \leq 1 \text{ and } a \in \mathbb{R}$$

These maps also define the elements of a stochastic matrix.

4.2.3 Reversible FA

None of the models described above forms a strictly reversible machine. In fact, it is the freedom to use maps that are *not* reversible that ensures that these machines are more powerful than their reversible counterparts. One may implement irreversible maps in a 1QFA, however, they

are strictly probabilistic. In the case of a regular irreversible 1DFA, one may deterministically map several states to a single state. This is not possible with a quantum machine allowing only unitary (reversible) transformations. Using a larger class of quantum transformations, namely the superoperators, or just even projective measurements and unitaries, one may map many different states to the same state. The work herein explores the structure of the reversible maps on a quantum system.

A special class of classical finite state automata are the group finite automata or GFA. These are defined as those 1DFA such that for every internal state q, upon reading a letter σ the machine is mapped to exactly one other, distinct, internal state q'.

$$\forall q' \; \exists \text{ exactly 1 } q \text{ s.t. } \delta(q, \sigma) = q'$$

If we look at the set of all 1-1 maps from a finite set to itself, this set of maps has the group structure of the permutation group, hence the name. The GFAs are a member of a larger family known as reversible 1DFA.

4.3 Regular Languages

We wish to have a notion of what is computationally hard within the context of automata. Naturally, the line between what is hard and what is not boils down to language recognition. If one class of machines, say classical, can recognize a language L, while another class, say quantum machines, cannot recognize L then we can say that the classical machines are more powerful. How can we define the set of languages recognized by classical automata? Generally, these are known as the regular languages, however looking in [13] we find a rather involved definition of regular languages. It is best to begin with a few preliminaries about languages in general.

There are three useful operations available on languages. These are union ($+$), multiplication or concatenation, and free product (*). By free product (*), one means the set of all possible products, or concatenations of the elements of some set including the empty word ϵ. Thus the set of all words Σ^* in the alphabet Σ is the set of all concatenations or combinations or multiplications of the letters in the alphabet. One might have the set of all concatenations of the word jump and that is given as $\text{jump}^* = \{\epsilon, \text{jump}, \text{jumpjump}, \ldots, \text{jump}^n, \ldots\}$. Using these operations, we find in [12] a set of rules for regular language construction

1. $\emptyset$ is a regular expression and denotes the empty set.
2. ϵ is a regular expression and denotes the set $\{\epsilon\}$.
3. For each a in Σ, $a^\dagger$ is a regular expression and denotes the set $\{a\}$.
4. If r and s are regular expressions denoting the languages R and S respectively, then $(r+s)$, (rs) and (r^*) are regular expressions that denote the sets $R \cup S$, RS and R^* respectively.

A language is called rational if it is formed by any finite number of applications of the operations defined above.

Technically, according to [13] a regular language is one generated by a regular grammar. We need not discuss grammars to understand what is important about regular languages. By definition, any language recognized (*recognized* being relative to the machine type) by a finite automaton is simply called recognizable. However, Theorem 2.5.2 in [13] states that any recognizable language is rational. Another theorem states that the set of rational languages is equivalent to the set of regular languages. It is also a theorem that any rational language is recognized by some finite automaton. Thus we may say that any language recognized by some finite automaton is regular.

There are two ways to define an equivalence class on a language. One is via inclusion in the language, while the other is how the words affect a machine. Relative to a language L, there is an equivalence relation $=_L$, such that if two words, u, u' are in L then $u =_L u'$ if $\forall y$ in Σ^* we find $uy \in$ L iff $u'y \in L$. In [22] we find the following theorem

Theorem 4.3.1. [22]

A set $T \subseteq \Sigma^*$ is a regular language if and only if the number of equivalence classes of Σ^* by the equivalence relation $=_L$ is finite. If the number of equivalence classes is $e < \infty$ then for a suitable $\mathfrak{U}, T = T(\mathfrak{U})$ where the automaton $\mathfrak{U}$ has e states. No automaton with fewer than e states defines T.

Theorem 4.3.1 essentially states that for a language T to be regular the equivalence relation $=_L$ must partition Σ^* into only finitely many equivalence classes. This can be understood by considering that, upon reading any word in Σ^*, the machine must be found in one of the *finite* number of states Q. Many words may induce a transition of the machine to some internal state

q_i, and all the words that do this are within the same equivalence class $[q_i]$. Since the machine is finite, the number of equivalence classes must be finite.

The other equivalence relation $=_M$ is a bit simpler to visualize as it is specific to the machine accepting words in a language L. Thus, if $w, w' \in L$ then $w =_M w'$ iff $\delta(s, w) =_M \delta(s, w')\ \forall s$. Recall that δ are the maps defining the automata. Being prepared in some fixed starting state, s, and reading two different words w, w', if the machine reaches the same state for both words then the words are equivalent.

It is a theorem that any of the aforementioned automata accepts a language L, then that language is considered regular. We find in [22] Theorem 3 which states that for every probabilistic machine that accepts a language L with bounded error, there exists a deterministic machine also accepting L. In [13] Theorem 2.3.6 states that for any nondeterministic automaton recognizing a language L, there exists a deterministic automaton that also recognizes L.

The set of languages accepted, with bounded error, by any of the above mentioned machines is the regular languages. Though the sizes may differ from machine to machine, any language accepted by one machine is accepted by any other machine [12]. It will be shown that, in fact, the set of languages accepted by the 1QFA is actually smaller than that accepted by the classical machines. In these cases, the quantum machines are said to be less powerful.

4.4 The Quantum Finite Automaton

The quantum finite state automaton (QFA) is defined similar to the probabilistic automaton. The maps, instead of being assigned a classical probability, are assigned a complex probability amplitude. Thus

$$\delta : Q \times \Sigma \times Q \to \mathbb{C}$$

and

$$\delta(q, \epsilon, q') = z \ \in \mathbb{C} \tag{4.1}$$

Issues of unitarity and reversibility are ensured by what is called well formedness of these maps [7]. The set of internal states $Q = \{q_0, q_1, \ldots, q_{accept}, q_{reject}\}$ is a set of orthogonal or distinguishable states and the set of all pure states is then a complex vector space of dimension Card(Q)=$|Q|$. Again, the states are vectors in this space with unit norm. There are several different types of

QFA, the one way (1QFA), the two way (2QFA), the measure once (MO) and measure many (MM). In the case of the one way, the tape is read sequentially, cell by cell, in one direction and the tape position is incremented by one after every internal transition. This is a restricted version of the two way automaton (2QFA). In the case of the 2QFA there is a freedom to either move left, right, or idle, and so the tape position is ultimately in a superposition. It is the quantum nature of the information input that gives the 2QFA more power than the 1QFA and all deterministic automata. In fact, in [7], it is shown that a 2QFA can accept non-regular languages with any error bound. We concern ourselves here with the one way QFA in order to simplify the comparisons between [1] and [7], [8].

4.4.1 One way Measure Many QFA or MM-1QFA

Aside from the one way or two way freedom of reading the tape, there is also the freedom of when to measure the automata. The measure many 1QFA (MM-1QFA) is one in which a measurement of the state of the system is made after every symbol is read. The internal state of this model is seen as the direct sum of three linear subspaces $E_{non} \oplus E_{rej} \oplus E_{acc}$ the non-halting, reject and accept subspaces respectively. These subspaces are simply those spanned by some choice of elements in the basis of a Hilbert space. For example, two qubits can be used and we would find $E_{non} = \alpha|00\rangle + \beta|10\rangle, E_{rej} = \delta|01\rangle, E_{acc} = \gamma|11\rangle$. Projection operators onto E_{non}, E_{rej} and E_{acc} are defined as P_{non}, P_a and P_r respectively.

After each symbol is read, and the corresponding transformation occurs, three projective measurements are made, one onto each of the three subspaces. This model is that of a physical machine, but being an entirely mathematical entity, one uses it to perform calculations in computational complexity. One does this by keeping track of the projection of the state vector on each of these subspaces. In practice, many preparations must be made and an ensemble measurement must be performed. The statistics gathered from those measurements would, theoretically, be equal to the amplitude of each of the projections.

To keep track of the projections, assign to each state of the system a three-tuple

$$v = (\psi, p_{rej}, p_{acc}) \tag{4.2}$$

where ψ is the normalized state vector of the system after projection onto the non-halting subspace, while p_{acc}, p_{rej} are the probabilities that the machine accepts or rejects a string of symbols.

These probabilities are given by

$$p_{acc} = \| P_a|\psi\rangle \| \tag{4.3}$$
$$p_{rej} = \| P_r|\psi\rangle \| \tag{4.4}$$

A computation begins with the machine in the state $v_{in} = (\psi, 0, 0)$. The state v is the resulting state after having read some string x. The transition induced by reading the word x is given by an operator T_x acting on the starting state

$$T_x v_{in} = v$$

The word x is formed from letters in the extended alphabet $\Gamma = \Sigma^* \cup \{\$, ¢\}$ as $x = ¢\sigma_N, \ldots, \sigma_2, \sigma_1\$$. After reading the first letter the machine undergoes a transformation

$$V_{\sigma_1}|\psi\rangle = |\phi\rangle$$

V_{σ_1} is a unitary operator defined by equations such as in eq. (4.1)

$$\delta(q, \sigma, q') = \langle q'|V_\sigma|q\rangle$$

The probability that the machine accepts x after having read the first letter is given by the magnitude of the projection of the state $|\phi\rangle$ onto E_{acc}, ie

$$p_{acc} = \| P_a V_{\sigma_1}|\psi\rangle \|$$

Likewise the probability that the machine rejects x is

$$p_{rej} = \| P_r V_{\sigma_1}|\psi\rangle \|$$

The state, after reading σ_1 is, again, given by a three tuple

$$v' = T_{\sigma_1} v = \left(\frac{P_{non} V_{\sigma_1}|\psi\rangle}{\|P_{non} V_{\sigma_1}|\psi\rangle\|}, \right.$$
$$\left. p_{rej} + \| P_r V_{\sigma_1}|\psi\rangle \|, p_{acc} + \| P_a V_{\sigma_1}|\psi\rangle \| \right)$$

and these are strung together when dealing with words $x = \sigma_N \ldots \sigma_2 \sigma_1$

$$v' = T_x v = T_{V_{\sigma_N}} \ldots T_{V_{\sigma_2}} T_{V_{\sigma_1}} v$$

$$v' = T_{V_{\sigma_N}} \dots T_{V_{\sigma_3}} T_{V_{\sigma_2}} \left(\frac{P_{non} V_{\sigma_1} |\psi\rangle}{\| P_{non} V_{\sigma_1} |\psi\rangle \|}, \right.$$
$$\left. p_{rej} + \| P_r V_{\sigma_1} |\psi\rangle \|, p_{acc} + \| P_a V_{\sigma_1} |\psi\rangle \| \right)$$

These vectors defines a larger space of three tuples in $\mathcal{H}_n \times \mathbb{R} \times \mathbb{R}$. A norm on this space, $\|\|_2$, is given in [7] as

$$\| (\psi, p_{acc}, p_{rej}) \|_2 = \frac{1}{2}(\| |\psi\rangle \| + |p_{acc}| + |p_{rej}|) \tag{4.5}$$

The set of all valid states is given as $\mathfrak{D} = \{v \in \mathcal{H}_n \times \mathbb{R} \times \mathbb{R} \ | \| v \| \leq 1\}$

4.4.2 The One Way Measure Once QFA

The measure once quantum finite automaton (MO-1QFA) is nearly identical to the (MM-1QFA). One may define an MO-1QFA which highlights this. Let $Q = \{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}$, so there are four internal states. Define the initial state, $q_0 = |00\rangle$, the rejecting state $q_{rej} = |01\rangle$, and accepting state, $q_{acc} = |11\rangle$. Next, define the maps as

$$\begin{aligned}
\delta(|00\rangle, \sigma, |01\rangle) &= 0 \\
\delta(|00\rangle, \sigma, |11\rangle) &= 0 \\
\delta(|10\rangle, \sigma, |01\rangle) &= 0 \\
\delta(|10\rangle, \sigma, |11\rangle) &= 0 \\
\delta(|10\rangle, \sigma, |01\rangle) &= 0 \\
\delta(|00\rangle, ¢, |01\rangle) &= 1 \\
\delta(|10\rangle, ¢, |11\rangle) &= 1
\end{aligned}$$

So that, when the machine is in the non-halting space, it never enters the accepting or rejecting state for any symbol in the extended alphabet, except for the end marker ¢. Thus, any word that maps $|00\rangle$ to $|00\rangle$ will appear in the rejecting state, while those that map $|00\rangle$ to $|10\rangle$ will appear in the accepting state. Further on I will describe a class of 1QFA each of which distinguish a class of pairs of regular languages. This machine should be seen as equivalent to both that proposed in [1], and the 1QFA in [7], [8] and elsewhere in the literature.

4.5 Quantifying the Power of The 1QFA

The number of internal states in a single qubit is not just infinite, but is uncountable. However, if we consider the set of possible states that can be reached by operators approximated by a finite gate set, then that set of states is countable and infinite. Of course, we do not have access to these states directly. Upon observing the qubit we find only one of two possible outcomes. The information that we extract tends to be statistical in nature. Each element of the large set of pure states dictates the probabilities that each of the two possible outcomes may be measured. Thus, to extract the state of the system one needs to perform a very large number of measurements. The global phase information is lost entirely, and only the relative phase information can be gathered by measurement.

Due to the need to have distinct outcomes with distinguishable statistics under measurement, we find that the 1QFA is at a disadvantage relative to the classical probabilistic automaton (1PFA). The proof is given in [7], and relies on the fact that in order to recognize a language, there has to be a distinction between the statistics of accepting words in the language and those outside the language. It is shown in [7] that any 1QFA accepts only a proper subset of the regular languages and is thus less powerful than a classical deterministic one way finite state automaton.

4.5.1 The MM-1QFA Accepts Only Regular Languages

Recall that when accepting a language, a machine may accept it with a certain error bound. If the error bound is strictly greater than zero then the machine is said to accept the language with bounded error. An MM-1QFA can be designed to accept non-regular languages with unbounded error. However, it is true that the set of languages that the MM-1QFA accepts with bounded error forms a proper subset of the regular languages.

To show this, we should refer to the vectors defined in (4.2). Assume that we have a 1QFA, M, that accepts a language L with probability $\frac{1}{2} + \epsilon$. We see that the quantum finite automaton accepting a language L, is a machine that, to every word in the language, assigns a probability distribution over the accepting and rejecting outputs. The machine thus identifies the language L with a set of probability distributions in that words in L correspond to a different unitary operator. Acting on different qubits prepared in the same state, each unitary is identified with a different pure state, and thus a different probability distribution. Since the machine accepts the language,

each distribution in this set must assign a probability of greater than $\frac{1}{2}+\epsilon$ to the accept state, q_{acc}. Likewise, for each word *not* in L, each distribution must assign a probability of $\frac{1}{2}+\epsilon$ to the rejecting state, q_{rej}. M has the starting state q_0. We know that every language has a natural equivalence relation, $=_L$, such that if two words, u, u' are in L then $u =_L u'$ if $\forall y$ in Σ^* we find $uy \in L$ iff $u'y \in L$.

Let us chose $w \neq_L w'$. Then for all $y \in \Sigma^*$ if $wy \in L$ then $w' \notin L$. Let W be a set of words in L, that are distinct relative to $=_L$, i.e. chose one representative from each equivalence class and those choices form W. As seen in Section 4.3, it is a theorem that a regular language, W must be finite, i.e. there must be a finite number of equivalence classes. Both of our choices w, w' are in W as we have chosen them to be inequivalent relative to $=_L$. Next we define the vectors

$$\begin{aligned} v &= T_{w\$}(q_0,0,0) = (\psi, p_{rej}, p_{acc}) \\ v' &= T_{w'\$}(q_0,0,0) = (\psi', p'_{rej}, p'_{acc}) \end{aligned}$$

and note that if $T_y v$ represents a state that should be accepted (because $wy \in L$) then $T_y v'$ represents a state that should be rejected. We require that all words in L be accepted with probability greater than $\frac{1}{2}+\epsilon$ and rejected with probability lower than $\frac{1}{2}-\epsilon$. Likewise, all words not in L, namely those in $\Sigma^* - L$, must be rejected with probability greater than $\frac{1}{2}+\epsilon$ and accepted with probability lower than $\frac{1}{2}-\epsilon$. A complete description of the statistics for words $x \in L$ and $x' \notin L$ is

$$\frac{1}{2}+\epsilon \leq p^x_{acc} \leq 1 \tag{4.6}$$

$$0 \leq p^{x'}_{acc} \leq \frac{1}{2}-\epsilon \tag{4.7}$$

$$0 \leq p^x_{rej} \leq \frac{1}{2}-\epsilon \tag{4.8}$$

$$\frac{1}{2}+\epsilon \leq p^{x'}_{rej} \leq 1 \tag{4.9}$$

It should be pointed out that without renormalizing, the projection of the vector onto the non-halting state $P_{non} V_\sigma \psi$ must decrease in norm. This is because, the unitary action of V_σ preserves the norm, while projection onto any subspace results in a vector of equal or smaller size. Also, take two vectors of arbitrary length and separated by an angle θ. Imagine a single unitary transformation, acting on both vectors that could take them into some orthogonal space, essentially

rotating them out of plane in which they both lie. A projection of these vectors back onto the original plane produces two new vectors separated by an angle θ'. It is true that $\theta' \leq \theta$. From these notions we can see that there exists some fixed constant such that

$$\| T_y v - T_y v' \|_2 \leq c \| v - v' \|_2$$

Recall that v, v' represent distinct equivalence classes, since w, w' do. If we can show that there are only finitely many v, v', then we can show that there are only finitely many w, w' and thus only finitely many equivalence classes. If we can do this, then by Theorem 4.3.1, we can show that this model for computation accepts only the regular languages. We do this by showing that, in the space of all v there is a minimum distance relative to the norm (4.5). It should be clear that given any set of points chosen from a compact manifold, if there exists a minimum distance between these points, then the set must be finite. The space we are considering is compact because we are looking at only vectors v, such that $\| v \|_2 \leq 1$. We can see that $\| v - v' \|_2$ is bounded from below by $\frac{1}{c} \| T_y v - T_y v' \|_2$. If we can show that this is also bounded from below, then we have it that $\| v - v' \|_2$ has an absolute minimum, and thus there are only finitely many v and finitely many w. Let us define the vectors $T_y v, T_y v'$ corresponding to words $yv\$, yv^{prime}\$$

$$\begin{aligned} T_y v &= (\phi, p_{rej}^{yv}, p_{acc}^{yv}) \\ T_y v' &= (\phi', p_{rej}^{yv'}, p_{acc}^{yv'}) \end{aligned}$$

Recognize, now, that since $yv \in L$ and $yv' \notin L$, there exist bounds on the values of $p_{rej}^{yv}, p_{acc}^{yv}, \ldots$. We can gather these bounds from equations (4.6) to (4.9) and deduce the following

$$\begin{aligned} \| T_y v - T_y v' \| &\geq \frac{1}{2}(\| \phi - \phi' \| + 2\epsilon + 2\epsilon) \\ &\geq 2\epsilon \end{aligned}$$

In [7] they give an example of a regular language that a 1QFA cannot recognize with bounded error. This, then, proves that the 1QFA is actually less powerful than the 1DFA, since it accepts only a proper subset of the set of all regular languages. However there are languages for which the 1QFA has less distinguishable states than the 1DFA [23], [24].

4.5.2 A 1QFA Accepts Non-Regular Languages with Unbounded Error

It will be shown in more explicit detail later on how a single qubit can accept non-regular languages with unbounded error. It is also shown that no 1QFA can accept a non-regular language on an alphabet consisting of only one symbol, regardless of the error bound. It will be shown that, whenever one assigns a unitary matrix to each of the symbols of the alphabet, one necessarily has a group structure which is generated by those matrices. If the group structure has a finite set of equivalence classes, then the 1QFA necessarily accepts only regular languages. Normally a 1QFA works to return a single state for every word in the language L. The error comes from the action of every word in the complement $\Sigma^* - L$. If the group generated by the unitaries is infinite, then the set of all words is dense in a manifold that must contain the unitary corresponding to the accepting state. Thus, there will be words, not in the language, whose matrix representations approach arbitrarily close to the transformations reserved for the elements of the language. Upon measurement, these words seem as though they are in the language.

Chapter 5

Group Theory

Group theory is a subject in abstract algebra, where the sets of interest are endowed with special algebraic properties. An abstract algebra must have a binary operator that acts on two elements of the set, familiar examples being addition and multiplication. A group G is a set that has at least one binary operation, let us call it $*$ for now, and the following properties

- $\exists\, \mathbb{I} \in G$ s.t. $\mathbb{I} * a = a * \mathbb{I} = a, \forall\, a \in G$ (existence of identity)
- $a * b = c \in G, \forall\, a, b \in G$ (closure)
- $\forall a \in G, \exists\, a^{-1}$ s.t. $a * a^{-1} = a^{-1} * a = \mathbb{I}$ (inverse)
- $(a * b) * c = a * (b * c)$ (associativity)

From this point on, binary operations will not be identified with any special symbol like $*$. Instead, concatenation of elements of a group will implicitly assume a binary operation between them.

Abstract algebra is so named, as it deals with the simplest set of rules governing a mathematical object. These rules may apply to a whole range of mathematical objects. In the case of quantum mechanics, since we describe our states as vectors, the natural objects that would act on them are matrices. Many algebraic structures of interest have what are called matrix representations.

5.1 Finitely Presented Groups

Given a set of symbols in an alphabet Σ, we can identify them with generators of a group. Combine the generators as you would letters to form words. Combining the letters this way naturally assumes the existence of some binary operation. The free group is the set of all combinations of the generators and their inverses. It will be denoted by $\mathbb{G} = \Sigma^* * (\Sigma^*)^{-1}$ which is a set that includes the identity I. Notice that any combination of words in $\mathbb{G}$ is in Σ^*. If we let each symbol have an inverse, then every word has an inverse by simply reversing the order of the letters and changing all letters to their inverses

$$\begin{aligned} x &= abcd \\ x^{-1} &= d^{-1}c^{-1}b^{-1}a^{-1} \\ xx^{-1} &= abcdd^{-1}c^{-1}b^{-1}a^{-1} = I \end{aligned}$$

Thus every element of the group has an inverse. The free group is not particularly interesting. For example, if the alphabet is just $\Sigma = 1$, and the binary operation is addition, then Σ^* would be isomorphic to $\mathbb{Z}$, the integers.

The interesting groups are formed when added structure is enforced by what are called relaters. A relater is any word in the alphabet that is equivalent to the identity. Any relater can be fathomed. For example, given $\Sigma = \{a, b\}$ one may wish to have it that a commutes with b, and thus $ab = ba$. We can rewrite that word and receive the equivalent expression $aba^{-1}b^{-1} = \mathbb{I}$ which is a relater. Along with the generators a, b, we now have an entirely new algebraic structure, namely the infinite abelian group with two generators.

With the generators and the relaters, we can form what is called a presentation of a group. Groups defined by generators and relaters are presented in a the following fashion

$$\langle a, b, c, ...m | abc = I, ... \rangle$$

i.e.

$$\langle \Sigma | Relator1, Relator2, \ldots \rangle$$

The presentation of $\mathbb{Z}_2$ is

$$\langle b | bb \rangle$$

We concern ourselves here with finitely presented groups, i.e., groups having a finite set of generators and relaters.

Keep in mind that the form of the presentation $\langle|\rangle$ has absolutely nothing to do with Dirac bracket notation. We could just as easily use a notation of the form $\{|\}$ or $(|)$. The notation $\langle|\rangle$ is standard in combinatorial group theory.

No relater can be equal to some other relater by insertion of other relaters. As an example, consider the group $\langle a, b|a^2, b^8\rangle$, and the word $ab^4a^2b^4a = \mathbb{I}$. Words of this type are called trivial relaters and their inclusion does not change the group structure. The word $ab^4a^2b^4a$ is trivial as it is formed by the insertion of $b^8 = \mathbb{I}$ into $a^2 = \mathbb{I}$, and the insertion of a^2 into the resulting word. We can see here that the group structure creates a natural equivalence relation on the free group. If x and y are words in G and x can be turned into y by inserting and removing various identity words, then $x =_G y$. Given any word y, there is an equivalence class $[y]$ associated with that word. To determine if a word is in the same equivalence class as the identity is called the word problem. The word problem is, actually a very interesting one and will be discussed in greater detail below.

The presentation of a group G offers an equivalence relation $=_G$ on the set of all words $\mathbb{G}$. Two words are equivalent according to $=_G$ if one can transform one word into another using the relaters. Recall that the relaters are all equal to the identity, thus insertion of those words, deletion of those words, or concatenation with those words has no effect on the group element. For every element $w \in G$, there is an equivalence class $[w]$ and every word in $[w]$ is equivalent relative to $=_G$. As an example, consider the group

$$G = \langle a, b|a^2\rangle$$

It should be clear that a^2 is equal to the identity, thus $a^2 \in [I]$. The following statements are also true

$$a^{2n} \in [I]$$
$$aab \in [b].$$

5.1.1 The Group Structure on H, T

In quantum computation, the algorithms are generated by a set of elements chosen from the set of transformations on a qubit. This set of transformations is the Lie group U(2). By choosing an

arbitrary set of generators for which the free product is dense in U(2) one is actually choosing another group structure, namely that defined by the subgroup generated by the gate set. It is best to give that group structure in a finite presentation. Then any words equivalent to the identity can be gathered strictly from the relaters. As an example, I have chosen the gate set H and T defined in equations (3.1). This pair of matrices was chosen only because they are so well known in quantum computation. Their specific group structure is derived directly from the forms of the matrices, but it is not well known. So far we have been able to find the following group structure

$$G = \langle H, T | H^2, T^8, (HT^4)^8, H(HT^4)^4 H(T^4H)^4, T(HT^4)^4 T^7 (T^4H)^4 \rangle \tag{5.1}$$

The last two relaters comes from the fact that $(HT^4)^4 = -I$. This means that $(HT^4)^4$ commutes with both generators and hence every word. By adding the commutation relation of $(HT^4)^4$ with both H and T, we can arrive, algebraically, at the commutation relation of $(HT^4)^4$ with anything. To see this, let $W = (HT^4)^4$ and take an arbitrary word as $H^{i_1}T^{i_2}\ldots H^{i_{N-1}}T^{i_N}$ where $i_j, N \in \mathbb{Z}^+$

$$\begin{aligned}
WH^{i_1}T^{i_2}\ldots H^{i_{N-1}}T^{i_N} &= H^{i_1}WT^{i_2}\ldots H^{i_{N-1}}T^{i_N} \\
WH^{i_1}T^{i_2}\ldots H^{i_{N-1}}T^{i_N} &= H^{i_1}T^{i_2}W\ldots H^{i_{N-1}}T^{i_N} \\
&\vdots \\
WH^{i_1}T^{i_2}\ldots H^{i_{N-1}}T^{i_N} &= H^{i_1}T^{i_2}\ldots H^{i_{N-1}}WT^{i_N} \\
WH^{i_1}T^{i_2}\ldots H^{i_{N-1}}T^{i_N} &= H^{i_1}T^{i_2}\ldots H^{i_{N-1}}T^{i_N}W
\end{aligned}$$

Thus we can algebraically reach the commutation of W with any word using two relaters that ensure that W commutes with the two generators H, T showing that all those other possible relaters are trivial. Unfortunately, this presentation is definitely not complete. All the relaters are not known, so let us call the actual group structure $\tilde{G}$. Aside from the fact that it is uncertain at this point if these are all the relaters, there is an infinite number of words like W that are equal to the identity times a complex number of magnitude one. These are the phase operators. This means that G contains an infinite normal subgroup since $e^{i\theta}I$ commutes with every element of G. It is certainly possible that this subgroup can be finitely presented, and thus a finite set of extra generators can be added, completing the presentation, however this seems unlikely. Another thing to keep in mind is that when measuring a qubit, words containing any relaters that give global phases are indistinguishable from words that do not contain those relaters. The inclusion

of those relaters has no effect on the statistics of measurement. These relaters will act only to partition every existing equivalence class into smaller equivalence classes. This means that distinguishing those equivalence classes is impossible, however it has no effect on distinguishing existing equivalence classes.

The task of finding the finite presentation of the group generated by the, arbitrarily chosen, matrices H, T is by no means trivial. Even after enlisting the help of several group theorists, we cannot decide on the group structure.

5.2 How Hard is the Word Problem?

We have seen that, beginning with any word w, one may arbitrarily insert the relaters and form new words that are equivalent to w as the relaters are all equal to the identity. These new words are all equivalent to the original word w. Thus, we can develop an equivalence class $[w]$ in this fashion. Naturally, there is an equivalence class for the identity $[I]$. Imagine that one would like to distinguish every element in $[I]$ from every other possible word, $\Sigma^* - [I]$. This task is known as the word problem. A very similar problem is trying to distinguish an arbitrary equivalence class, $[w]$, from every other word.

It is a well known fact in group theory, that the word problem is, in general, a very difficult problem. A famous result in this subject is that there cannot exist a single algorithm which solves the word problem for every finitely presented group [16]. Furthermore, it turns out that even specific groups have no deterministic algorithms that solve their word problems. A good overview of the word problem can be found in [15]. The following theorem is courtesy of Sara Rees and was originally proven by Anisimov [17]. It can also be found in [18].

Theorem 5.2.1. *A finite state automaton can solve the word problem for a finitely presented group iff the group is finite*

Proof. Let M be a minimal automaton that solves the word problem for G, which has generating set X. Let s be a state of M and let v and w be strings over X which both lead to s. Now let u be a string from s to an accept state. That means that vu is accepted and also wu is accepted. So both vu and wu represent the identity element, and hence v and w both represent the same element (equal to u^{-1}) of G. So M must have at least as many states as there are elements of G.

Another proof, due to Muller [18] is given as follows. Let M be a finite automaton accepting every word in the equivalence class of the identity $[I]$ of some group G. There must exist words of arbitrary length $w = uu^{-1}$ such that there are no sub-words of u that are equal to the identity. We can rewrite w as $w = uvx$. Since M has a finite set of internal states of size $|Q|$, it must be true that we can find elements of the group such that both u and uv arrive at the same state s. If this is true, then we find that both $w = uu^{-1}$ and $w' = uvu^{-1}$ arrive at the accepting state. However, since the sub-word v is not identity, neither is uvu^{-1} and thus the automaton cannot accept only words in $[I]$. □

The simplest example of an infinite group is the free group on a single generator. This group is presented as

$$G = \langle a \rangle$$

One should notice that this is the abstract group with the very familiar representation known as the integers. An example of a finite group is $\langle a, b | a^n, b^m, aba^{-1}b^{-1} \rangle$. Given any word in this group we can rearrange the elements using the commutation relation to receive an equivalent word of the form $a^x b^y$. Now, without the other relations we still have an infinite group, since x and y are arbitrary. In this case we can use the other two identity relations to reduce any word to a word of finite length, since

$$\begin{aligned} a^x b^y &= a^n a^n a^n \dots a^i b^m b^m b^m \dots b^j \\ &= a^{\mathrm{mod_n(x)}} b^{\mathrm{mod_m(y)}} \end{aligned}$$

which leaves a finite group of size nm.

If we look at the 1DFA, we see that the task of recognizing a language is very similar to the word problem. In each case, the goal is a machine and an algorithm such that a special subset of Σ^* maps the machine to one state, typically called the accepting state. In the word problem, the machine will return, "yes, this word is equivalent to the identity", while the 1DFA, when recognizing a language, returns a similar answer "yes, this word is in the language". If we define a language L_I as the set of all words equal to the identity, given some finitely presented group, we have indeed defined a subset of the set of all words. This is very close to a formal language expression. In fact, finitely presented groups are more easily seen as formal grammars [18]. Regardless, given this subset, it is then natural to ask, can a 1DFA recognize every element of L_I

and thus solve the word problem? The fact that a 1DFA cannot solve the word problem indicates that it is a non-regular language. These aspects of classical computation were discussed in detail in [18], and some of the applications to the 1QFA are discussed in [9].

5.2.1 The 1QFA Solves the Word Problem With Unbounded Error

Let $\mathcal{L}/U(1)$, represent the Lie group which contains the set of all unitary transformations, mod $U(1)$, of a quantum system of dimension n. By mod $U(1)$, I mean that there are no elements in $\mathcal{L}/U(1)$ that are of the form $e^{i\theta}$I. where $\theta \in \mathbb{R}$. If there were elements of this sort, then words not equal to the identity would be mistaken for those equal to the identity when measuring the system, and thus the word problem is not solved. If one chooses any set of elements $\Sigma \subseteq \mathcal{L}$, then the free product of those elements forms a proper subgroup $G \subset \mathcal{L}/U(1)$. It may be that the group structure of G can be finitely presented, so we may find

$$G = \langle \Sigma | R \rangle$$

where Σ is the generating set, and R is the set of relaters. This group has a natural equivalence relation $=_G$ described in 5.1. It may be that this group is infinite or finite.

Theorem 5.2.2. *Given any finitely generated, infinite subgroup G of $\mathcal{L}/U(1)$, and a finite presentation for the group G, a 1QFA with $dim(E_{non}) > n$ can solve the word problem for this group with only unbounded error.*

Before we prove this we must state the following lemma

Lemma 5.2.3. *Any infinite subgroup G of a compact Lie group $\mathcal{L}$ is dense in some submanifold or continuous subgroup M of $\mathcal{L}$.*

In the case of $SU(2)$ any infinite subgroup is dense in one of only two sub-manifolds, either $SU(2)$ itself or some circle $U(1)$.

Proof. Since, we are identifying the group G with elements of a compact Lie group $\mathcal{L}$ which acts on a Hilbert space, we have a natural notion of its action on the quantum system as it evolves under words in G. In the case that G is infinite, the error is unbounded. The reason for this is that $\mathcal{L}$ is always compact for any quantum system of finite dimension n. Since $\mathcal{L}$ is compact,

any infinite subgroup is dense in some continuous subgroup. Naturally, the subgroup G and thus the submanifold M contain the identity and points arbitrarily close to the identity. This means that G must contain points arbitrarily close to the identity as it is dense in M. Choosing a word $w \in G$, not equal to but arbitrarily close to I, under repeated measurements we will find that the probability distribution associated with w, is arbitrarily close to that of I and is indistinguishable from it. □

A similar theorem can be immediately stated

Theorem 5.2.4. *Given any finitely generated, finite subgroup G of $\mathcal{L}/U(1)$, and a finite presentation for the group G, a 1QFA can solve the word problem for this group with bounded error.*

Proof. Clearly, if the group is finite, then there must be some distance between the elements of the group when considering their embedding in the manifold of $\mathcal{L}/U(1)$. If G is finite, then it must be true that there is a finite nonzero distance between any two elements of G. The distance is gathered from a metric on the manifold $\mathcal{L}$ in which G is embedded. This fixed minimum distance forms a bound on the error in distinguishing any element from the identity. Thus there is a distance between the identity and all other group elements. This distance in the Lie group translates into a distance in the probability distributions over the possible outcomes. Likewise, this distance can be used to distinguish the identity from all other group elements as long as sufficiently many measurements are made. □

Implicit in the preceding proof is the assumption of the following Lemma that is given without proof.

Lemma 5.2.5. *Given any finite subgroup G of a compact Lie group $\mathcal{L}$, and a distance measure on $\mathcal{L}$, then there exists a minimum distance between elements of G.*

5.3 $U(2)$ vs. $SU(2)$

$U(2)$ is the class of all unitary transformations of a two level system. This class includes global phases. If we, in an abstract group sense, mod out by these global phases, and construct the group $U(2)/U(1)$ we get $SU(2)$. It was an unfortunate accident that this research was begun and persisted in understanding the group structure of the gates (3.1) as they are dense in $U(2)$ and

as a consequence contain global phases. When talking about solving the word problem with a qubit, this becomes problematic because, under measurement, all these words are equal to the identity map. Let us imagine for a moment that the subgroup N of G (5.1) equal to global phases, could be finitely generated by repeated applications of a single word w. An example could be the following matrix for w

$$U_w = e^{i\delta\pi}\mathbb{I}$$

where delta is some irrational fraction. Again, if we could say that the subgroup generated by w, $\langle w \rangle$, was equal to N, we could then mod out N by simply stating that the word w itself was equal to the identity. By this we construct a new group

$$G' = \langle H, T | H^2, T^8, w \rangle \tag{5.2}$$

It would then be true that a single qubit could solve the word problem on G' with unbounded error. Of course, this is not how the actual normal subgroup of G (5.1) is formed. As it stands a single qubit cannot solve the word problem on G as it will confuse global phases for identities which it should not.

Chapter 6

Language Recognition and Language Distinction

While, at first it may seem that there is a serious contradiction between [1] and [7], this is not the case. In [1] we see a task that, in the classical, deterministic case, scales linearly with K, while in the quantum case the size complexity is constant. Does this then imply the existence of 1QFA that accept every word in a non-regular language while rejecting every word in the complement with bounded error? The answer is no. We see only the ability of the 1QFA to distinguish languages in a more space efficient way. The properties of a 1QFA that allow it to perform this task do not allow it to recognize non-regular languages with bounded error. However, the previous discussion on finitely presented groups affords us the ability to discuss in very general terms a very special ability of the 1QFA. Namely, the 1QFA can recognize non-regular languages with unbounded error.

6.1 Recognizing Non-Regular Languages with Unbounded Error

We have so far relied heavily on a single infinite subgroup of $U(2)$, namely that generated by H and T (3.1). This has brought up very interesting questions but also some severe and unnecessary burdens like those addressed above and in Section 5.3. To generalize these ideas we need to

consider all and any infinite subgroups of $SU(2)$ that have finite presentations. To be even more general we could quite easily consider any Lie group, $\mathcal{L}$ that acts on some n-dimensional Hilbert space $\mathcal{H}_n$. For now, let us focus on just $SU(2)$. Let G be an infinite subgroup of $SU(2)$ with generating set Σ. If x is a word in G. then let U_x be the corresponding matrix representation of x. We know that

$$\nexists w \in G \text{ s.t. } w = e^{i\theta}\mathbb{I}, \text{ and } \theta \neq 0$$

This ensures one cannot create global phases which the qubit does not distinguish from identity maps.

Let M be a 1QFA. Let the maps δ assign unitary transformations U_{σ_i} to each symbol of the alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$. Let G be the countable group generated by the elements U_{σ_i}.

Theorem 6.1.1. *If G is finite, then M accepts only regular languages and with only bounded error. If G is infinite, then M can accept non-regular languages with only unbounded error.*

Proof. Let M be defined by identifying the generators in G with the maps δ. Clearly, if G is finite, then it is equivalent to a group finite automaton as is pointed out in [8]. This means that it will accept only regular languages. It should be noted that if G is finite, then there is a fixed minimum distance, ϵ between any two nearest neighbours in G. The distance can be gathered from the metric on $\mathcal{L}$. This distance translates into a distance between states when the unitaries act on $\mathcal{H}_n$ which then translates into a distance between probability distributions. Essentially, this distance becomes your error bound and the machine defined by G accepts languages with bounded error.

If G is infinite, we know that the equivalence class $[I]$ is a non-regular language and that the machine accepts this language with unbounded error. □

6.1.1 Unary Languages

Unary languages are ones in which the alphabet has only a single symbol $\Sigma = \{\sigma\}$. In defining a 1QFA that accepts either regular languages with bounded error, or non-regular languages with unbounded error, one must suitably define the unitary transformation that is identified with σ. This unitary, U_σ, essentially defines the automaton. Given here is a general prescription for defining a 1QFA that *either* accept only regular languages with bounded error, or accept non-regular languages with unbounded error as well as (possibly) regular languages.

Let M be a 1QFA recognizing a non-regular language L with unbounded error, and let the alphabet be $\Sigma = \{0\}$. Recall that, the unitary transformation induced by 0, U_0, can have only one of two possible properties. We either have it that $U_0^n = I$ for some n, or $U_0^n \neq I \; \forall n \in \mathbb{Z}$. If $U_0^n = I$ for some n, then L is regular. Thus, for L to be non-regular, we must have $U_0^n \neq I \; \forall n \in \mathbb{Z}$. It is instructive to note the following lemma at this point

Lemma 6.1.2. *Any infinite subgroup G of a compact Lie group $\mathcal{L}$ is dense in some closed submanifold M of $\mathcal{L}$.*

Proof. We know that $\{U_0^{f(n)}\}$ must be infinite if the image of f is infinite. If it were not, then there must be elements $f(n_1), f(n_2)$ such that

$$\begin{aligned} U_0^{f(n_1)} &= U_0^{f(n_2)} \\ U_0^{f(n_1)-f(n_2)} &= I \end{aligned}$$

which is in contradiction with our original assumptions for $f(n_1) \neq f(n_2)$. If $\{U_0^{f(n)}\}$ is not dense in any M, then we can find $q \in M$ such that there are two elements $U_0^{f(n_1)}, U_0^{f(n_2)} \in \mathcal{L}$ such that they sit on either side of q and there are no other elements in $U_0^{f(n)}$ that sit closer to q. If $U_0^{f(n)}$ is dense in M, then this cannot be the case. Let it be that $U_0^{f(n_1)}, U_0^{f(n_2)}$ are separated by a distance ε. If this is true, that there are elements that only come as close as ε to one another, then, because M is compact (as $\mathcal{L}$ is compact), ε must partition M into a finite number of open subsets, indicating that $U_0^{f(n)}$ is finite. This is in contradiction with what we know about $\{U_0^{f(n)}\}$, that it is infinite. □

Next define a very generic form for our unary language L

$$L = \{0^{f(n)} | n \in \mathbb{Z}\} \tag{6.1}$$

Where $f(n)$ is a function of $\mathcal{Z}$. The image S_f of f is necessarily infinite, or else L would be regular.

$$f : \mathbb{Z} \to S_f \subseteq \mathbb{Z}$$

In general, it is not true that if $\{U_0^n | n \in \mathbb{Z}\}$ is dense in M, then so is $\{U_0^{f(n)}\}$ when S_f, the image of f, is infinite. To see this, note that M, the manifold is necessarily U(1). Consider a small connected submanifold $X \subset M$, namely a little line segment. Looking at a circle we can denote

X by the region between two points in the circle labelled a and b. Note that U_0 can be seen as a rotation by $\delta 2\pi$ where delta is an irrational number. For some m, it is true that $U_0^m \in X$. In fact, since U_0^n is dense in M, it is also dense in X. Thus there is an infinite set S_f of integers such that $\{0^n | n \in S_f\} \subset M$. We can even calculate what these integers are

$$a < \mathrm{mod}_{2\pi}(\mathrm{n}\delta 2\pi) < \mathrm{b}$$

This defines a language

$$L = \{U_0^n | a < \mathrm{mod}_{2\pi}(\mathrm{n}\delta 2\pi) < \mathrm{b}\}$$

This language is certainly non-regular and a 1QFA can accept it with unbounded error. The points a, b mark the distinction between the accepting region and the rejecting region. Points in both regions approach a, b with arbitrary closeness i.e. words both in L and not in L approach arbitrarily close to the same points a, b. Thus there are words in L that are indistinguishable from those not in L and the error of rejection is unbounded. One may ask the following question, with the freedom to chose U_0 by adjusting δ, would it be possible to accept, with bounded error, the following non-regular languages?

$$L = \{0^n | n \text{ is prime}\} \tag{6.2}$$

$$L = \{0^{n!+n} | n \in \mathbb{Z}\} \tag{6.3}$$

$$\tag{6.4}$$

Could these strange sequences be forced to fall within that small region defined above simply by choosing the appropriate value U_0? No attempt at answering these questions will be found here. It is assumed that these problems remain open.

6.2 Distinguishing Non-Regular Languages

The group theoretic word problem has opened some interesting new questions. Could it be that a classical 1DFA cannot distinguish the languages discussed in the previous chapters? While it will be shown that a single qubit can distinguish non-regular languages and also more importantly, context free grammars, questions remain about the classical analog. The free group is used to develop a simple example of a pair of languages that are derived from the equivalence class of the identity $[I]$. These languages will bear out to be classically distinguishable.

6.2.1 The Quantum Case

The ability of the 1QFA to accept non-regular languages with unbounded error is also what gives it the ability to distinguish non-regular languages deterministically. Recall that for any quantum system $\mathcal{H}_n$, there is a Lie group $\mathcal{L}$ acting on it. If we choose elements from $\mathcal{L}$, those elements generate a subgroup $G \subset \mathcal{L}$. Let G have a finite presentation given by

$$G = \langle \Sigma | R_i \rangle$$

If G is infinite, there are non-regular languages that can be defined by equivalence classes $[I], [w]$ in G. Since G is infinite, it is dense in some sub manifold of $\mathcal{L}$. This submanifold not only contains the identity, but it must contain unitary transformations U' that will take any given state ψ to an orthogonal state. Since G is dense we can approximate U' to any accuracy. Let U_w be the matrix representation of the word $w \in G$ which approximates U'. We thus have it that, given any $\epsilon > 0$

$$\exists w \in G \text{ and states } |\psi\rangle, |\phi\rangle \in \mathcal{H}_n \text{ s.t.}$$

$$U_w|\psi\rangle = |\phi\rangle \quad \text{and} \quad \langle\psi|U_w|\psi\rangle < \epsilon$$

which is in contrast with words in $[I]$ where we find

$$\langle\psi|U_I|\psi\rangle = 1$$

Thus there exist nearly orthogonal states, one of which can be reached from the other by words in G. If we now consider the equivalence classes $[I]$ and $[w]$ in G. we note that

$$U_v|\psi\rangle = |\psi\rangle, \forall v \in [I]$$

and

$$U_u|\psi\rangle = |\phi\rangle, \forall u \in [w]$$

We have seen already that $[I]$ and $[w]$ can be identified with formal languages L_I and L_w over the alphabet Σ. With all of this defined, we can have the more general theorem

Theorem 6.2.1. *A 1QFA can distinguish the very generally defined languages L_I and L_w with any prescribed accuracy.*

Proof. By definition, we have identified every word in L_I with a matrix in the equivalence class $[I]$, these matrices form the maps for our 1QFA that distinguish the languages. We define the measure once 1QFA as $M(Q, \Sigma, \delta, q_I, q_w)$ where q_I, q_w are two distinguishable pure states. They form both the starting state and the set of final states. If one were to wish to have the-non halting subspace as distinct from the two states that distinguish $[I]$ from $[w]$ then the following maps could be added

$$\delta(\psi, \sigma, q_I) = \begin{cases} 1 & \text{for } \sigma = ¢ \\ 0 & \forall \sigma \neq ¢ \end{cases}$$

$$\delta(\phi, \sigma, q_w) = \begin{cases} 1 & \text{for } \sigma = ¢ \\ 0 & \forall \sigma \neq ¢ \end{cases}$$

Thus we identify q_I, q_w with $|\psi\rangle$ and $|\phi\rangle$ respectively. The maps $\delta(q, \sigma, q')$ can be given in the form of the unitary matrices which we gather from the matrix representations of the generators. It clearly follows, that on the input of any word in L_I and a subsequent measurement we receive q_I. The reverse is also evident, since, upon reading any word in $[w]$, the unitary that is applied produces the orthogonal state ϕ. After the input of the end-of-tape symbol and a measurement, the machine will produce q_w. □

6.2.2 An Example

We have shown that the set of algorithms on H, T that map a qubit from $|\psi\rangle$ to $|\psi\rangle$ forms a non-regular language L_I. The languages can be defined by the group structure G, given in equation (5.1)

$$L_I = \{x \in [I]_G\}. \tag{6.5}$$

The words that map $|0\rangle$ to $|1\rangle$ forms another non-regular language

$$L_{NOT} = \{x \in [HT^4H]_G\}. \tag{6.6}$$

The qubit can clearly distinguish these languages with probability 1. The question as to whether or not a 1DFA can distinguish these languages is open. Once it is completed, it will be of interest to Sarah Rees at Newcastle who is also trying to solve this problem in terms of what we both call a "restricted word problem ". The qubit can distinguish the languages $L_I, L_{NOT} = L_{[HT^4H]}$

relative to any of the following groups

$$G_1 = \langle H, T | H^2, T^8, (HT^4)^8, (HT^4)^4 \rangle \tag{6.7}$$

$$G_2 = \langle H, T | H^2, T^8, (HT^4)^8 \rangle \tag{6.8}$$

$$G_3 = \langle H, T | H^2, T^8 \rangle \tag{6.9}$$

Could it be that new relaters may be found for $\tilde{G}$ that induce a nonzero intersection between the languages we defined in (6.5) and (6.6). This seems unlikely. As long as the word $HT^4H \neq I$, we will always find that the group theoretic equivalence classes are disjoint. Let $\tilde{G}$ be the exact group structure defined by the matrices H and T, and let G_i be any group in equations (6.7) to (6.9). We can imagine many more groups that have all the known relaters and new ones that simply preserve the fact $HT^4H \neq I$. Let there be equivalence classes $[HT^4H]_{\tilde{G}}, [HT^4H]_{G_i}, [I]_{\tilde{G}}, [I]_{G_i}$. It is true that any new relaters that may be found for the group $\tilde{G}$ will preserve the relation

$$[I]_{G_i} \cap [HT^4H]_{\tilde{G}} = \emptyset \tag{6.10}$$

$$[HT^4H]_{G_i} \cap [I]_{\tilde{G}} = \emptyset \tag{6.11}$$

as long as none of the new relaters are $HT^4H = I$. Say that we found a word $w \in [HT^4H]_{\tilde{G}}$ and also $w \in [HT^4H]_{G_i}$. It must be true that we can rewrite $w = HT^4H$ according to the relators of $\tilde{G}$. This word cannot be made into the identity using the relaters of G_i, and thus cannot be in $[I]_{G_i}$ which is a contradiction. The reverse, wherein $w \in [HT^4H]_{G_i}$ and also $w \in [I]_{\tilde{G}}$ gives a similar contradiction.

6.2.3 Distinguishing Languages is Distinguishing Probability Distributions

When distinguishing languages with a 1QFA, the task becomes equivalent to distinguishing statistics. In the case of L_I, L_{NOT} the statistics of L_I is $|0\rangle$ with probability one, while the other languages is $|1\rangle$ with probability one. The statistics can be distinguished with a single measurement. Of course, distinguishing statistics can only be done in the infinite measurement limit. To see this, take the language $L_{NotQuiteNOT}$, all the words of which map $|0\rangle$ to $|\phi\rangle = \sqrt{\varepsilon}|0\rangle + \sqrt{1-\varepsilon}|1\rangle$, for some tiny ε. The smaller ε becomes, the more measurements one may need in order to find the $|0\rangle$ state and discover that $L_{NotQuiteNOT}$ is not quite L_{NOT}. What is the smallest difference in the probability distributions that still allows one to distinguish

them? Because the languages are disjoint, one should never find a word in $L_{NotQuiteNOT}$ that maps the qubit to the same pure state as any word in L_{NOT}. However, many different pure states have the same measurement statistics when measured in the same computational basis. While this indicates that there are many equivalence classes that cannot be distinguished with a single choice of basis, there remains many that can. As an example, take the next simplest case, distinguishing the three languages given by L_I, L_{NOT} and L_H where L_H is the set of all words that are in $[H]$. The statistics of L_H are 50% $|0\rangle$ and 50% $|1\rangle$. Even after a few measurements, it can be distinguished from the other languages. What is the limit to the number of languages, given by the group structure (5.1), that can be distinguished by a qubit? This limit is simply the limit to which one can distinguish probability distributions. Let us say that if one has to perform more than N measurements, to properly distinguish two distributions, then one cannot distinguish the two languages. We can talk about the probability, given N measurements, that the statistics will be distinguishable. If G is finite, $\exists \epsilon > 0$ and we can distinguish all equivalence classes of G. If G is infinite $\epsilon \to 0$ and we can distinguish all the equivalence classes with only unbounded error.

6.3 Distinguishing Unary Languages and Modular Adding in FA theory

The problem addressed in [1] can be translated into a task of distinguishing languages. First note that the task amounts to counting the length of a word in one symbol. This kind of language is called unary, namely where $\Sigma = \{0\}$, i.e. the alphabet has one symbol. Let us define a language, L_{Odd} as all words of length mK for m odd and some fixed $K = 2^i, i \in \mathbb{Z}$

$$L_{Odd} = \{0^{K(2n+1)} | \text{ for } n \in \mathbb{Z}\}. \tag{6.12}$$

Likewise we define L_{Even} as all words of length mK for m even.

$$L_{Even} = \{0^{K(2n)} | \text{ for } n \in \mathbb{Z}\}. \tag{6.13}$$

These languages are regular. This is analogous to [1] in that it was always possible to construct a classical deterministic machine that could perform the sum mod $2K$ for any K. The classical deterministic machine described in [1] is equivalent to a 1DFA. It would be significant to show

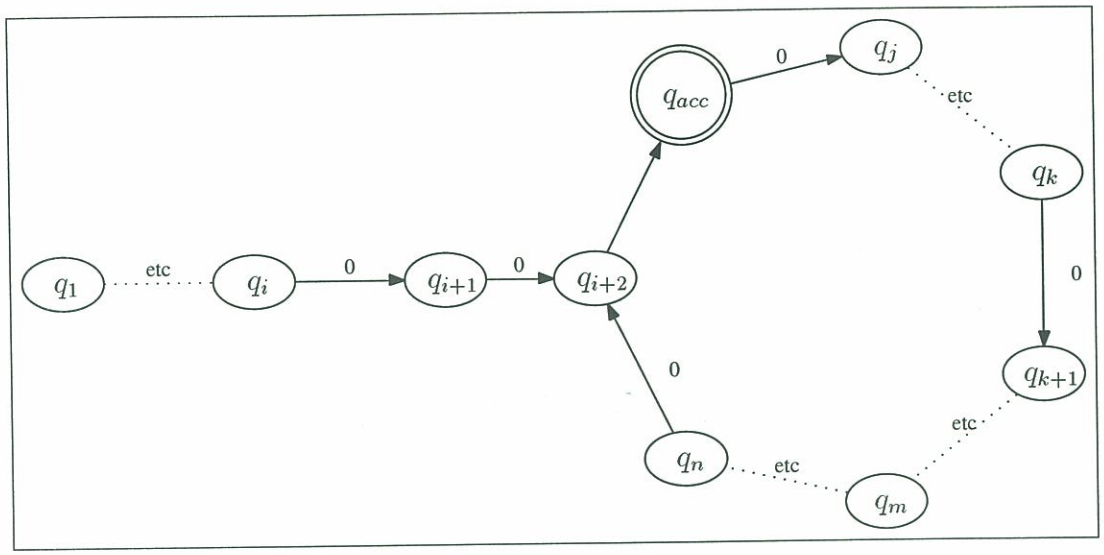

Figure 6.1: The evolution of a unary language machine forms a ring

that, while a 1DFA cannot distinguish the languages L_{Odd}, L_{Even}, a single qubit can. In fact, since these two languages are regular, a 1DFA *can* distinguish them. However, the machine that does the job must have at least $2K$ internal states. This was proved in [1].

6.3.1 The Deterministic Case

In order to show that a 1DFA needs $2K$ internal states to distinguish L_{Odd} from L_{Even}, we note an interesting property of machines that recognize unary languages. Let L be some unary language, and let M be a minimal automaton that recognizes L, with $|Q|$ internal states. The alphabet is $\Sigma = \{0\}$. Let the maps which define M be given in the following, sequential order

$$\begin{aligned} \delta(0, q_1) &= q_2 \\ \delta(0, q_3) &= q_4 \\ &\vdots \end{aligned} \tag{6.14}$$

Given this kind of ordering of the internal states, a single drawing of the evolution of the system becomes extremely general, and is given in fig. 6.1. We can simply relabel the states, $q_i = i$ and we find that the action of reading the symbol 0, becomes adding one to i. Since, the machine is

finite, and every language has arbitrarily long words, eventually, the reading of 0 should point to a state that has already been crossed. This means that, beyond a certain length of zeros, any unary machine with $|Q|$ internal states, acts exactly as an adder mod X for $X \leq |Q|$. Thus, every possible algorithm can be seen as modular adding. The only exceptions to this rule are algorithms that have the property that for some q_i, $\delta(0, q_i) = q_i$. These types of algorithms terminate at state q_i, and recognize the set of languages of the type $0^i\{0\}^*$ where i is a fixed integer, and star is the free product on the set $\{0\}$. Thus those machines accept only strings longer than some fixed length.

Looking at Figure 6.1 we also see that, regardless of $|Q|$ we find the following overlapping transitions

$$\begin{aligned} \delta(0, q_{i+1}) &= q_{i+2} \\ \delta(0, q_n) &= q_{i+2} \end{aligned} \tag{6.15}$$

Thus, translating this to modular addition, we find the following identification

$$q_{n+1} \rightarrow q_{i+2}$$

and we find that we have identified q_{n+1} with q_{i+2}. We have already seen that the set of all possible sums mod $2K$ is a modular set which was formed by identifying 0 with $2K$, and the resulting mathematical structure, is a sort of adder mod $2K$. However, for the machine in figure 6.1, if $n+1$ is less than $2K$, one cannot use it to add mod $2K$ successfully, and thus it cannot be used to distinguish the languages L_{Odd}, L_{Even}.

6.3.2 The Quantum Case

A single qubit can distinguish the languages L_{Odd}, L_{Even} deterministically for all K. The 1QFA distinguishing these two languages is given by

$$M = \{Q, \Sigma, \delta, |0\rangle, \{|0\rangle, |1\rangle\}\}$$

where $Q = |0\rangle, |1\rangle$, and $\Sigma = \{0\}$. The maps are given as

$$\begin{aligned}
\delta(|0\rangle, 0, |0\rangle) &= \frac{1}{2}(1 + e^{i\pi/K}) \\
\delta(|0\rangle, 0, |1\rangle) &= \frac{1}{2}(1 - e^{i\pi/K}) \\
\delta(|1\rangle, 0, |0\rangle) &= \frac{1}{2}(1 - e^{i\pi/K}) \\
\delta(|1\rangle, 0, |1\rangle) &= \frac{1}{2}(1 + e^{i\pi/K})
\end{aligned} \tag{6.16}$$

This set of maps defines the four elements of the unitary matrix U_0, such that

$$\begin{aligned}
U_0^K &= NOT \\
&= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}
\end{aligned} \tag{6.17}$$

(6.18)

which is the NOT gate. We see that under an odd number of applications of U_0, the qubit is mapped to $|1\rangle$ and under an even number, it is mapped to $|0\rangle$. The qubit can then be measured and the languages are deterministically distinguished.

The machine defined by the maps δ accepts L_{Even} with bounded error. The abstract finite group known as $\mathbb{Z}/K$ is generated by a single element and has the following finite presentation.

$$G = \langle a, b | ab, a^K, b^K \rangle \tag{6.19}$$

It should be clear that the set $\{U_0^n | n \in \mathbb{Z}\}$ is a matrix representation of that group. This group is finite and because of this it forms finite subgroup of $U(2)$, and there is a fixed minimum distance ϵ between all nearest neighbours. This distance forms an error bound and the machine employing U_0 accepts only regular languages with bounded error. The action of the remaining words in $\Sigma^* - L_{Even}$ becomes arbitrarily close to the identity as K becomes larger. As K becomes large the set of maps, $\{U^n | n \in \mathbb{Z}\}$, becomes large (as does $\mathbb{Z}/K$, and thus the set of internal states that are reached under the maps δ becomes large. This set is uniformly spread over some U(1) submanifold of the set of pure states. The distance between those pure states becomes small. The error is inversely proportional to the distance between those pure states, and thus it becomes large.

This still leaves us with an unanswered question. We have not shown why these added abilities are not in conflict with [7], [8] and [10].

6.4 The Answer

The machine described in [1] and reproduced in Section 2.3, is equivalent to a deterministic one way finite state automaton (1DFA), in which the alphabet Σ is the integers $\{0, 1, \ldots, 2K-1\}$. The quantum machine that is considered in Section 2 is the same as that considered in [7]. To show this, one may translate [1] directly into automata theory. In [1] they use the notation

$$l_n = f_{n-1}^n(k_n)$$

to mean that, when k_n is read, and the system is in state l_{n-1} it undergoes a transition to state l_n. This is the same as defining the maps $\delta(q, \sigma) = q'$. In other words, the deterministic automaton model is implicit in [1].

While once it may have seemed that a contradiction existed between [1] and the existing work on the 1QFA, it has been shown that there is none. The secret that resolves this contradiction comes from the promise. Recall that the job that the integrating qubit is doing, is simply distinguishing the even sums from the odd sums (integrals). The classical task is reduced to counting mod $2K$. In terms of a 1DFA we can reduce the language to one symbol and this is explained in Section 6.3. In the quantum case, the promise amounts to a restriction on the unitary transformation that is applied to the qubit. In the case of an odd number of 1's, the resulting unitary is the NOT gate, while in the case of an even number, only the identity is applied. Each class of functions is identified with a different class of unitary transformations, one equivalent to the identity, the other, the NOT. This means that the set of words, in your choice of universal gate set, is restricted to words in either the equivalence class of the NOT or the identity. Normally, the qubit would be passed all words in Σ^*. Only a subset of all words, $L \subset \Sigma^*$, namely the language which the qubit accepts, would cause the machine to go into some prescribed accept state. By referring to the group structure, it can be shown that the language that corresponds to all transformations equal to the identity is not regular. If the qubit could reject all words not equal to the identity, and accept all words equal to the identity, then the qubit would recognize a non-regular language and this would contradict the existing work on the 1QFA. The qubit *does* recognize this language, but with unbounded error. This result was first published in [8].

6.5 Distinguishing Elements in the Free Group

In [8] we see that a single qubit solves the free group with unbounded error. The group given there has only one generator and no relaters, hence it is free

$$G = \langle a \rangle \tag{6.20}$$

It is suggested in [8] that this is similar to the fact that a probabilistic machine can accept all words in a non-regular language with bounded error, while rejecting all words with unbounded error. The probabilistic machine cannot reject all words in the complement with bounded error, and thus a 1PFA cannot recognize a non-regular language.

Along these lines, it can be shown that a qubit can distinguish two non-regular languages with unbounded probability of success. If we just explicitly add the inverse of a to the representation of G (6.20)

$$G = \langle a, b | ab \rangle \tag{6.21}$$

then, relative to this group there are two distinct equivalence classes $[a]_G$ and $[I]_G$. We now define two languages

$$\begin{aligned} L_a &= w \in [a]_G \\ L_{Ident} &= w \in [I]_G \end{aligned}$$

and define the following unitary

$$V_a = \begin{pmatrix} \cos\delta & \sin\delta \\ -\sin\delta & \cos\delta \end{pmatrix} \tag{6.22}$$

(6.23)

with V_b defined by the matrix $V_b = V_a^{-1}$. If δ is an irrational fraction, close to 1, times $\pi/2$, then $V_a^n \neq \mathbb{I}\, \forall n \in \mathbb{Z}$, and its action on $|0\rangle$ will approach an orthogonal state

$$\langle 0 | V_a | 0 \rangle \approx 0$$

Thus words in $[a]$ will, under measurement of the qubit, be statistically, arbitrarily distant from those in $[I]$.

The exact classical case, using a probabilistic 1PFA would be difficult to work out, and a one bit solution may exist. It is likely that a single classical bit, under the influence of stochastic matrices, would not be able to assign two arbitrarily distinct probability distributions over the two distinguishable states. It is instead more likely that the 1PFA will map every word in L_a to a distinct probability distribution which leans to $|1\rangle$ by a margin greater than $\frac{1}{2} + \epsilon$.

6.5.1 From the Free Group to Distinguishing Languages

What does the previous section tell us about distinguishing languages? Well, it points to another algorithm that a single qubit can do. Take the following language

$$L_{a^n b^n} = \{a^n b^n | n \in \mathbb{Z}, n > 0\} \tag{6.24}$$

the language where a number of a's is followed by the same number of b's. This language is not regular. One may see this by noting that in order to know if there are an equal number of a's as b's one needs to count them. Since n is not constrained, one needs an arbitrarily large number of internal states to count n. A 1QFA with only two internal states can accept this language with unbounded error. To see this, just assign to the symbol a the same unitary transformation as was described in the previous section

$$V_a = \begin{pmatrix} \cos\delta & \sin\delta \\ -\sin\delta & \cos\delta \end{pmatrix} \tag{6.25}$$

$$\tag{6.26}$$

Again, let δ be an irrational fraction, close to 1, times π. We can see that V_a maps a state to a nearly orthogonal state

$$\langle\psi|V_a|\psi\rangle < \varepsilon$$

for any ε we wish, since we just have to adjust δ. Also, assign the inverse of V_a to the symbol b so that

$$V_b = V_a^{-1}$$

We can see immediately, that a number of applications of V_a followed by an equal number of applications of V_b will result in an identity operation. Thus every element of (6.24) maps a state to itself. However, if we accidentally threw in an extra application of V_a, the state would be

mapped to an almost orthogonal state. From here we can see that a single qubit can distinguish (6.24) from another non-regular language

$$L_{a^{n+1}} = \{a^{n+1}b^n | n \in \mathbb{Z}, n > 0\} \tag{6.27}$$

Of course, many more examples exist, for example

$$L_{a^{n+m}} = \{a^{n+m}b^n | n \in \mathbb{Z}, n > 0\} \tag{6.28}$$

This language can also be distinguished from $L_{a^n b^n}$.

6.5.2 In More General Terms

Let $G = \langle a \rangle$ be the free group represented by the single matrix U_a where $U_a^n \neq \mathbb{I}, \forall n \in \mathbb{Z}$. Furthermore, let U_a be an element of some lie group $\mathcal{L}$ which acts on a Hilbert space $\mathcal{H}_n$. $\mathcal{L}$ is compact, and the set $\{U^n | n \in \mathbb{Z}\}$ is dense in a $U(1)$ submanifold M of $\mathcal{L}$. There must exist a $U' \in M$ and a state $|\psi\rangle \in \mathcal{H}_n$, such that

$$\langle \psi | U' | \psi \rangle = 0 \tag{6.29}$$

Since G is dense in M, it must be true that

$$\exists n \in \mathbb{Z}\ s.t.\ \forall \varepsilon \in \mathbb{R}\ \parallel U_a^n - U' \parallel < \varepsilon$$

We can see that U_a^n is naturally identified with the group element $a^n \in G$, and we can create a matrix representation of the equivalence class of $[a^n] \to [U_a^n]$. Because the action of $[U_a^n]$ on $|\psi\rangle$ is to map it to a nearly orthogonal state we can see that using a quantum system $\mathcal{H}_n$, we can distinguish these two equivalence classes. The creation of these equivalence classes is only possible with the introduction of a new matrix $U_b = U_a^{-1}$, and thus new symbol, b. For, without these, we could never have an identity matrix, since by definition $U_a^n \neq \mathbb{I}, \forall n$.

Chapter 7

Distinguishing Non-Regular Languages: The Classical Case

A 1DFA can distinguish non-regular languages. However, can a 1DFA distinguish the kinds of languages that have been outlined here, namely those derived from finitely presented infinite groups? One possible method of finding an automaton that distinguishes these languages is to find disjoint regular covers of each language. The question remains, do these languages have minimal regular covers? This question may not be decidable. To understand this, we need to flesh out the idea that finitely presented groups and formal grammars are related. We see this in [18]. Furthermore, looking at [14], we see that the question of whether or not a context free grammar has a minimal regular cover is not decidable.

There will be great difficulties in finding classical machines that distinguish the non-regular languages defined by infinite groups. In order to understand why, we need to look at a special type of language construction called a formal grammar. First, though, we must look at the mechanism that allows us do distinguish languages of any type.

7.1 Covers of Formal Languages

If one wishes to know if two languages are distinguishable, one needs to find what are called minimal regular covers for the two languages. Let it be that two languages, say, L_I, L_{NOT} are subsets of two disjoint languages R_1 and R_2 respectively, where $R_1 \cap R_2 = \emptyset$. Thus we require

$L_I \subseteq R_1$ and $L_{NOT} \subseteq R_2$. Also, let it be that R_1 and R_2 are regular. If this is the case, then the two languages are distinguishable. Now, if we can show that either R_1 or R_2 is equal to Σ^* then we know that they are not disjoint. A given machine that accepts every word in R_1 with probability 1 identifies every word in that language with exactly one state. That is, upon reading any word in R_1, the machine ends up in the same state for all words in R_1. Likewise, for R_2. Let the machine M accept every word in both R_1 and R_2, and distinguishes them by identifying each word in R_1 with state q_1 and each word in R_2 with state q_2. Now, if $R_1 \cap R_1 \neq \emptyset$ then there exists at least one element in R_1 and R_2 that map to the same point. However, this means that every element in R_1 and R_2 must be identified with the same point, and the languages are not distinguished.

We have seen already that quantum systems have the ability to recognize non-regular languages with unbounded error. We have also seen that a 1QFA distinguishes non-regular languages by embedding them in one of two distinct equivalence classes of a finitely presented group of operations. Take a single qubit as a 1QFA and consider some gate set that has two elements which generate an infinite subgroup. This subgroup will have an identity equivalence class $[I]$ which forms a non-regular language, and it will also have many other equivalence classes $[w]$ that can be distinguished from $[I]$. We may define languages, $L_{[I]}$ and $L_{[w]}$, based on these equivalence classes. It will certainly be true that these languages will have a very large number of non-regular sublanguages. Recall the discussion on regular covers, where L_1 was embedded in R_1. In the quantum case we may have a non-regular language L_i embedded in the non-regular language $L_{[I]}$. Let $\{L_i\}$ be some set of languages (regular or not), such that $L_i \subset L_{[I]}$. Also, define $\{L_w\}$ as the set of languages for which $L_w \subset L_{[w]}$. It is then true that the qubit can distinguish $\{L_i\}$ from $\{L_w\}$ either deterministically or with some error bound, by the use of non-regular covers.

7.1.1 Classically Distinguishing Classes in a Free Group

Let us look again at the simplest possible example, namely the free group on a single generator

$$G = \langle a, b | ab \rangle$$

The identity equivalence class is made up of the set of all words consisting of equal numbers of a and $b = a^{-1}$. This is a non-regular language in itself and forms a non-regular cover for plenty

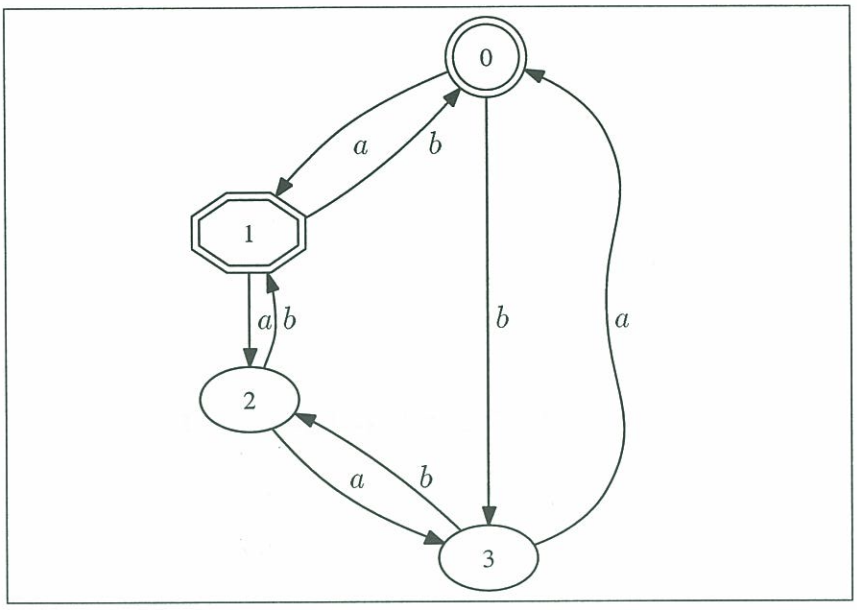

Figure 7.1: An automaton distinguishing L_1 from L_2

of other non-regular languages. From that set lets choose the following two languages over the alphabet $\Sigma = \{a, b\}$

$$\begin{aligned} L_1 &= \{a^n b^n | n \in \mathbb{Z}\} \\ L_2 &= \{a^{n+1} b^n | n \in \mathbb{Z}\} \end{aligned}$$

These languages were found in Section 6.5.1. We can build an automaton that distinguishes these languages quite easily. It has no more than four states. A diagram for it is shown in figure 7.1. All words in L_1 map the automaton to state 0, while all those in L_2 map it to state 1. With m states, this automaton also accepts the arbitrarily sparse language $L_m = a^{\mathrm{mod_m(n)}} b^{\mathrm{mod_m(n)}}$, as m can be arbitrarily large. What does it mean for a language $L \subseteq \Sigma^*$ to be sparse relative to Σ^*? The notion of sparse can be understood in terms of finding *minimal* covers of languages. It is clear that as m grows large, there will be fewer elements of L_m within the set of all words of some given length l. Thus, as m grows, $L(m)$ becomes sparse. Another point to note, is that, the number of equivalence classes defined by $=_L$ grows with m. A regular cover of a non-regular

language may have infinitely many elements, but finitely many equivalence classes $=_L$. If the covering language is made arbitrarily sparse with m, then as m grows, it approaches a sort of minimal regular cover for L_1. However, there is no limit to this process for L_m, and the language can become arbitrarily sparse. L_2 can be covered by the language $L_{mc} = a^{\mathrm{mod_m(n+c)}}b^{\mathrm{mod_m(n)}}$, for $c = 1$, and thus the two are distinguished. As c grows, the automaton that recognizes it must also grow. It is unclear at this point how to use this result to find two regular languages that are distinct and cover the languages in (6.5), (6.6), namely L_I and L_{NOT}.

7.2 A Simple Example of Language Distinction

The NOT gate is formed, using H, T as follows

$$NOT = HT^4H$$

Imagine a set of words that were formed by concatenating an arbitrary bunch of NOT gates, written in this fashion. In a formal language sense, consider the two languages

$$L_{even} = (HT^4H)^{2n}, \quad n \in \mathbb{Z}$$
$$L_{odd} = (HT^4H)^{2n+1}, \quad n \in \mathbb{Z}$$

If a single qubit reads these languages, L_{even} maps $|0\rangle \to |0\rangle$ and L_{odd} maps $|0\rangle \to |1\rangle$. Clearly we have chosen $L_{even} \subset L_I$ and $L_{odd} \subset L_{NOT}$. In the quantum case, these are the two simplest languages one may distinguish deterministically, and it requires only a single qubit. It can be shown that one can distinguish these two languages with two bits, or four states. It can also be shown that one cannot distinguish these two languages with only two states.

Theorem 7.2.1. *A single bit cannot distinguish L_{even} from L_{odd}.*

Proof. Case 1; one bit.

If we look at the set of all maps from one bit to one bit, we see it is the set of maps from $\{0,1\}$ to $\{0,1\}$. The maps are

$a = I$			$b = Const_1$			$c = Const_0$			$d = NOT$		
0	$\to$	0	0		0	0	$\to$	0	0		0
				$\searrow$			$\nearrow$			X	
1	$\to$	1	1	$\to$	1	1		1	1		1

When attempting to distinguish the two languages, we assign to each letter in the alphabet, one of the above transitions. This corresponds to eight different possibilities. Let $W = HT^4H$

1. $H = I, T = Const_{1,0}$

$$\begin{aligned}(HT^4H)^{2N} &= (T^4)^{2N} \\ &= (Const_{1,0})^{2N} \\ &= (Const_{1,0})^{2N+1}\end{aligned}$$

   and thus does not distinguish $2N$ from $2N + 1$

2. $T = I, H = Const_{1,0}$

   As in the previous case

$$\begin{aligned}W^{2N} &= Const_{1,0}^{2N} \\ &= Const_{1,0}^{2N+1} \\ &= W^{2N+1}\end{aligned}$$

3. $H = NOT, T = I$ and $T = NOT, H = I$

$$\begin{aligned}W^{2N} &= (NOT^2)^{2N} \\ &= (I)^{2N} \\ &= W^{2N+1}\end{aligned}$$

   a similar argument applies in the reversed case

4. $H = NOT, T = Const_{1,0}$ and $H = Const_{1,0}, T = NOT$

$$\begin{aligned}W^{2N} &= (NOT^2.Const_{1,0}^4.NOT)^{2N} \\ &= (NOT^2.Const_{1,0})^{2N} \\ &= (NOT^2.Const_{1,0})(NOT^2.Const_{1,0})^{2N-1} \\ &= (NOT^2.Const_{1,0})\end{aligned}$$

the final form is independent of the exponent $2N$ and thus does not distinguish $2N$ from $2N+1$. A similar argument stands for the reversed case.

We can use this sort of algebra for the remaining cases and show that all possibilities cannot distinguish these two languages. □

I have not shown whether or not this can be done with three states. In order to perform the above type of analysis one needs to consider the 27 different possible transitions on a three level system. At this point the solution becomes unnecessarily burdensome. It seems that the task can be reduced to, at best, counting mod 2 the occurrence of a two symbol pattern like HT or TH. This takes at least four states, two to recognize the pattern and two to count the instances mod 2. Of course, if one wishes to consider a probabilistic machine that distinguishes the languages with probability $\frac{1}{2}+\epsilon$, one should be able to find a two level system that can perform this task for any ϵ.

7.3 Classically Distinguishing L_I from L_{NOT}

I have developed an automaton which, I believe can distinguish L_I from L_{NOT} for the group (6.9). It is simply the group automaton for the finite group

$$G_5 = \langle a, b | a^2, b^8, ab = ba \rangle$$

This automaton is shown in Figure It is the Cayley graph for the group G. The maps defining this automaton are given in Appendix 1. I have not proven that for all words in L_I, this automaton always arrives at the state labelled I.

To prove that this automaton truly distinguishes these languages, we can either look at the group structure or the grammatical structure that generates the graph. Let us label the infinite group

$$G_6 = \langle a, b | a^2, b^8 \rangle$$

and thus the grammars that define all words in the equivalence class of the identity for each group are given by $\Gamma_{G_5}(I), \Gamma_{G_6}(I)$. The languages derived from these grammars are $L(\Gamma_{G_5}(I)), L(\Gamma_{G_6}(I))$. Note that $L(\Gamma_{G_6}(I))$ was previous called L_I, and $L(\Gamma_{G_6}(ab^4a))$ was called L_{NOT}.

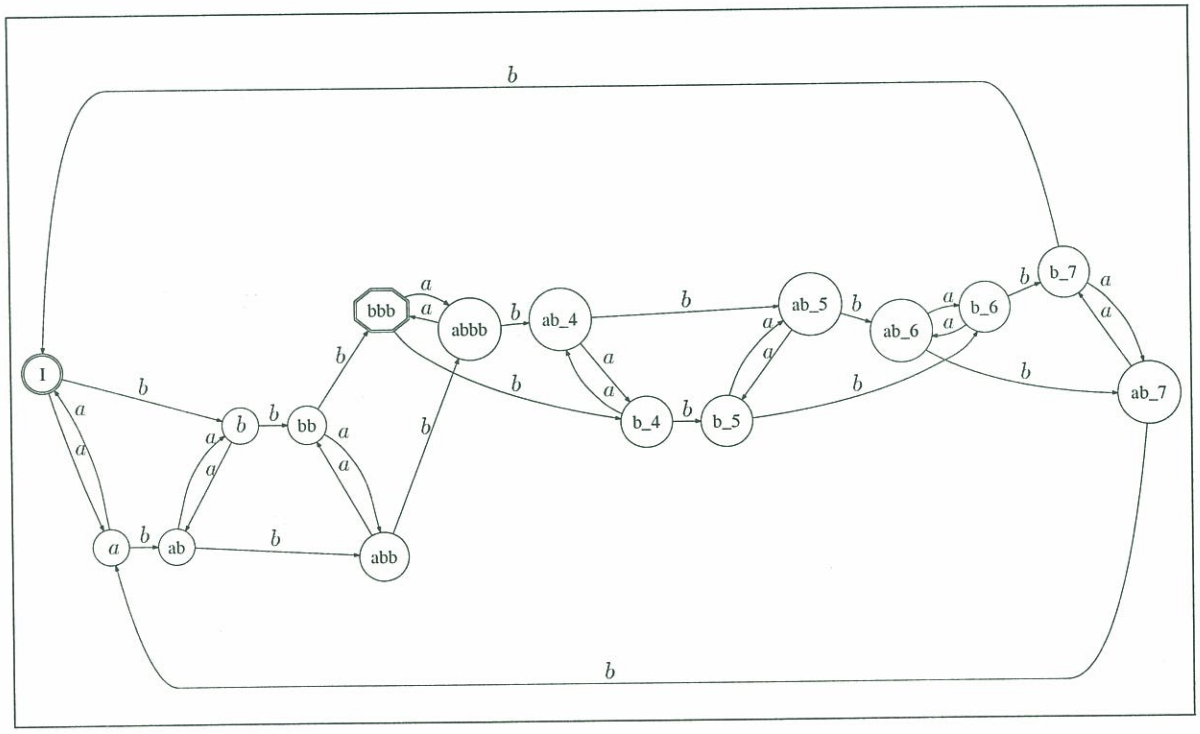

Figure 7.2: A classical automaton distinguishing L_I from L_{NOT}

We know that the automaton in Figure 7.2 accepts $L(\Gamma_{G_5}(I))$ and it also accepts the language $L(\Gamma_{G_5}(ab^4a))$ at a different state, thus distinguishing these languages. How do we generate the grammar $\Gamma_{G_6}(I)$?

$$\begin{array}{cccc} & \pi_{G_6}(I) & & \\ R & \longrightarrow & (Ra)^2R & \\ R & \longrightarrow & (Rb)^8R & \\ R & \longrightarrow & \epsilon & \end{array}$$

Thus the group is not commutative. If we add one production, the one inducing commutativity, we get $\Gamma_{G_5}(I)$.

$$\begin{array}{cccc} & \pi_{G_5}(I) & & \\ R & \longrightarrow & (Ra)^2R & \\ R & \longrightarrow & (Rb)^8R & \\ aRb & \longrightarrow & bRa & \\ bRa & \longrightarrow & aRb & \\ R & \longrightarrow & \epsilon & \end{array}$$

It should be clear that the set of words produced by $\Gamma_{G_5}(I)$ is greater than that produced by $\Gamma_{G_6}(I)$ because it includes all words formed from permutations of letters in each word generated by $\Gamma_{G_6}(I)$. Because of this, we can say that $L(\Gamma_{G_6}(I)) \subset L(\Gamma_{G_5}(I))$. Thus if an automaton accepts $L(\Gamma_{G_5}(I))$ at one state, say I as in Figure 7.2, then it accepts $\Gamma_{G_6}(I)$ at that state. It also follows that $L(\Gamma_{G_6}(ab^4a)) \subset L(\Gamma_{G_5}(ab^4a))$. Thus we see that, indeed, this automaton does distinguish L_I from L_{NOT}.

Looking at the group structure, we want to show that $[I]_{G_6} \subset [I]_{G_5}$ where G_6 is the non-commutative, infinite group. If we look at the set of all words in $[I]_{G_5}$ we can see that it is not diminished but is, instead simply added to due to the inclusion of the relater $ab = ba$.

At this point we can very easily confound this result by pointing out the fact that the qubit still can distinguish $[I]$ from the class $[ab^4ab^4]$ with some probability. However, the given automaton will certainly confuse it with the identity. Clearly a more general view needs to be taken and questions remain open.

7.4 Formal Grammars

Defining a grammar is similar to defining a language in that there is an alphabet, and a set of rules for defining the words. A grammar is thus given as $\Gamma = \{V, A, \pi, \sigma\}$. The finite set of symbols V, called the vocabulary, is composed both of what are called terminal symbols as well as non-terminal symbols. The best way to understand the difference between terminals and non-terminals is as follows. If at some point while building a word in Γ you find a non-terminal in the word, then the word is not yet complete. If you find only terminal symbols, then the word cannot be changed further and is thus complete. The nonempty subset $A \subset V$ is composed of only terminal symbols. π is a set of what are called productions which are identifications between words in V. π is a finite subset of $(V \setminus A)^+ \times V^*$, and $\sigma \in V \setminus A$ is the start symbol, where the words begin before any productions.

As an example, let R be a non-terminal symbol, and let a, b be terminal symbols. Thus, $V = \{R, a, b\}$, $A = \{a, b\}$, $\sigma = R$ and here is the set of productions

$$\begin{aligned} & \overset{\pi}{} \\ R &\longrightarrow aRb \\ R &\longrightarrow \epsilon \end{aligned} \tag{7.1}$$

ϵ is the empty string. We begin constructing words by first writing down σ and then whatever we find that is equal to that, we may substitute, so $\sigma \to R$. We can then do transformations based on what R is identified with. A typical chain of productions would work as

$$\sigma \overset{\sigma \to R}{\longrightarrow} R \overset{R \to aRb}{\longrightarrow} aRb \overset{R \to aRb}{\longrightarrow} aaRbb \overset{R \to aRb}{\longrightarrow} aaaRbbb \overset{R \to \epsilon}{\longrightarrow} aaabbb$$

It should be apparent that this grammar constructs the familiar language

$$L_{a^n b^n} = \{a^n b^n | n \in \mathbb{Z}, n > 0\}$$

from equation (6.24). The language derived from a grammar Γ is called $L(\Gamma)$.

7.4.1 Context Free Grammars

Let us look at the following production

$$bR \to bRa \tag{7.2}$$

We see that, in order for us to replace R with Ra we must find it beside the symbol b. We could not perform that production in the word $aaaRba$ as R does not follow b. However, we could transform the word $aabRa$ into $aabRaa$ because R is set in the proper context. If, in the set of all productions that define a grammar, we find none that are of this form, we call the grammar a context free grammar.

7.5 Groups as Grammars

It should be clear that the grammar defined in (7.1) is a subset of $[I]_G$ from the group given by the group structure (6.21). That group is given as

$$G = \langle a, b | ab \rangle$$

and it should be clear that this is a commutative group, in that a commutes with b. This means that if we simply insert a few extra relations into π, we can have a grammar that produces $[I]_G$. That grammar is as follows

$$\pi \tag{7.3}$$

$$R \rightarrow RaRbR \tag{7.4}$$

$$R \rightarrow RbRaR \tag{7.5}$$

$$R \rightarrow \epsilon \tag{7.6}$$

A typical production in this grammar is as follows

$$\sigma \overset{\sigma \rightarrow R}{\longrightarrow} R \overset{R \rightarrow RaRbR}{\longrightarrow} RaRbR \overset{R \rightarrow RbRaR}{\longrightarrow} RbRaRaRbR \overset{R \rightarrow RaRbR}{\longrightarrow} RbRaRaRbRaRb \overset{R \rightarrow \epsilon}{\longrightarrow} baabab$$

Let us check that this final word, $baabab$, is in $[I]_G$. Well, we know that a and b commute, so let us rearrange the letters grouping all of the a's and the b's

$$\text{baabab} = \text{aaabbb}$$

We know that $ab = I$ and thus, discarding ab pairs from the center we can reduce the word to the identity and the word is in $[I]_G$.

The relationship between grammars and finitely presented group is not superficial. The two are deeply related as we see in [18]. Instead of presenting a language in the form

$$L_{a^n b^n} = \{a^n b^n | n \in \mathbb{Z}, n > 0\}$$

we can also present it in the form of a grammar. Thus we may now upgrade our question of distinguishing languages to the one of distinguishing grammars. We have found that a very small deterministic automaton can distinguish the grammar (7.3) from a very similar one. That was shown in Section 7.1.1. To generate the grammar that produces the language with one extra a,

$$L_{a^{n+1}} = \{a^{n+1} b^n | n \in \mathbb{Z}, n > 0\}$$

we simply need to add a non-terminal and make it equivalent to the letter a. The grammar for $L_{a^{n+1}} = L(\Gamma_1)$ is given by $V = \{\text{Q,R,a,b }\}$, $A = \{a, b\}$, $\sigma = Q$ and here is the set of productions

$$\pi \tag{7.7}$$

$$Q \rightarrow RaR \tag{7.8}$$

$$R \rightarrow RaRbR \tag{7.9}$$

$$R \rightarrow RbRaR \tag{7.10}$$

$$R \rightarrow \epsilon \tag{7.11}$$

$$\tag{7.12}$$

All of the above grammars are context free grammars. Clearly, there are no productions π that are of the form $aQb \rightarrow c$ for $a, b, c \in A$ and $Q \in V \setminus A$, and thus there is no context for performing a production. The importance of this will be apparent when we look at Theorem 7.6.1.

7.6 Is The Classical Case Undecidable?

It was quite simple to envision a 1QFA that could distinguish $L(\Gamma)$ from $L(\Gamma_1)$. However, if we are to find a classical automaton that distinguishes those grammars we need to know that those languages have distinct minimal regular covers. We see in [14] that we are unable to decide if a language has a minimal regular cover if we are given the grammar that generates it.

Theorem 7.6.1. *(Domaratzki, Shallit and Yu* [14]*)*
The Following decision problems are unsolvable

- *Given a context-free grammar G, does L(G) have a minimal regular cover?*
- *Given a context free grammar G, does* $\Sigma^* - L(G)$ *have a minimal regular cover?*

7.7 Open Problems

It has now been shown that a single qubit can distinguish an entire class of regular languages, each subsequent element of which requires a larger classical 1DFA. It has also been shown that a single qubit can distinguish two non-regular languages. At this point it seems clear that this field is open to further investigation. What is the class of languages that can be distinguished by a 1QFA, which cannot be distinguished by a 1DFA? The class of languages that can be classically distinguished is currently unknown. It then suggests that finding the class of languages that can be distinguished by a 1QFA is much harder. Nevertheless, it should be a significant achievement to show that a 1QFA can distinguish languages that have been shown to be classically indistinguishable.

The next step is to consider the probabilistic finite automaton. We see in [22] that all words in some non-regular languages are accepted with bounded probability. It may be possible that a probabilistic automaton may also be able to reject all words in another non-regular language with bounded probability, and thus perform the same task as the quantum automaton.

Chapter 8

Conclusion

It has essentially been shown that classical automata that do the task of distinguishing languages must scale whereas the quantum automaton does not need to scale. In order to understand the implications and possible forward directions of this kind of research, it is instructive to see some of the shortcomings of available information on group theory and its intersection with automata and formal language theory.

The first obstacle encountered during my research was trying to gather the exact algebraic structure for the famous Hadamard and $\pi/8$ gates, H and T. It seems that there may not be any known way of finding this structure. It would be useful to have some means, given an arbitrary set of elements in $SU(n)$, to find any possible finite presentation for the group they generate. It was discovered that several finitely presented groups do have exact representations in $SU(2)$ and that work can be found in [25] and elsewhere.

Let us just imagine that there is some way of defining the class C of finitely presentable groups which have representations in $SU(n)$. This class would then produce a class of grammars C_Γ, and then, in turn, a class of languages $C_{L(\Gamma)}$. Each element of $C_{L(\Gamma)}$ can be recognized by a 1QFA with either bounded or unbounded error. The grammars in C_Γ are both context free and context sensitive. For example, the word problem for the free abelian group

$$G = \langle a, b | ab = ba \rangle$$

can only be generated by a context sensitive grammar due to the necessary inclusion of the production $aRb \longrightarrow bRa$. It is very easy to find a representation of this group in $SU(n)$ for

some n. Simply take any unitary U_a acting on one qubit such that $U_a^n \neq I, \forall n \in \mathbb{Z}$ and a unitary acting on another qubit U_b and having the same property. Thus, $C_{L(\Gamma)}$ contains part (all?) of the set of context sensitive languages and part of the set of context free languages. One may ask, can a deterministic push down automaton accept all of these languages with bounded or unbounded error? Of course, one may also ask, can the probabilistic automaton accept all languages in $C_{L(\Gamma)}$?

One decision problem exists which the 1QFA successfully answers and that is distinguishing elements in $C_{L(\Gamma)}$. It seems likely that there are others. It was shown that there exists a deterministic classical automaton which can distinguish any of the language pairs investigated here. However, while it was easy to find one for the NOT word, it seems like a very hard problem to find one for other words. Is it possible that there exists a decision problem over $C_{L(\Gamma)}$ which is classically undecidable, but may be decidable using a quantum automaton? Clearly that was the point of this research from the beginning. It would be a holy grail to find that a quantum automaton, using its continuum of internal states, could reverse some proof of decidability.

It can be said without hyperbole that the single greatest irony in physics, and perhaps in all of science lies in the fact that quantum systems may hold the only remaining continuum outside of the human mind. With the race on in quantum gravity to show that space time itself is not even a smooth manifold, but is instead discrete, we find that our notions of the continuum to be a potential hallucination. Could it be that our beloved metaphor, that of a smooth space, is just a dream induced by the inability to perceive the tiny pieces that make up the universe? It seems likely since it has already been proven that energy and matter come in only discrete packets, and now space-time itself is going to be discretized. The irony is that, while quantum mechanics is paving the way for this discretization, it remains that quantum systems live in another strange world where reality is made up of a continuum of what are called pure states. This continuum may be the last remaining continuum in the known universe. It is curious how we have been able to say so much, both in physics and mathematics, using our notions of the continuum. How did we ever happen upon this notion of the continuum? One may ask what the continuum is, and in fact we find many interesting viewpoints. We know that the considerations of the continuum, its cardinality and the cardinality of its infinite subsets formed the seed for, potentially, the most startling result in mathematics, namely the incompleteness theorem. It is my belief that the inferences we make, based on our collective agreement on the existence of

a thing called a continuum, is a strictly incomputable action. Our innate ability to handle the concept of infinity offers us so much in the way of creating new mathematics. The inferences we make using this ability, I believe, are also incomputable actions. Furthermore, when one considers the possibility that quantum systems have the only known continuous freedom, aside from the momenta of unbounded systems, one begins to wonder where this ability comes from. Where ever did we get the notion? What I am suggesting, of course, is that this may be one of the indicators of quantum actions in the human mind.

Another interesting connection between quantum systems and the human mind has been demonstrated strictly in this work. By showing that many finitely generated subgroups of SU(n) have finite group presentations, we can see that they also have a grammatical structure. These formal grammars are the same as those defined by the linguist Noam Chomsky. They are still used to understand natural languages. In fact, formal grammar theory forms the earliest passages in Stephen Pinker's book *Words and Rules*[26].

Hopefully this work may form the beginning of some very interesting directions in computation and even cognitive science.

Appendix A

Automaton Transition Table for Distinguishing L_I from L_{NOT}

$$
\begin{aligned}
I &\longrightarrow a[label = a] \\
I &\longrightarrow b[label = b] \\
a &\longrightarrow I[label = a] \\
a &\longrightarrow ab[label = b] \\
b &\longrightarrow ab[label = a] \\
ab &\longrightarrow b[label = a] \\
ab &\longrightarrow abb[label = b] \\
abb &\longrightarrow bb[label = a] \\
b &\longrightarrow bb[label = b] \\
bb &\longrightarrow abb[label = a]
\end{aligned}
$$

$$
\begin{aligned}
bb &\longrightarrow bbb[label = b] \\
abb &\longrightarrow abbb[label = b] \\
abbb &\longrightarrow bbb[label = a] \\
bbb &\longrightarrow abbb[label = a] \\
abbb &\longrightarrow ab_4[label = b] \\
bbb &\longrightarrow b_4[label = b] \\
ab_4 &\longrightarrow b_4[label = a] \\
b_4 &\longrightarrow ab_4[label = a] \\
ab_4 &\longrightarrow ab_5[label = b] \\
b_4 &\longrightarrow b_5[label = b] \\
ab_5 &\longrightarrow b_5[label = a] \\
b_5 &\longrightarrow ab_5[label = a] \\
ab_5 &\longrightarrow ab_6[label = b] \\
b_5 &\longrightarrow b_6[label = b] \\
ab_6 &\longrightarrow b_6[label = a] \\
b_6 &\longrightarrow ab_6[label = a] \\
ab_6 &\longrightarrow ab_7[label = b] \\
b_6 &\longrightarrow b_7[label = b] \\
ab_7 &\longrightarrow b_7[label = a] \\
b_7 &\longrightarrow ab_7[label = a] \\
ab_7 &\longrightarrow a[label = b] \\
b_7 &\longrightarrow I[label = b]
\end{aligned}
$$

Bibliography

[1] L. E. F. Galvão and L. Hardy, Phys. Rev. Lett. 90, 087902 (2003).

[2] L. Hardy, e-print http://arxiv.org/pdf/quant-ph/0101012 (2001).

[3] L. Vaidman and Z. Mitrani, Phys. Rev. Lett. 92, 217902 (2004).

[4] L. Vaidman and A. Kalev, e-print quant-ph/0406024 (2004)

[5] S. Aaronson and D. Gottesman, e-print arXiv:quant-ph/0406196 (2004).

[6] M.A. Nielsen and I.L. Chuang, *Quantum computation and quantum information*. Cambridge University Press, Cambridge U.K. 2000.

[7] A. Kondacs and J. Watrous, *On the power of quantum finite state automata* In Proceedings of the 38th Annual Symposium on Foundations of Computer Science, pages 66-75, 1997.

[8] A. Brodsky and N. Pippenger, SIAM Journal on Computing, 31(5), 1456-1478, 2002.

[9] A. Brodsky, *Models and Characterizations of 1-Way Quantum Finite Automata*. MSc. Thesis UBC (1998).

[10] J. Gruska, *Quantum Computing*. McGraw-Hill, 1999.

[11] D. Mowbray, *Coefficient Based Simulations of Quantum Algorithms*. MSc. Thesis University of Waterloo (2003).

[12] J.E. Hopcroft, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass. 1979.

[13] J.M. Howie, *Automata and Languages*. Clarendon Press, Oxford ; Oxford University Press, New York, (1991).

[14] M. Domaratzki, J. Shallit and S. Yu, *Minimal Covers of Formal Languages*. Lecture Notes in Computer Science Publisher vol. 2295.Springer-Verlag Heidelberg (2002)

[15] S. Rees, *How hard is the word problem?*. in Proceedings of Conference for European Women in Mathematics, Varna, Bulgaria, September (2002).

[16] J. M. Lee, *Introduction to Topological Manifolds*. Springer, New York 2000.

[17] Anisimov A.V. Anisimov and F.D. Seifert, *Zur algebraischen charateristik der durch kontextfreie Sprachen definierten Gruppen* Elektron. Informationsverarb. Kybernet. 11 695-702 (1975).

[18] D. Muller and P. Schup, Journal of Computer and System Sciences 26, 295-310 (1983).

[19] C. Fuchs, e-print quant-ph/0205039 (2002).

[20] N. Weaver, Journal of Mathematical Physics, Volume 41, Number 1 January (2000).

[21] S.Lloyd, Phys. Rev. Lett. 75, 346 (1995).

[22] M. Rabin, Information and Control 6, 230-245 (1963).

[23] A. Kikusts, e-print arXiv:quant-ph/9810065 (1998).

[24] A. Ambainis and R. Frievalds, e-print arXiv:quant-ph/9802062 (1998).

[25] J. Z. Gonçalves, A. Mandel, and M Shirvani J. Algebra 214no. 1, 301-316 (1999).

[26] S. Pinker, *Words and Rules: the ingredients of language*. New York : Basic Books, 1999.