THE UNIVERSITY OF CALGARY

Non-Determinism and Quantum Information

by

Chris Marriott

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

November, 2003

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Non-Determinism and Quantum Information" submitted by Chris Marriott in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

[illegible]

Chairman, Dr. John Watrous
Department of Computer Science

[illegible]

Dr. Peter Høyer
Department of Computer Science

[illegible]

Dr. Hugh Cowie Williams
Department of Mathematics and Statistics

January 22, 2004
Date

Abstract

In this thesis we study the union of two tools of computation: quantum information and non-determinism. We begin by showing that the class of languages QMA, which characterizes bounded error non-deterministic quantum polynomial time, is robust in a stronger sense, and we present some applications of this result. Second we will show how the quantifier characterization of non-determinism can be extended into the quantum realm, by treating both classical and quantum non-determinism. To make this extension we will shift our focus to promise problems instead of the standard decision problems. In order to characterize quantum non-determinism we will introduce the concept of quantum sets, an analogue of classical languages. Finally we will show new oracle separations regarding the quantum non-deterministic classes that we consider.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The science of computing is necessarily a study of models of computation, whether the model sits on your desk, or has an infinite quantum tape. We study what these models can do and we try to figure out if there is anything they can not do in an attempt to determine what makes the best model. The discovery of the halting problem and the set of computable languages led to the Church-Turing thesis which states that, at least by this measure, all models were created equal. However there exist vastly more measures that can be applied to determine the best model, and complexity theory is rich with subclasses of the computable languages, each defined by its own specific model of computation. We consider models that are limited by space or time, or number of gates, or depth of circuit. We consider models that have access to infinite look up tables, and all knowing confessors. Each of these models selects a specific class of languages and each of these classes illustrate the power of the computational models that define it. Models that select a larger subset of the computable languages can be considered more powerful than those that select smaller sets, and we can identify different hierarchies of models of ever increasing power.

Sometimes we discover that two different models of computation select the same set of languages. We find that concepts like "non-determinism" and "classical randomness" can be defined in numerous ways and still select the same class of languages. Discovering that there are numerous ways to define a particular class is useful not only for using the class in future research, but also grants a legitimacy to the model being studied. For instance, the fact that non-deterministic classes generally have a number of definitions suggests that non-determinism is a unique property of models of computation that deserves understanding. Also by having an array of identified definitions for an abstract tool of computation we can speak about that tool without reference to a specific model which allows us to compare these abstract tools at a higher level than on the specific class to class basis. Thus we can compare the power of randomness to the power of non-determinism and form judgments like "Unbounded classical randomness is at least as powerful as error free non-determinism."

Quantum information added another layer to the classes that already existed. Almost every classical class has a potential quantum analogue, so long as the model used to define that class has a quantum variant. Thanks in part to excitement generated by Shor's polynomial time algorithm for factoring integers, these classes have started being studied in earnest. Quantum is another abstract tool of computation, characterized primarily by quantum circuits, but also characterized by quantum Turing machines. The majority of complexity classes are defined using the Turing machine as the base model of computation, subject to various restrictions, and potentially with added functionality. By changing the base model to quantum Turing machines, but leaving all other concerns unchanged, we create a new level of quantum complexity classes, that is not known to collapse to the classical level. In the remainder of this Chapter we define a number of primary classes that are defined by the Turing machine and its variants, under restrictions of polynomial time computation. We consider the tools of non-determinism as well as bounded error and exact computation in combination and in comparison to the quantum models. These tools combine to create five classical complexity classes and their five quantum analogues.

We begin our study with the quantum class QMA, which characterizes bounded-error non-deterministic quantum polynomial time. Bounded error classes commonly have a property known as robustness that means the specific bound on the error is flexible within a range. It is already known that QMA is a robust bounded error class, but the current proof of this fact makes a small concession that links the length of the proof to the probability of error. In Chapter 2 we will present a new proof of robustness that does not rely on this concession.

Our study continues with an exploration of how the quantifier characterization of non-determinism can be expanded for use with the quantum models of computation. While it is not surprising that this characterization can be applied in the quantum world, it is interesting to notice that the introduction of quantum information makes the application non-trivial, and necessitates a minor change of perspective due to the continuous nature of quantum information which characterizes one of the primary differences between quantum and classical models. This change of perspective highlights the idea of promise problems, or rather problems defined on a restricted domain. We introduce and study promise problems in Chapter 3 also in which we introduce the concept of quantum languages, or sets of quantum superpositions, and we find that it is again natural to consider these quantum languages as promise problems to make up for the continuous nature of quantum information.

We conclude with a look at the non-deterministic quantum classes in the relativized world. Relativized separations fill a unique void in complexity theory, where unrelativized separations are the Golden Fleece of every theorist. Though the field is undecided on the benefit of relativized separations, it has become a common practice to consider the relativized relationships of any new classes. We will add to the list new relativized separations concerning the non-deterministic quantum classes that we study in Chapter 4.

In the final Chapter we will consider what relationships our explorations have uncovered and what these discoveries mean for the bigger picture. We will compare the tools of quantum information, classical randomness, and non-determinism on an abstract level in light of the current state of the art.

1.1 Complexity Theory

The primary topic of our project is complexity theory, and for a more thorough review than we will present please see [5] and [6]. To begin we will look at the paradigmatic classical complexity classes, P and NP, and the equally critical probabilistic class, BPP. These classes form the backbone of complexity theory and have been studied from many angles since they were first defined. The Turing machine, in its various forms, is the primary model of computation used to define these classes. A Turing machine is a finite state machine equipped with an infinite tape upon which the machine may compute by reading and writing information. It was shown by Turing that this machine characterizes the computable languages. To define complexity classes we restrict the resources of the machine, in our case we will look at machines restricted to polynomial time computation, but we will also consider restrictions placed on the acceptable error of the machine as well as expand the power of the

Turing machine to take advantage of other tools of computation.

Below we state common definitions for these classes using the standard alphabet $\Sigma = \{0,1\}$. For the classes we look at the definitions will contain a deciding machine of a specific type (in this project we will look at Turing machine variants and circuit variants), as well as two propositions that describe the accepting and rejecting conditions of the decider. We will borrow the terms "completeness" and "soundness" to refer to the accepting and rejection conditions respectively. Furthermore, if M is a decider of a particular type, then we write $M(x)$ to mean the result of the decider M run on input x. This value will always be 0 or 1, but in some cases it may be a probabilistic or even quantum combination of the two, in which cases the bias of the mixture becomes important.

Definition 1 *A language, $L \subseteq \Sigma^*$, is in the class* P *(deterministic polynomial time) if and only if there exists a polynomial time deterministic Turing machine, M, for which*

1. *(Completeness)* $x \in L \Rightarrow M(x) = 1$
2. *(Soundness)* $x \notin L \Rightarrow M(x) = 0$

The class NP makes use of the concept of non-determinism. This concept has been studied in great depth and serves to identify a property in common among most of the classes we will discuss. There are many characterizations of this property. One characterization proposes the non-deterministic creation of a string, called a proof, witness, or certificate, prior to computation that is then used by a deterministic Turing machine to decide acceptance or rejection. In this characterization we think of the non-deterministic function as interference by an all knowing being, whose primary motivation is to cause the deterministic Turing machine to accept. Thus the being will always be able to find a proof of acceptance if one exists, but we design the Turing machine to never be tricked should the input really be rejected.

Another characterization depicts the computation of a non-deterministic Turing machine as a computation tree, where at each node the Turing machine can choose non-deterministically to follow either the right or left branch out of the node. In this case we say that an input is accepted if there is any path in the tree that leads to an accepting configuration. It is not difficult to show that these characterizations are equivalent, that is that they select the same class of languages, and we shall more often consider the first when discussing non-determinism in later sections. We can define the class of languages NP below using the idea of a non-deterministic Turing machine.

Definition 2 *A language, $L \subseteq \Sigma^*$, is in the class* NP *(non-deterministic polynomial time) if and only if there exists a polynomial time non-deterministic Turing machine, M, for which:*

1. *(Completeness)* $x \in L \Rightarrow \exists_y : M(x, y) = 1$
2. *(Soundness)* $x \notin L \Rightarrow \forall_y : M(x, y) = 0$

Here we call y the proof *or* certificate *and is assumed to be polynomial in the length of x.*

The probabilistic Turing machine is sometimes compared to the non-deterministic Turing machine because the models have identifiable similarities. Consider the non-deterministic computation tree described earlier. The choice of which branch to take is made by some all knowing power, but an obvious modification, given that such an all knowing power is generally hard to find, is to randomly choose a branch by flipping a coin. This characterizes a probabilistic Turing machine, one that combines the deterministic Turing machine with the power to make a truly random and unbiased choice. We find that this power is of no use if we do not allow the machine to make some error so often the ideas of randomness and (bounded or unbounded) error prone computation are considered to be inseparable. Sometimes we allow the error to be unbounded, that is we will say that the machine accepts if even the slightest majority of computation paths are accepting. We find that in practice this type of model makes too many errors and so primarily we study models that err only with some acceptably bounded probability. We can characterize this model by restricting the error to below a constant strictly less than $\frac{1}{2}$, but we find that this model can allow error even polynomially close to $\frac{1}{2}$ without altering the class of languages that it selects. This property is called robustness and is common in most bounded error classes. Unfortunately we find that classically there is no source of true randomness but due to the successful use of pseudo-random number generators and the extensive research into derandomization most people consider BPP, defined below, to characterize the efficiently computable problems.

Definition 3 *A language, $L \subseteq \Sigma^*$, is in the class* BPP *(bounded error probabilistic polynomial time) if and only if there exists a polynomial time probabilistic Turing machine, M, for which:*

1. *(Completeness)* $x \in L \Rightarrow Pr[M(x) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \notin L \Rightarrow Pr[M(x) = 1] \leq \frac{1}{3}$

Because the choice of constant is arbitrary, commonly BPP is defined using the error bound of $\frac{1}{3}$. The study of probabilistic computation leads to another open problem, second in fame likely only to the P = NP question. That is, is P equal to BPP? There are many theorists attempting to "derandomize" BBP or rather show that P = BPP, and it is more widely believed that a solution to this problem is in the foreseeable future. We note that there is also a much larger complexity class associated with randomness that is characterised by unbounded probabilistic Turing machines. We state the definition below.

Definition 4 *A language, $L \subseteq \Sigma^*$, is in the class* PP *(unbounded error probabilistic polynomial time) if and only if there exists a polynomial time probabilistic Turing machine, M, for which:*

1. (Completeness) $x \in L \Rightarrow Pr[M(x) = 1] \geq \frac{1}{2}$

2. (Soundness) $x \notin L \Rightarrow Pr[M(x) = 1] < \frac{1}{2}$

As the study of these seminal classes became widespread, interest was garnered as to the combination of the tools of randomness and non-determinism. As above, this meant we had to allow some error into our computations so that we could take advantage of randomness. Babai [3] first introduced these classes as characterized by a game between two players, the all knowing wizard Merlin, and the young King Arthur, who, though intelligent himself, is limited by bounded error polynomial time computation. The game acts much in the same way as our non-determinism was described above. The games that Arthur and Merlin play consist of turns. In Merlin's turn Merlin is allowed to send a message to Arthur, limited only in that it must be polynomial in the length of the input string. In Arthur's turn Arthur is allowed to flip coins and compute deterministically (so long as he takes only polynomial time) and provided there is another turn for Merlin, he sends the results of any coin flips to Merlin. The goal of the game for Merlin is to convince Arthur to accept the input string regardless of whether it is a member of the language. The goal for Arthur is to answer correctly, that is he must catch Merlin if Merlin is trying to convince him to accept if the input string should be rejected. We say that a language has a Merlin-Arthur protocol if for every input in the language Merlin has a sequence of moves in the protocol that convinces Arthur to accept with bounded error probability, and for every input not in the language, no matter Merlin's moves Arthur will reject with bounded error. The length of the games and who gets to take their turn first determines a whole hierarchy of complexity classes. We find though that in the classical setting this hierarchy collapses to two turn games. We note that the class M (or rather the class where only Merlin gets a turn and Arthur is not allowed to flip coins) is another characterization of NP, and that A (no Merlin turn, so only Arthur and his coins) is another characterization of BPP. Below we define the two classes characterized by two message games.

Definition 5 *A language, $L \subseteq \Sigma^*$, is in the class* MA *(Merlin-Arthur) if and only if there exists a polynomial time Merlin-Arthur game, M, for which:*

1. (Completeness) $x \in L \Rightarrow \exists_y : Pr[M(x,y) = 1] \geq \frac{2}{3}$

2. (Soundness) $x \notin L \Rightarrow \forall_y : Pr[M(x,y) = 1] \leq \frac{1}{3}$

Here we call y the proof *or* certificate *and is assumed to be polynomial in the length of x.*

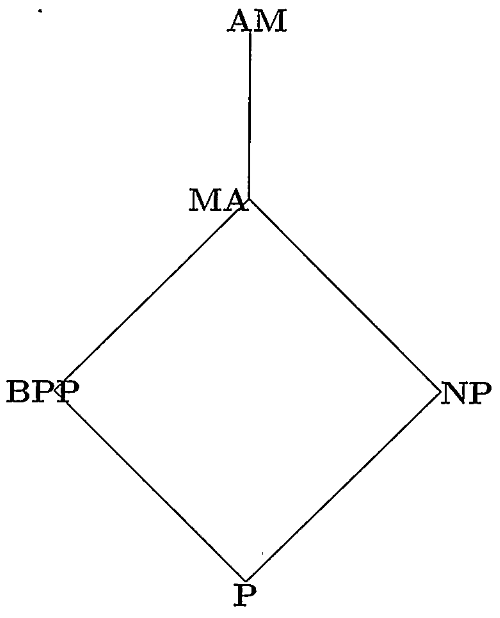

Figure 1.1: The classical class structure.

At its core is the same standard model of computation, the Turing machine, but the machine is aided by both non-determinism (Merlin) and classical bounded error randomness (Arthur). In a way we can think of this as adding non-determinism to an already probabilistic Turing machine. This analogy will become much more obvious when we begin discussing quantifiers. The other possibility is adding classical randomness to an already non-deterministic Turing machine. This case we find is characterized by the other type of two round Merlin-Arthur games in which Arthur is allowed to play first. In the definition below the probability is taken over the choice of string y.

Definition 6 *A language, $L \subseteq \Sigma^*$, is in the class* AM *(Arthur-Merlin) if and only if there exists a polynomial time Arthur-Merlin game, M, for which:*

1. *(Completeness)* $x \in L \Rightarrow Pr[\exists_y : M(x,y) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \notin L \Rightarrow Pr[\exists_y : M(x,y) = 1] \leq \frac{1}{3}$

Here we call y the proof *or* certificate *and is assumed to be polynomial in the length of x.*

Though it is not immediately obvious, MA is contained in AM. One way to show this relies on the fact that all 3 round games or longer select the same languages as AM. Thus the class AMA = AM and we can also note that MA $\subseteq$ AMA. It is not known if AM collapses to MA, but it is widely believed that it does not. MA is also known to be contained in PP, whereas AM is only known to be contained in PSPACE. Both are also contained in the second level of the polynomial hierarchy. The first figure presents the relationships among these classical complexity classes. In the figure the lines represent containment, with the lower class contained in the higher class. The relationship between BPP and NP remains unknown.

1.2 Quantum Computing

The area of quantum complexity has grown on the apparent advantage quantum information has in the polynomial time solution to the integer factoring problem [17]. For an indepth look at the field please see [15]. Many quantum classes have been defined and studied to accommodate the new field and now there is quite a family of quantum complexity classes, some of which are defined below. Of interest to this project are the classes that are commonly thought of as quantum analogues of the five classical classes discussed above. One of the first quantum classes introduced and studied was BQP, or bounded error quantum polynomial time [9]. This class contains the quantum factoring problem and is the quantum analogue of BPP.

To define quantum classes we must make reference to a method of manipulating quantum information. One such model that was studied early in quantum complexity [9] was the quantum Turing machine. This model is not easily programmed since quantum superpositions are far from intuitive. As a result the quantum Turing machine becomes a tool only of the complexity theorist while most who are concerned with actually programming with quantum information prefer another model: the quantum circuit.

In the circuit model we are concerned with the number of universal gates that are in the circuit as comparable to the time measure on Turing machine. We will use a common universal set that includes the Toffoli gate, the Hadamard gate, and an i-shift gate. Furthermore, since a particular circuit has a fixed size it will only work on a given input size, so we consider *circuit families* or sets of circuits, one for each input length, that are generated by a uniform function, required to be polynomial time computable.

Definition 7 *A set of circuits* $Q = \{Q_1, Q_2, ...\}$ *is a* uniformly generated polynomial sized circuit family *if*

1. *Q_i is a circuit that takes input of length i*
2. *There exists a polynomial time computable function f such that $f(1^n) = \langle Q_n \rangle$ where $\langle Q_n \rangle$ is an encoding of the circuit Q_n.*

We restrict the function to polynomial time computable so that it does not have enough time to solve the language itself (unless the language is polynomial time computable already) and return the trivial solution: the circuit that just accepts or rejects. Further, the polynomial time restriction will ensure that the circuit has polynomial size, since it does not have the time to write down a larger circuit. We call these families "uniformly generated polynomial sized circuit families", but sometimes the "uniformly generated" is dropped since we will not consider families that are not uniformly generated. We use circuit families to define the classes below but we note that the quantum Turing machine can also be used to characterize these quantum classes. For a circuit family Q we will write $Q(x)$ to mean $Q_n(x)$ for x of length n.

Definition 8 *A language, $L \subseteq \Sigma^*$, is in the class* BQP *(bounded error quantum polynomial time) if and only if there exists a uniformly generated polynomial sized quantum circuit family, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow Pr[Q(x) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \notin L \Rightarrow Pr[Q(x) = 1] \leq \frac{1}{3}$

In the definition of BQP, as in definitions to follow, bounded error is used in the same way that it is used in the class BPP, that is, bounded away from $\frac{1}{2}$ by a constant. As was natural in the classical case, but somewhat unnatural in the quantum case where the probability, and error, seem built in, we can consider circuits with no error. The natural quantum analogue of P is EQP, sometimes called QP, defined below.

Definition 9 *A language, $L \subseteq \Sigma^*$, is in the class* EQP *(exact quantum polynomial time) if and only if there exists a uniformly generated polynomial sized quantum circuit family, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow Q(x) = 1$

2. *(Soundness)* $x \notin L \Rightarrow Q(x) = 0$

It was not long before non-determinism was also studied in the world of quantum information. The natural quantum non-deterministic class is QMA or the quantum analogue of MA. This class has seen much study as well [18, 2]. The model underlying the definition of this class is the same protocol of interaction between the prover, Merlin, and the verifier, Arthur, who is now limited to bounded error polynomial sized quantum circuits. The interaction in this class is only one message, passed from Merlin to Arthur, and can be any quantum state limited only in that it must be polynomial in length of the input size. We usually call a protocol of this nature a quantum verification procedure, sometimes indicating the number of messages, and we measure its size in the number of gates in the verifier's circuit.

A *quantum verification procedure* Q is a uniformly generated quantum circuit family $\{Q_n : n > 0\}$, where each circuit Q_n acts on $n + k(n) + m(n)$ qubits for $k(n) \geq 0$, where n is the input size and m and k are polynomials. The function m specifies the length of the quantum certificate, while the function k specifies the number of work qubits used by the circuit. When the input x has been fixed or is understood, we will write m and k to mean $m(n)$ and $k(n)$, respectively, in order to simplify our notation. When we want to emphasize the length of the quantum certificate, we will refer to Q as an *m-qubit quantum verification procedure.*

Suppose we are given a string $x \in \Sigma^*$ and a quantum state $|\psi\rangle$ on $m(n)$ qubits, and we perform the following process:

1. Run the circuit Q_n on the state $|x\rangle|\psi\rangle|0^k\rangle$.

2. Measure the last qubit of the resulting state in the $\{|0\rangle, |1\rangle\}$ basis, interpreting

the outcome of the measurement as *accept* or *reject*, accordingly.

We use the quantum verification procedure to characterise a class of languages analagous to the classical Merlin-Arthur games. In the definition below we will write $Q(x, |\psi\rangle)$ to express the outcome of the appropriately sized member of the quantum circuit family Q run on the input x and $|\psi\rangle$ (0 if it rejects 1 if it accepts).

Definition 10 *A language, $L \subseteq \Sigma^*$, is in the class* QMA *(quantum Merlin-Arthur) if and only if there exists a uniformly generated polynomial sized one message quantum verification procedure, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow \exists_{|\psi\rangle} : Pr[Q(x, |\psi\rangle) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \notin L \Rightarrow \forall_{|\psi\rangle} : Pr[Q(x, |\psi\rangle) = 1] \leq \frac{1}{3}$

Here we call $|\psi\rangle$ the proof *or* certificate *and is assumed to be polynomial in the length of x.*

Also studied is the quantum class QAM which characterizes the class with two messages, first from Arthur to Merlin, then back from Merlin to Arthur. Arthur is still limited by polynomial sized quantum circuits to decide his answer based only on the communication and the input. We note that in the definition of QAM that we consider, Arthur is only allowed to compute classically until the last turn. Thus any messages sent by Arthur will consist only of classical coin flips as in the classical class AM.

Definition 11 *A language, $L \subseteq \Sigma^*$, is in the class* QAM *(quantum Arthur-Merlin) if and only if there exists a uniformly generated polynomial sized two message quantum verification procedure, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow Pr[\exists_{|\psi\rangle} : Q(x, |\psi\rangle) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \notin L \Rightarrow Pr[\exists_{|\psi\rangle} : Q(x, |\psi\rangle) = 1] \leq \frac{1}{3}$

Here we call $|\psi\rangle$ *the* proof *or* certificate *and is assumed to be polynomial in the length of* x*.*

Notice here that the source of the probability is no longer only classical coin flips made by Arthur prior to Merlin's turn but also his measurements afterward, in the final step. Thus the error bound is a bound on the sum of acceptance probabilities over all possible classical coin flips where the maximally accepting proof is assumed. As would be expected, QMA is contained in QAM. QMA is also known to be contained in PP, by an unpublished result of Watrous and Kitaev. QAM however is only known to be contained in PSPACE. We do note, though it is beyond the scope of this thesis, that Watrous has shown that QMAM is equal to QIP and thus contains PSPACE.

We can also consider the exact setting with respect to non-deterministic quantum computing. The protocol for quantum non-determinism is the same as above, though no error is permitted for the class QNP.

Definition 12 *A language,* $L \subseteq \Sigma^*$*, is in the class* QNP *(quantum non-deterministic polynomial time) if and only if there exists a uniformly generated polynomial sized one message quantum verification procedure, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow \exists_{|\psi\rangle} : Q(x, |\psi\rangle) = 1$
2. *(Soundness)* $x \notin L \Rightarrow \forall_{|\psi\rangle} : Q(x, |\psi\rangle) = 0$

Here we call $|\psi\rangle$ *the* proof *or* certificate *and is assumed to be polynomial in the length of* x*.*

The non-determinism in these three classes can be considered quantum non-determinism, since the proof can be quantum information. We can consider quantum classes where the non-deterministic parts are limited to classical information only. This restriction leads to another trio of classes that are quite similar to those just discussed.

Definition 13 *A languages, $L \subseteq \Sigma^*$, is in the class* QCMA *(quantum Merlin-Arthur with classical non-determinism) if and only if there exists a uniformly generated polynomial sized one classical message quantum verification procedure, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow \exists_y : Pr[Q(x,y) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \notin L \Rightarrow \forall_y : Pr[Q(x,y) = 1] \leq \frac{1}{3}$

Here we call y the proof *or* certificate *and is assumed to be polynomial in the length of x.*

Definition 14 *A language, $L \subseteq \Sigma^*$, is in the class* QCAM *(quantum Arthur-Merlin with classical non-determinism) if and only if there exists a uniformly generated polynomial sized two classical message quantum verification procedure, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow Pr[\exists_y : Q(x,y) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \notin L \Rightarrow Pr[\exists_y : Q(x,y) = 1] \leq \frac{1}{3}$

Here we call y the proof *or* certificate *and is assumed to be polynomial in the length of x.*

Definition 15 *A language, $L \subseteq \Sigma^*$, is in the class* QCNP *(classically non-deterministic quantum polynomial time) if and only if there exists a uniformly generated polynomial sized classical one message quantum verification procedure, Q, for which:*

1. (Completeness) $x \in L \Rightarrow \exists_y : Q(x,y) = 1$

2. (Soundness) $x \notin L \Rightarrow \forall_y : Q(x,y) = 0$

Here we call y the proof *or* certificate *and is assumed to be polynomial in the length of x.*

We have introduced the quantum classes to help us study the power of quantum information. The quantum classes we have looked at already are all thought of as natural analogues to the classical classes because the quantum element is added to the characteristic computation model of the particular class. As would be expected the quantum classes have a predictable structure with respect to each other, and the quantum world contains the classical in a strong sense. Below we state these relationships.

Proposition 1 *The following containments hold:*

1. $\mathrm{P} \subseteq \mathrm{EQP}$

2. $\mathrm{BPP} \subseteq \mathrm{BQP}$

3. $\mathrm{NP} \subseteq \mathrm{QCNP} \subseteq \mathrm{QNP}$

4. $\mathrm{MA} \subseteq \mathrm{QCMA} \subseteq \mathrm{QMA}$

5. $\mathrm{AM} \subseteq \mathrm{QCAM} \subseteq \mathrm{QAM}$

This proposition is quite easy to prove. The quantum models of computation can easily simulate classical algorithms by not using the quantum elements, so all the classical classes are contained within the quantum analogue. For the second pair of containments we can see that for the same reason classical proofs can be simulated

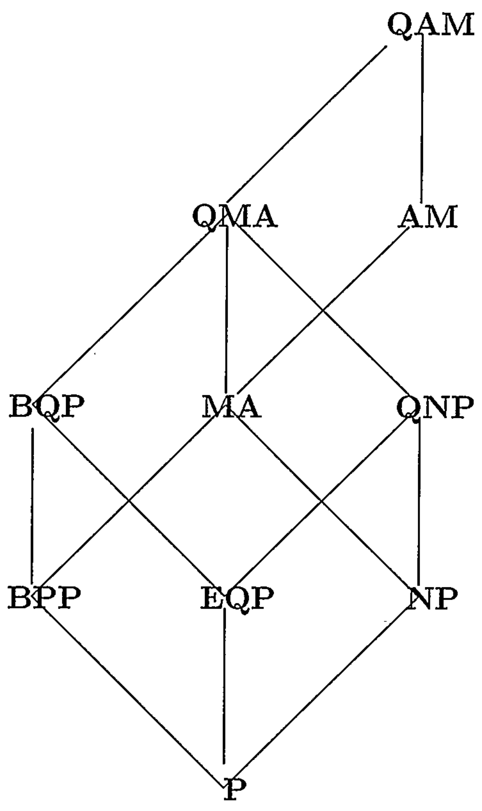

Figure 1.2: The quantum and classical class structure.

with quantum proofs, and thus classical message non-determinism is contained within quantum message non-determinism.

Furthermore relations can be established among the quantum classes in analogous ways to their classical parallels. For instance the relations between the functions of non-determinism and determinism remain constant when moving to the quantum model.

Proposition 2 *The following relationships hold:*

1. EQP $\subseteq$ QNP

2. $\mathrm{EQP} \subseteq \mathrm{BQP}$

3. $\mathrm{EQP} = co - \mathrm{EQP}$

4. $\mathrm{BQP} = co - \mathrm{BQP}$

5. $\mathrm{BQP} \subseteq \mathrm{QMA} \cap co - \mathrm{QMA}$

6. $\mathrm{QNP} \subseteq \mathrm{QMA}$

Again these relationships are not difficult to prove, since the proofs for the classical relationships will work with minor modification. The second figure shows all these relationships, though we leave out the QC-classes to avoid unneccesary clutter.

Chapter 2

Strong Robustness for QMA

The complexity class QMA characterizes the languages that can be decided in polynomial time by bounded error quantum verification procedures. These procedures have a simple structure that was introduced in the last Chapter. The prover, Merlin, and the verifier, Arthur, begin with access to the input string. The prover then prepares a message for the verifier that may be any quantum state of polynomial size. The verifier is limited to using polynomial sized quantum circuits to act on the input string and the message when deciding whether to accept or reject the input. Another restriction is placed on the verifier. Accepting inputs must be accepted with probability at least $\frac{2}{3}$ and rejecting inputs must be accepted with probability less than $\frac{1}{3}$.

Kitaev first demonstrated that QMA is a robust class [12]. This means that choice of constants above is arbitrary. That QMA is robust is not surprising since robustness is a property of most bounded error classes, including QMA's classical namesake MA. The procedure used to prove the robustness by Kitaev was a simple adaptation of the proof for MA, and used the standard amplification procedure of repetition and majority vote. The quantum nature of the class forced a small concession for this adaptation to work. Since quantum information cannot be copied and because the verifier's computation is potentially destructive to the message prepared by the prover, there is no guarantee that the proof can be used again after the first run. To overcome this the verifier must ask for multiple copies of the same proof to repeat the computation the necessary number of times. Since the size of the proof only increases by a polynomial to achieve exponentially small error, the proof of classical robustness adapts.

Though the concession was small it limits the use of this fact to cases where the cost of increasing proof length can be ignored. An example where this is not the case is with respect to the QMA subclass characterized by procedures limited to logarithmic length proofs. In the classical setting it is known that logarithmic length proofs are no more powerful than no proof at all. The proof of this fact relies on the robustness of the class MA so that a procedure is guaranteed to exist with insignificant error. In the quantum extension of this result, using the amplification procedure above, the length of the proof will expand to polylogarithmic lengths, which ruins the attempt to merely adapt the classical proof. In other cases, using the amplification procedure from above may require additional complexity in any demonstrations that rely on it, since the researcher must account for the increasing proof size.

In the next section we present an amplification procedure for quantum verification procedures that reuses the quantum information provided by the prover to achieve exponentially small error. The amplification procedure uses a technique that tries to repair any damage caused to the initial message during observations in the verification step. Though a recreation of the proof is only a possibility, it turns out that even when the proof is not recreated there remains enough information in the quantum state that further trials can be run. In the second section we apply our new robustness technique to the problem of logarithmic length proofs mentioned above.

2.1 Amplification

Now we state a more specific definition of the quantum class QMA that emphasizes the variability of both the probability of error and the length of the proof.

Definition 16 *A language, $L \subseteq \Sigma^*$, is in the class* $\text{QMA}_m(a,b)$ *(quantum Merlin-Arthur) if and only if there exists a uniformly generated polynomial sized one message quantum verification procedure, Q, for which:*

1. *(Completeness)* $x \in L \Rightarrow \exists_{|\psi\rangle} : Pr[Q(x, |\psi\rangle) = 1] \geq a$
2. *(Soundness)* $x \notin L \Rightarrow \forall_{|\psi\rangle} : Pr[Q(x, |\psi\rangle) = 1] \leq b$

Here we call $|\psi\rangle$ the proof *or* certificate *and it has length m.*

It is known that $\text{QMA} = \text{QMA}_{poly}$ is robust with respect to error bounds in the following sense.

Theorem 1 *Let q be any positive polynomial and let $a > b$ be constants in* $(0,1)$. *Then* $\text{QMA}(a,b) = \text{QMA}(\frac{2}{3}, \frac{1}{3}) = \text{QMA}(1 - 2^{-q}, 2^{-q})$.

A proof of this theorem appears in Section 14.2 of Kitaev, Shen, and Vyalyi[12]. A downside of this proof is that the size of the quantum certificate grows as the error decreases. In the case above the growth is only polynomial and so the resulting certificate is still polynomial in size, but if the function $m(n)$ is sub-polynomial then the increase in size is substantial. In this section we give a different procedure for reducing error that does not require the size of the certificate to grow. This result was achieved by the present author and John Watrous and is currently unpublished[14].

Theorem 2 (Marriott and Watrous) *Let q be any positive polynomial, let $a > b$ be constants $\in (0,1)$, and let $m : \mathbb{N} \to \mathbb{N}$ be any polynomial-time computable function. Then* $\mathrm{QMA}_m(a,b) = \mathrm{QMA}_m(\frac{2}{3}, \frac{1}{3}) = \mathrm{QMA}_m(1-2^{-q}, 2^{-q})$.

Proof. Assume $L \in \mathrm{QMA}_m(a,b)$, and V is a verification procedure that witnesses this fact. We will describe a new m-qubit verification procedure W with exponentially small completeness and soundness error.

It will simplify matters to assume hereafter that the input x is fixed. (It will be clear that the new verification procedure W is polynomial-time uniform.) We will write V to denote the unitary operator V_n, which acts on $n+m+k$ qubits. Let $\Pi_1 = I_{n+m+k-1} \otimes |1\rangle\langle 1|$ and $\Pi_0 = I_{n+m+k-1} \otimes |0\rangle\langle 0|$ denote the projections onto the spaces for which the output qubit (of V) is set to 1 and 0, respectively. (In general we will write I_l to denote the identity operator acting on l qubits.) Also define projections Δ_1 and Δ_0 as follows: $\Delta_1 = |x\rangle\langle x| \otimes I_m \otimes |0^k\rangle\langle 0^k|$ and $\Delta_0 = I_{n+m+k} - \Delta_1$. The projection Δ_1 is the projection onto the space for which the k ancilla qubits of V are all set to their initial state $|0^k\rangle$, and the n input bits still hold the input x, and Δ_0 is the projection onto the orthogonal complement of this space. We may therefore view the projections $\{\Delta_0, \Delta_1\}$ as describing a measurement whose outcome is 1 when the ancilla qubits of V are initialized to all zeros and the input has not been changed, and whose outcome is 0 otherwise.

The new verification procedure W takes an n qubit input string and an m qubit certificate as input and uses some number of work qubits as well. These work qubits will be viewed as consisting of the work qubits of V along with some additional work qubits. We will view the qubits comprising the input string and the certificate

```
Given input $|x\rangle$ and certificate $|\psi\rangle$. Assume the first $n$ qubit of $X$ contain the input and the next $m$ qubits of $X$ contain $|\psi\rangle$ and the remaining $k$ qubits of $X$ are set to the state $|0^k\rangle$.
Set $r_0 \leftarrow 1$ and $i \leftarrow 1$.
Repeat
    Apply $V$ to $X$ and measure $X$ with respect to the measurement described by $\{\Pi_0, \Pi_1\}$.
    Let $r_i$ denote the outcome, and set $i \leftarrow i + 1$.
    Apply $V^\dagger$ to $X$ and measure $X$ with respect to the measurement described by $\{\Delta_0, \Delta_1\}$.
    Let $r_i$ denote the outcome, and set $i \leftarrow i + 1$.
Until $i \geq N$ (where $N$ is chosen depending on the desired error bound).
For each $i = 1, \ldots, N$ set $s_i \leftarrow \begin{cases} 1 & \text{if } r_i = r_{i-1} \\ 0 & \text{if } r_i \neq r_{i-1}. \end{cases}$ Accept if $\sum_{i=1}^{N} s_i \geq N \cdot \frac{a+b}{2}$, reject otherwise.
```

Figure 2.1: Specification of verification procedure W.

together with the work qubits of V collectively as a single $n + m + k$ qubit register X. The procedure is described in Figure 2.1.

Figure 2.2 gives a quantum circuit illustrating this procedure for the case $N = 5$. (In this figure, S represents the computation described in the last step of the description of W. This step is assumed to be performed reversibly.)

First, suppose that $|\phi\rangle = |x\rangle|\psi\rangle|0^k\rangle$ is an eigenvector of the operator $A = \Delta_1 V^\dagger \Pi_1 V \Delta_1$ with eigenvalue p. Notice that if $|\psi\rangle$ were given as a certificate to V for the input x it would be accepted with probability p, since the probability of acceptance is

$$||\Pi_1 V|\phi\rangle||^2 = ||\Pi_1 V \Delta_1|\phi\rangle||^2 = \langle\phi|\Delta_1 V^\dagger \Pi_1 V \Delta_1|\phi\rangle = p.$$

We claim that the probability associated with obtaining the sequence $(s_1, \ldots, s_N)$ is precisely

$$p^{w(s)}(1-p)^{N-w(s)}$$

where $w(s) = \sum_{i=1}^{N} s_i$. By standard use of Chernoff-type bounds this will imply that W has exponentially small error for an appropriately chosen value of N. This is straightforward if $p = 0$ or $p = 1$, so assume $0 < p < 1$. Define unit vectors $|\gamma_0\rangle$, $|\gamma_1\rangle$, $|\delta_0\rangle$, and $|\delta_1\rangle$ as follows:

$$|\gamma_0\rangle = \frac{\Pi_0 V \Delta_1|\phi\rangle}{\|\Pi_0 V \Delta_1|\phi\rangle\|}, \qquad |\gamma_1\rangle = \frac{\Pi_1 V \Delta_1|\phi\rangle}{\|\Pi_1 V \Delta_1|\phi\rangle\|},$$
$$|\delta_0\rangle = \frac{\Delta_0 V \Pi_1|\gamma_1\rangle}{\|\Delta_0 V \Pi_1|\gamma_1\rangle\|}, \qquad |\delta_1\rangle = \frac{\Delta_1 V \Pi_1|\gamma_1\rangle}{\|\Delta_1 V \Pi_1|\gamma_1\rangle\|}.$$

Note that $|\delta_1\rangle = |\phi\rangle$, which follows from the fact that $|\phi\rangle$ is an eigenvector of A. We will show that

$$\begin{aligned} V|\delta_0\rangle &= -\sqrt{p}\,|\gamma_0\rangle + \sqrt{1-p}\,|\gamma_1\rangle \\ V|\delta_1\rangle &= \sqrt{1-p}\,|\gamma_0\rangle + \sqrt{p}\,|\gamma_1\rangle. \end{aligned} \tag{2.1}$$

We have $\|\Pi_1 V \Delta_1|\phi\rangle\| = \sqrt{p}$ and thus $\|\Pi_0 V \Delta_1|\phi\rangle\| = \sqrt{1-p}$. This implies that

$$\sqrt{1-p}\,|\gamma_0\rangle + \sqrt{p}\,|\gamma_1\rangle = \Pi_0 V \Delta_1|\phi\rangle + \Pi_1 V \Delta_1|\phi\rangle = V\Delta_1|\phi\rangle = V|\delta_1\rangle.$$

Similarly,

$$\|\Delta_1 V^\dagger \Pi_1|\gamma_1\rangle\| = \frac{\|\Delta_1 V^\dagger \Pi_1 V \Delta_1|\phi\rangle\|}{\|\Pi_1 V \Delta_1|\phi\rangle\|} = \frac{p}{\sqrt{p}} = \sqrt{p},$$

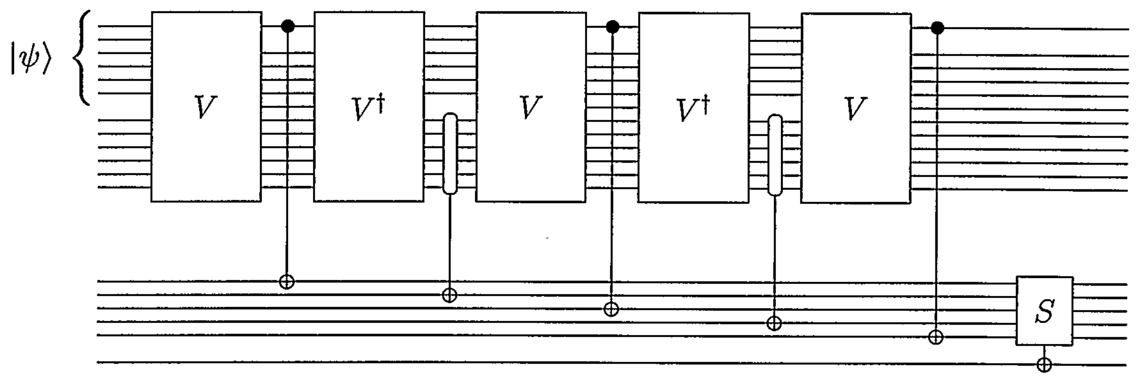

Figure 2.2: Example circuit diagram for verification procedure W.

and thus $\|\Delta_0 V^\dagger \Pi_1 |\gamma_1\rangle\| = \sqrt{1-p}$. This implies that

$$\sqrt{1-p}\,|\delta_0\rangle + \sqrt{p}\,|\delta_1\rangle = \Delta_0 V^\dagger \Pi_1 |\gamma_1\rangle + \Delta_1 V^\dagger \Pi_1 |\gamma_1\rangle = V^\dagger \Pi_1 |\gamma_1\rangle = V^\dagger |\gamma_1\rangle.$$

This equation implies that $\langle\gamma_1|V|\delta_0\rangle = \sqrt{1-p}$, so we have $V|\delta_0\rangle = \sqrt{1-p}\,|\gamma_1\rangle + |\mu\rangle$ for some vector $|\mu\rangle$ of length $\sqrt{p}$ that is orthogonal to $|\gamma_1\rangle$. Since $V|\delta_0\rangle$ is orthogonal to $V|\delta_1\rangle$, which we already know is $\sqrt{1-p}\,|\gamma_0\rangle + \sqrt{p}\,|\gamma_1\rangle$, we must have that $|\mu\rangle = -\sqrt{p}\,|\gamma_0\rangle$. We have therefore shown (2.1). It will be convenient to also write these equations as follows:

$$\begin{aligned} V^\dagger|\gamma_0\rangle &= -\sqrt{p}\,|\delta_0\rangle + \sqrt{1-p}\,|\delta_1\rangle \\ V^\dagger|\gamma_1\rangle &= \sqrt{1-p}\,|\delta_0\rangle + \sqrt{p}\,|\delta_1\rangle. \end{aligned}$$

With the above equations in hand, it is not difficult to determine the probability associated with each sequence of measurement outcomes. We begin in state $|\phi\rangle = |\delta_1\rangle$ and apply V. After the measurement described by $\{\Pi_0, \Pi_1\}$ the (renormalized)

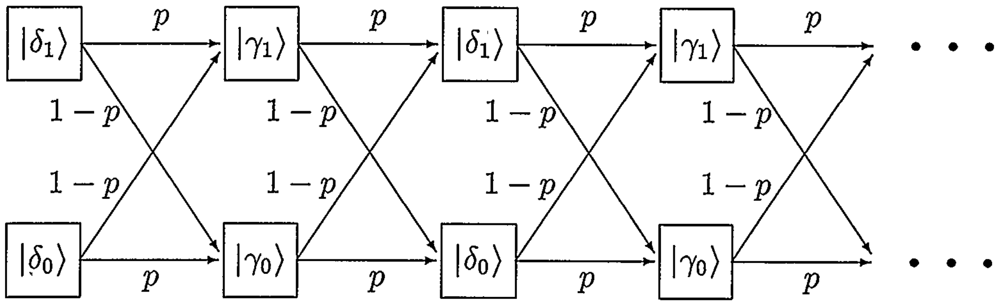

Figure 2.3: Transition probabilities for verification procedure M.

state of register X becomes $|\gamma_0\rangle$ or $|\gamma_1\rangle$ according to whether the outcome is 0 or 1, with associated probabilities $1-p$ and p, respectively. If instead we were to start in state $|\delta_0\rangle$, the renormalized states after measurement would be the same, but the probabilities are reversed: we have probability p associated with outcome 0 and probability $1-p$ with outcome 1. For the second step of the loop the situation is similar: if the register X is in state $|\gamma_1\rangle$, the transformation $V^\dagger$ is applied, and the state is measured via the measurement $\{\Delta_0, \Delta_1\}$, the renormalized state after measurement will be either $|\delta_1\rangle$ or $|\delta_0\rangle$, with associated probabilities p and $1-p$, and if instead the initial state is $|\gamma_0\rangle$ rather than $|\gamma_1\rangle$ the renormalized states after the measurement are again the same, but the probabilities are reversed. These transition probabilities are illustrated in Figure 2.3. In all cases we see that the probability of obtaining the same outcome as for the previous measurement is p, and the probability of the opposite outcome is $1-p$. (We set $r_0 = 1$ to include the first measurement outcome in this pattern, since the process starts in state $|\delta_1\rangle$.) The probability associated with a given sequence $s = (s_1, \ldots, s_N)$ is $p^{w(s)}(1-p)^{N-w(s)}$ as claimed, since each s_i is 1 if the measurement outcomes r_{i-1} and r_i are equal, and is

0 otherwise.

The above pattern forms only if we are provided with a certificate that makes $|\phi\rangle$ an eigenvector of $A = \Delta_1 V^\dagger \Pi_1 V \Delta_1$. We can use this pattern that forms with eigenvectors to show that the soundness and completeness conditions hold for W with exponentially small error. First we note that since the operator A is Hermitian, then by the Raleigh-Ritz theorem, the maximally accepting certificate $|\psi_{max}\rangle$ will make $|\phi_{max}\rangle$ an eigenvector of the operator A. Let this maximally accepting eigenvector have eigenvalue p_{max}. Then every other eigenvector of A has eigenvalue less than or equal to p_{max}. Every eigenvector exhibits the pattern recognized above, and so every eigenvector will cause W to accept with probability less than or equal to the acceptance probability of $|\phi_{max}\rangle$.

Now consider quantum states written using the eigenvectors as a basis. In general we will consider the state $|\Psi\rangle = \sum_i \alpha_i |\psi_i\rangle$, where the α_is meet the standard requirement and where each $|\psi_i\rangle$ is an eigenvector. Now assume that $|\psi_i\rangle$ has eigenvalue p_i and assume the eigenvectors are ordered such that $p_1 \geq p_2 ... \geq p_{2^m}$, where 2^m is the number of eigenvectors. Then if this state is input to V it will be accepted with probability $\sum_i |\alpha_i|^2 p_i \leq p_{max}$, with equality only when the vectors with non-zero amplitude all have eigenvalue p_{max}. Since this case is trivial to solve we will only consider when this is not so. We note now that both eigenvectors exist as one state in a set of four that parallel the states above $|\delta_0^i\rangle, |\delta_1^i\rangle, |\gamma_0^i\rangle, |\gamma_1^i\rangle$, where $|\psi_i\rangle = |\delta_1^i\rangle$. When we apply the first step of our circuit W we see that we get the pattern we expect from both eigenvectors, weighted by α_i.

$$\begin{aligned} V|\Psi\rangle &= \sum_{i=1}^{2^m} \alpha_i V|\psi_i\rangle \\ &= \sum_{i=1}^{2^m} \alpha_i \left(\sqrt{p_i}|\gamma_1^i\rangle + \sqrt{1-p_i}|\gamma_0^i\rangle\right) \end{aligned}$$

The observation made at this point to determine acceptance will select the same outcome for each eigenvector and the state will collapse. When the state is normalized the weightings α_i will change depending on the outcome. Whatever the outcome, the readjusted weightings will favor the eigenvectors for which the outcome happens with greater probability. So if the outcome is an accept then the new weightings will see the greatest rise for the maximally accepting eigenvector and the greatest fall for the minimally accepting eigenvector, and vice versa. The accepting case is displayed below without the normalizing factor.

$$\Pi_1 V|\Psi\rangle = \sum_{i=1}^{2^m} \alpha_i \sqrt{p_i}|\gamma_1^i\rangle$$

Note that since p_1 is the greatest, the weight of the first vector is increased the most. If we now consider the next step of the algorithm with a new set of α_is we will see the observation at the next step behaves the same. So with each observation the bias of the proof shifts to one end of the spectrum or the other. What is important to note is that a combination of this type can not be used to increase the acceptance of W beyond that achieved by $|\psi_{max}\rangle$. For even if the bias of the mixture can eventually be shifted onto the set of four vectors with maximal acceptance (that is to the set containing $|\psi_{max}\rangle$) then the remaining cycles of the algorithm will continue as though $|\psi_{max}\rangle$ were given in the first place. Since the initial steps of the loop will have evolved with lesser probability with the mixture than with the pure $|\psi_{max}\rangle$ then the mixture will be accepted by W with probability less than that of $|\psi_{max}\rangle$. So $|\psi_{max}\rangle$ is the maximally accepting vector for both V and W. We can restrict our analysis of the soundness and completeness conditions to this maximally accepting certificate.

Now consider the case that the input is in the language. In this case we know that there exists a quantum certificate that will cause V to accept with probability greater than a, and so $p_{max} > a$. Since $|\phi_{max}\rangle$ is an eigenvector the above analysis shows that this proof will achieve exponentially small error if provided to the new procedure W. Again, if $|\psi_{max}\rangle$ is the maximally accepting certificate for V and it is provided as a certificate for W then the error is exponentially small for some choice of N that is polynomial in n. So the completeness condition holds.

Consider also the case where the input is not in the language. In this case we know that all quantum states are poor certificates, or more specifically that all quantum states will cause acceptance in V with probability less than b. This obviously includes $|\phi_{max}\rangle$. Since we know this state is an eigenvector it will have exponentially small error, and thus exponentially small probability of acceptance. Again, if $|\psi_{max}\rangle$ is the maximally accepting certificate for V and it is provided as a certificate for W then it will accept with the highest probability and for some choice of N that is polynomial in n this probability is exponentially small. So the soundness condition also holds.

We will choose the larger value of N from these two cases for our new procedure W and we note that W witnesses that L is in $\text{QMA}_m(1 - 2^{-q}, 2^{-q})$. ■

2.2 Logarithmic Length Proofs

We will now consider quantum verification procedures that are restricted to logarithmic length proofs. We can describe the length of the proof as a function in the input size, and we will say that a procedure has a logarithmic length proof if this function grows slower than $c \cdot \log(n)$ for some constant c. We will call the classes of languages that can be decided by these procedures $\mathrm{QMA}_{log}(a,b) = \cup_{i=1}^{\infty} \mathrm{QMA}_{i \cdot \log(n)}(a,b)$. The robustness demonstrated in the last section means that as long as a and b satisfy the conditions of the last theorem, any a and b will do. We will choose the common values of $a = \frac{2}{3}$ and $b = \frac{1}{3}$. So we will denote $\mathrm{QMA}_{log}(\frac{2}{3}, \frac{1}{3}) = \mathrm{QMA}_{log}$.

Classically we find that logarithmic length proofs are of no use since the proof is so short that you can try all the possible proofs in polynomial time. One can construct a quantum argument that parallels this one by trying all possible proofs in tandem. The argument, however, hinges on the fact that QMA is a robust class, and with only the weak proof of robustness the argument is squashed since amplifying also implies increasing the size of the proof. With the strong proof of the last section we can construct the argument. For this theorem we use the operator BQTIME to generate a class of problems such that $\mathrm{BQTIME}(f(n))$ is the class of languages solvable by bounded error quantum circuits with size $f(n)$. $\mathrm{BQTIME}(\mathrm{poly}(n))$ is equal to BQP where $\mathrm{poly}(n)$ is the class of all polynomial time computable functions.

Theorem 3 (Marriott) *Let $m(n)$ be a polynomial-time computable function. Then* $\mathrm{QMA}_{m(n)} \subseteq \mathrm{BQTIME}(2^{m(n)} \cdot \mathrm{poly}(n))$.

Proof: Assume $L \in \mathrm{QMA}_{m(n)}$ and V is a verification procedure that witnesses this fact with error less than $\frac{1}{2^{2 \cdot m(n)}}$, and let q be a polynomial that bounds the size of

```
Given input $|x\rangle$. Assume that $|x\rangle$ is in register $X$ and that there are two more
registers each of length $m$ initialized to $|0^m\rangle$, called $Y_0$ and $Y_1$.
Set $i \leftarrow 1$.
Repeat
    Apply $m$ Hadamard gates to register $Y_0$.
    Use controlled-not gates from $Y_0$ to $Y_1$ to create $m$ copies of $|00\rangle + |11\rangle$.
    Run the verification procedure $V$ using register $X$ as the input and register
    $Y_0$ as the certificate.
    Let $r_i$ be the outcome and set $i \leftarrow i + 1$
Until $i \geq N$ (where $N$ is $O(2^{m(n)})$)
Accept if $\sum_{j=1}^{N} r_i \geq N \cdot \frac{2^{m(n)-1}+1}{2^{2 \cdot m(n)+1}}$, reject otherwise.
```

Figure 2.4: Specification of quantum circuit W for log length proof simulation.

V. Here it is important that we use the strong robustness from above so that we can ensure that the length of the proof remains $m(n)$. If we use the weak robustness instead we find that to ensure that the verification procedure has the error we desire will cause the proof to expand in length, which will frustrate the final argument of the proof for some choices of $m(n)$. Figure 2.4 describes a new quantum circuit, W, that decides L.

W first constructs a totally mixed state over a number of qubits equal to the length of the proof then simulates the procedure V using this state in place of the proof and accepts as V.

Fix the input x and let $|\psi_{max}\rangle$ be the maximally accepting proof for the input. $|\psi_{max}\rangle$ is of length $m(n)$ so the dimension of the space in which it resides is $2^{m(n)}$. So $|\psi_{max}\rangle$ is a member of set of vectors, of size $2^{m(n)}$, that span this space. The totally mixed state can be written as a uniform linear combination of any set of spanning vectors, including the set described above

$$\rho = \frac{1}{2^{m(n)}} \sum_{i=0}^{2^{m(n)}-1} |\psi_i\rangle\langle\psi_i|$$

The totally mixed state can also be written using the computational basis as

$$\rho = \frac{1}{2^{m(n)}} \sum_{i=0}^{2^{m(n)}-1} |i\rangle\langle i|$$

which shows how we can construct this state in time $O(m(n))$. The first step in the loop creates this state in register Y_0 if we trace out the contents of register Y_1. After the controlled-not gates in this step we can discard of the qubits in register Y_1. Describing this state as a uniform mixture corresponds to the second equation above, though as stated both equations above characterize the totally mixed state.

Consider now when the input x is in the language L. We know that V would accept $|\psi_{max}\rangle$ with probability greater than $1 - \frac{1}{2^{2 \cdot m(n)}}$ so the probability of accepting the totally mixed state is greater than $\frac{1}{2^{m(n)}}(1 - \frac{1}{2^{2 \cdot m(n)}}) \geq \frac{1}{2^{m(n)+1}}$. Next consider when the input is not in L. Now we know that every proof will accept with probability less than $\frac{1}{2^{2 \cdot m(n)}}$ and this includes the totally mixed state. The gap between these probabilities is greater than $\frac{1}{2^{m(n)+2}}$. We need to amplify this gap to greater than $1/3$ to be considered bounded error. The number of repetitions necessary to amplify such a gap is inversely proportional to the size of the gap using the standard procedure of repetition and majority vote (this is a basic application of Chernoff type bounds). Thus, N, the bound on the loop is $O(2^{m(n)})$ to achieve a constant error bound. Every step of W is polynomial time except for the number of repetitions which may be exponential for some choices of $m(n)$. W takes time $O(q \cdot 2^{m(n)})$. So W witnesses that L is in BQTIME$(2^{m(n)} \cdot \text{poly}(n))$. ■

The immediate corollary for logarithmic length proofs follows:

Corollary 1 QMA$_{log}$ = BQP

Chapter 3

Quantum Quantifiers

Logical quantifiers were first introduced as an alternate definition of non-determinism, see [19] for an overview of their application. This characterization was an obvious extension if one recognized the logical significance of the difference in the soundness and completeness conditions for non-deterministic classes given in the first chapter. Recall the definition of NP. For input in the language there *exists* a certificate that will cause acceptance, and for input not in the language *all* certificates cause rejection. This observation led to a more general picture of non-determinism in terms of an operator that could be applied to a base class.

The way this worked was to first choose a base class, in the simplest case this class was P. The completeness condition of the class P states that the solving machine must accept. The soundness condition states that the solving machine must reject. If we wish to apply a logical quantifier to this class we apply the quantifier to the completeness condition and the logical negation of the quantifier to the soundness condition. Applying the exists quantifier to the base class P yields the following soundness and completeness conditions. If x is the input, then if $x \in L$ there must exists some string y such that the solving machine must accept $\langle x, y \rangle$, and if $x \notin L$ for all strings y the solving machine must reject $\langle x, y \rangle$. This new class defined by $\exists \cdot$P is NP, and the other possibility $\forall \cdot$P is co-NP.

Other base classes could be chosen. For instance one could use BPP instead of P. Then one would expect that as $\exists \cdot$P = NP so should $\exists \cdot$BPP = MA. This turns out to be an open question however. The bounded error property of the base class BPP ends up fudging the adaptation of the proof for the $\exists$·P = NP case. In this chapter we will explore an expansion of the quantifiers that allows us to address this open question, if not answer it, as well as redefine all of the quantum classes we have already introduced.

3.1 Quantifiers in the Classical Setting

Three primary quantifiers will be studied: $\exists, \forall$, and $\exists^+$. It is assumed that most readers are familiar with the $\exists$ and $\forall$ quantifiers and their definitions but the $\exists^+$ may be new to some. This quantifier can often be read as "for the overwhelming majority of", where overwhelming is a vague statement implying that the majority is bounded away from $\frac{1}{2}$ by a constant in the same way as we have used it in the previous chapters. So the logical predicate $\exists_y^+ : P(y)$ is true if and only if $P(y)$ is true for at least $\frac{1}{2} + \epsilon$ of the possible values of y for some $0 < \epsilon < \frac{1}{2}$. Below we define what it means for these quantifiers to be applied to a complexity class.

Definition 17 *A language, $L \subseteq \Sigma^*$, is said to be in the class $\exists \cdot \mathcal{C}$ if and only if:*

$$x \in L \Rightarrow \exists_{y \in \Sigma^{p(|x|)}} : (x, y) \in A$$

$$x \notin L \Rightarrow \forall_{y \in \Sigma^{p(|x|)}} : (x, y) \notin A$$

for some polynomial p and some language $A \in \mathcal{C}$.

Definition 18 *A language, $L \subseteq \Sigma^*$, is said to be in the class $\forall \cdot \mathcal{C}$ if and only if:*

$$x \in L \Rightarrow \forall_{y \in \Sigma^{p(|x|)}} : (x, y) \in A$$

$$x \notin L \Rightarrow \exists_{y \in \Sigma^{p(|x|)}} : (x, y) \notin A$$

for some polynomial p and some language $A \in \mathcal{C}$.

Definition 19 *A language, $L \subseteq \Sigma^*$, is said to be in the class $\exists^+ \cdot \mathcal{C}$ if and only if:*

$$x \in L \Rightarrow \exists^+_{y \in \Sigma^{p(|x|)}} : (x, y) \in A$$

$$x \notin L \Rightarrow \exists^+_{y \in \Sigma^{p(|x|)}} : (x, y) \notin A$$

for some polynomial p and some language $A \in \mathcal{C}$.

In all of these definition we note that the length of the quantified string is restricted to be polynomial in the length of the input string. This is a common restriction and one we will adopt in later sections to describe our modified quantifiers. Sometimes we will omit this restriction in the notation, but that is not to suggest that the restriction is not there, but rather it is intended to make the statements easier to read due to less clutter.

We apply these definitions to the obvious class, P, and state the following theorem.

Theorem 4 *The following all hold:*

1. $\mathrm{NP} = \exists \cdot \mathrm{P}$
2. *co*-$\mathrm{NP} = \forall \cdot \mathrm{P}$

3. BPP $= \exists^+ \cdot \mathrm{P}$

These quantifiers became the building blocks of infinite complexity classes that are formed by repeatedly applying these quantifiers to P or in the general case any complexity class. When using P as the derivative class we discover that this formulation already matches up with two well studied hierarchies. Alternating $\exists$ and $\forall$ quantifiers characterize the various levels of the polynomial time hierarchy, PH. Furthermore when the $\exists^+$ quantifier is added to the mix we can characterize the Arthur-Merlin classes of Babai. Below the number of alternating quantifiers is equal to the level of the hierarchy.

Theorem 5 *1.* MA $= \exists\exists^+ \cdot \mathrm{P}$

2. AM $= \exists^+\exists \cdot \mathrm{P} = \exists^+\exists\exists^+ \cdot \mathrm{P}$

3. AMA... $= \exists^+\exists \cdots \mathrm{P} = \mathrm{AM}$

4. MAM... $= \exists\exists^+ \cdots \mathrm{P} = \mathrm{AM}$

We will note that the $\cdot$ used in the notation is meant to emphasize an important property when stacking these quantifiers. Treating the $\exists$ quantifier as an operator on complexity classes we might write $\exists(\mathrm{P})$ to define NP, and $\exists^+(\mathrm{P})$ to define BPP. We could try $\exists(\exists^+(\mathrm{P}))$ for MA but would run into trouble since $\exists(\exists^+(\mathrm{P})) = \exists(\mathrm{BPP})$, and as stated above this is an open question. However it is known that MA $=$ $\exists\exists^+(\mathrm{P})$ when the two quantifiers are taken together as one operator. We use the $\cdot$ to emphasize that the entire string of quantifiers taken together is applied as an operator to the base class. It turns out that for the new quantifiers defined later that this distinction is unnecessary since all interpretations are equal, but for now the $\cdot$ has a role.

We return to the observation that the class $\exists\cdot$BPP is not known to be equal to MA. When attempting to prove such an equality by straightforward simulation we find that we have problems showing that MA $\subseteq \exists\cdot$BPP because we must show that the new machine runs with bounded error even when an incorrect proof is given. This problem also exists if attempting to show that $\exists^{+}\cdot$MA $=$ AM.

It is desirable to attempt to extend the quantifier results to the quantum case, and it is interesting to note that this is not a trivial task. The natural randomness built into quantum mechanics causes problems for the straightforward extension of these results in the same way the bounded error was a problem classically. Consider the task of demonstrating that $\exists\cdot$BQP $=$ QCMA (an obvious extension of the results above). It is easy to show that any language from $\exists\cdot$BQP is contained in QCMA, that is $\exists\cdot$BQP $\subseteq$ QCMA. However the other direction poses an interesting problem. It is not hard to show that the verification part of the procedure runs with bounded error when the maximally accepting proofs are provided, but we must also show that it works with bounded error for any proof. There is no reason to believe that the machine works with bounded error for every proof, and in fact this seems likely false. So the desire to extend the quantifier results requires us to look at quantifiers in a slightly different setting, one that will eliminate this problem.

Furthermore, it is desirable to characterize the more interesting quantum non-deterministic classes like QMA and QAM. However for these classes we will have to consider a quantum quantifier, or rather a logical quantifier over quantum states, rather than over classical strings. The next section outlines some of the novel changes we must make to the study of quantifiers so that we can extend these results into the burgeoning quantum world.

3.2 Promise Problems

Since the problem we encountered with our attempt to apply the quantifiers to the quantum classes was a problem with the certificates that were not important in determining the acceptance or rejection criteria it was hypothesized that switching from languages to promise problems would allow us to ignore the problematic cases. Another way that we can view this change of perspective is that we allow the characteristic function of a language to be a partial functions, or rather defined on any subset of Σ^*. In fact this change will allow us to solve our problem and this will be demonstrated in the following section. Promise problems are a way to shrink the domain of possible members of a problem from all of Σ^* to one of its subsets. This idea has been studied in the past by Even, Selman, and Yacobi [11]. Below we show two equivalent definitions for promise problems and even these differ from the definition in [11], though remain equivalent. We will adopt the second definition for our study.

Definition 20 *A promise problem is a pair of sets (A, B) such that $A \subseteq B \subseteq \Sigma^*$.*

Equivalently we can define a promise problem a second way:

Definition 21 *A promise problem is a pair of disjoint subsets of Σ^*, (A_{yes}, A_{no}).*

In this case we can think of A_{yes} and A_{no} as sets of *yes-instances* and *no-instances* of this problem. We can see that the promise set $B = A_{yes} \cup A_{no}$ and it is clear that

$A_{yes} = A \subseteq B$ and $A_{no} = B \backslash A$.

We can then define classes of promise problems based on the power of the circuits that compute them in a way similar to regular complexity classes. We shall do this following the lead of Even, Selman and Yacobi who defined the class NPP as a natural extension of the class NP. We will adopt a different notation by adding "promise-" to the beginning of any standard complexity class that we wish to extend to promise problems. Using this notation the class NPP is equivalent to promise-NP.

The primary difference between the criteria of membership in a standard complexity class and its promise problem extension is that the acceptance/rejection criteria of the standard complexity class need only be met within the sets A_{yes} and A_{no}. We will say that a Turing machine or circuit is $\mathcal{C}$-consistent with a given promise problem if the machine(circuit) meets the acceptance/rejection criteria of the complexity class $\mathcal{C}$ for all members of the promise set. This is of course an informal idea of "consistence" although we can define a more precise idea with respect to the classes we have already looked at. Below we make use of the fact that we have defined all the complexity classes from the first chapter with explicit soundness and completeness predicates. Predicates of this type take as input a machine and an input string, and are true or false given the relationship between the machine and the input.

Definition 22 *Let $\mathcal{C}$ be any class with an explicit completeness predicate, P_C, and soundness predicate, P_S, and let (A_{yes}, A_{no}) be a promise problem. Then we will say that a Turing machine (or circuit family), M, is $\mathcal{C}$-consistent with (A_{yes}, A_{no}) if and only if:*

1. *(Completeness)* $x \in A_{yes} \Rightarrow P_C(x, M) = 1$

2. (Soundness) $x \in A_{no} \Rightarrow P_S(x, M) = 1$

We can look back at the classes defined in the first chapter and we notice that each has been stated with these predicates made explicit. We now show how this idea of consistence can be used with respect to the classes BPP, EQP, and BQP.

Definition 23 *A Turing machine, M, is* BPP*-consistent with respect to a promise problem, (A_{yes}, A_{no}), if and only if M runs in polynomial time for all inputs from the promise set and*

1. *(Completeness)* $x \in A_{yes} \Rightarrow Pr[M(x) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \in A_{no} \Rightarrow Pr[M(x) = 1] \leq \frac{1}{3}$

Definition 24 *A circuit family, Q, is* EQP*-consistent with respect to a promise problem, (A_{yes}, A_{no}), if and only if Q is a uniformly generated polynomial sized quantum circuit family and*

1. *(Completeness)* $x \in A_{yes} \Rightarrow Q(x) = 1$
2. *(Soundness)* $x \in A_{no} \Rightarrow Q(x) = 0$

Definition 25 *A circuit family, Q, is* BQP*-consistent with respect to a promise problem, (A_{yes}, A_{no}), if and only if Q is a uniformly generated polynomial sized quantum circuit family and*

1. *(Completeness)* $x \in A_{yes} \Rightarrow Pr[Q(x) = 1] \geq \frac{2}{3}$
2. *(Soundness)* $x \in A_{no} \Rightarrow Pr[Q(x) = 1] \leq \frac{1}{3}$

It should be clear that the constants $\frac{2}{3}$ and $\frac{1}{3}$ can be replaced with $\frac{1}{2}+\epsilon$ and $\frac{1}{2}-\epsilon$ for any $0 < \epsilon < \frac{1}{2}$ in the definition of promise-BPP and promise-BQP. These promise classes inherit their robustness from their related complexity classes. Using these definitions we can define the classes of promise problems, promise-BPP, promise-EQP, and promise-BQP, which will be used throughout, as well as any other extensions of complexity class where a notion of $\mathcal{C}$-consistency can be naturally defined.

Definition 26 *Let (A_{yes}, A_{no}) be a promise problem. $(A_{yes}, A_{no}) \in$ promise-$\mathcal{C}$ if and only if there is some Turing machine (or circuit family) Q, such that Q is $\mathcal{C}$-consistent with respect to (A_{yes}, A_{no}).*

We can note that if $L_{yes} \in \mathcal{C}$ then for all sets L_{no} disjoint from L_{yes} it is true that $(L_{yes}, L_{no}) \in$ promise-$\mathcal{C}$. From this fact we can see that $\mathcal{C} \subseteq$ promise-$\mathcal{C}$ or more formally, $L_{yes} \in \mathcal{C} \Leftrightarrow (L_{yes}, \overline{L_{yes}}) \in$ promise-$\mathcal{C}$. We can use this fact to characterize a relationship between regular complexity classes and their promise class analogues that will allow us to prove properties of one class and have them be equally true for the other class. For instance the proposition below demonstrates another way that these classes are linked.

Proposition 3 *Let $\mathcal{C}$ and $\mathcal{D}$ be classes with explicit soundness and completeness predicates. Then promise-$\mathcal{C}$ = promise-$\mathcal{D} \Leftrightarrow \mathcal{C} = \mathcal{D}$*

Proof: If the promise classes are the same then since the complexity classes represent a specifically defined subsets (namely all promise problems where the union of the "yes" and "no" sets is Σ^*) of the promise class the complexity classes must be equal.

Now assume that the promise classes are different; promise-$\mathcal{C} \neq$ promise-$\mathcal{D}$. We may assume, without loss of generality, that (A_{yes}, A_{no}) is a promise problem in promise-$\mathcal{C}$ and not in promise-$\mathcal{D}$. Since (A_{yes}, A_{no}) is in promise-$\mathcal{C}$ there must exists a set $B \in \mathcal{C}$ such that $A_{yes} \subseteq B$ and $A_{no} \subseteq \overline{B}$. B cannot be a member of $\mathcal{D}$ or else (A_{yes}, A_{no}) would be in promise-$\mathcal{D}$. So $\mathcal{C} \neq \mathcal{D}$. ■

We note that though complexity classes and promise classes have many similarities these classes should not be considered equal since promise-$\mathcal{C}$ contains many more promise problems than just those that are also in $\mathcal{C}$. For any language L, $(L, \emptyset) \in$ promise-$\mathcal{C}$. These facts bring to light the importance of the "no" set. In promise problems the "no" set is as important to the problem as the "yes" set, since changing either set can change whether a problem is in a given promise class.

It will be desirable to discuss the complement of a promise class, which requires the notion of the complement of a promise problem. The natural extension of the complement of a language is to take the complement of the problem set with respect to the domain of the promise problem. This leads to the next two definitions.

Definition 27 *The complement of a promise problem (A_{yes}, A_{no}) is the promise problem (A_{no}, A_{yes}).*

Definition 28 $(A_{yes}, A_{no}) \in$ *co-promise-$\mathcal{C}$* $\Leftrightarrow$ $(A_{no}, A_{yes}) \in$ *promise-$\mathcal{C}$.*

We notice then that the complement class of a promise class is formed by swapping the soundness and completeness predicates in the definition of the base class. We will say that a promise class is closed under complement if for every $(A_{yes}, A_{no}) \in$ promise-$\mathcal{C}$, $(A_{no}, A_{yes}) \in$ promise-$\mathcal{C}$. The proposition below follows immediately from the proposition above.

Proposition 4 *Let $\mathcal{C}$ be a class with explicit soundness and completeness predicates. Then promise-$\mathcal{C}$ is closed under complement if and only if $\mathcal{C}$ is closed under complement.*

3.3 Quantifiers and Promise Classes

We will now extend the idea of quantifiers to operating on promise classes. Below we have the definitions for the $\exists$, $\exists^+$, and $\forall$ quantifiers. We take $\exists^+_y$ to mean "for the overwhelming majority of strings y", or rather for some number of strings y greater than $\frac{1}{2}$ plus a constant. Commonly the number $\frac{2}{3}$ is used, though any other constant bounded away from $\frac{1}{2}$ and 1 will do.

Definition 29 *We will say that a promise problem, (L_{yes}, L_{no}), is in $\exists\cdot$promise-$\mathcal{C}$ if and only if:*

$$x \in L_{yes} \Rightarrow \exists_{y\in\Sigma^{p(|x|)}} : (x,y) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \forall_{y\in\Sigma^{p(|x|)}} : (x,y) \in A_{no}$$

for some $(A_{yes}, A_{no}) \in$ promise-$\mathcal{C}$ and some polynomial p.

Definition 30 *We will say that a promise problem, (L_{yes}, L_{no}), is in $\exists^+\cdot$promise-$\mathcal{C}$ if and only if:*

$$x \in L_{yes} \Rightarrow \exists^+_{y\in\Sigma^{p(|x|)}} : (x,y) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \exists^+_{y\in\Sigma^{p(|x|)}} : (x,y) \in A_{no}$$

for some $(A_{yes}, A_{no}) \in$ promise-$\mathcal{C}$ and some polynomial p.

Definition 31 *We will say that a promise problem, (L_{yes}, L_{no}), is in $\forall$·promise-$\mathcal{C}$ if and only if:*

$$x \in L_{yes} \Rightarrow \forall_{y \in \Sigma^{p(|x|)}} : (x, y) \in A_{yes}$$

$$x \in L_{no} \Rightarrow \exists_{y \in \Sigma^{p(|x|)}} : (x, y) \in A_{no}$$

for some $(A_{yes}, A_{no}) \in$ promise-$\mathcal{C}$ and some polynomial p.

In all three cases the quantifiers place specific restrictions on the promise problem in question. This restriction is made in both the soundness and completeness restrictions of each definition by choosing "yes"- and "no"-instances of a particular problem that must be included in the promise. This differs from the quantifiers considered earlier in that before the "yes" set completely determined the "no" set, whereas now each is defined independently.

Let us begin by showing a simple relationship between promise classes defined with these quantifiers. As is the case with the quantifiers used on standard complexity classes, the $\exists$ and $\forall$ quantifiers exist as complements. To prove this we use the fact that $\exists$ and $\forall$ are logically complementary and an assumption that the base class is closed under complement.

Proposition 5 *co-$\exists$·promise-$\mathcal{C}$ $=$ $\forall$·promise-$\mathcal{C}$ for any $\mathcal{C}$ that is closed under complement.*

Proof: Let $(L_{yes}, L_{no}) \in \exists$·promise-$\mathcal{C}$. Let $(A_{yes}, A_{no}) \in$ promise-$\mathcal{C}$ be the implied promise problem associated with (L_{yes}, L_{no}) and let Q be a circuit family (or Turing machine) that is $\mathcal{C}$-consistent with (A_{yes}, A_{no}). We will show that $(L_{no}, L_{yes}) \in$ $\forall$·promise-$\mathcal{C}$. Let $\overline{Q}$ be the circuit family that answers the opposite of Q. Then since $\mathcal{C}$ is closed under complement $\overline{Q}$ is $\mathcal{C}$-consistent with (A_{no}, A_{yes}). So $(A_{no}, A_{yes}) \in$ promise-$\mathcal{C}$. Now

$$x \in L_{no} \Rightarrow \forall_y : (x, y) \in A_{no}$$
$$x \in L_{yes} \Rightarrow \exists_y : (x, y) \in A_{yes}$$

so $(L_{no}, L_{yes}) \in \forall$·promise-C.

For the other direction we can show that co-$\forall$·promise-C $\subseteq \exists$·promise-C which follows by an identical argument. ∎

We can also note another similarity to the previously defined quantifiers in that stacking similar quantifiers is of no use. That is, two $\exists$ quantifiers reduce to one, two $\forall$ quantifiers reduce to one, and two $\exists^+$ quantifiers reduce to one. The proof is quite simple. We prove that two $\exists$ quantifiers reduce to one below and note that the proof for two $\forall$ or $\exists^+$ quantifiers is identical.

Proposition 6 $\exists\exists$·*promise-*$\mathcal{C}$ $= \exists$·*promise-*$\mathcal{C}$.

Proof: Let $(L_{yes}, L_{no}) \in \exists\exists$·promise-$\mathcal{C}$. Then there is a circuit family (or Turing machine), Q, that is $\mathcal{C}$-consistent with a promise problem (A_{yes}, A_{no}) such that

$$x \in L_{yes} \Rightarrow \exists_{y_1}\exists_{y_2} : (x, y_1, y_2) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \forall_{y_1}\forall_{y_2} : (x, y_1, y_2) \in A_{no}$$

We can combine the strings y_1 and y_2 into one string y, still of polynomial length, and make a new machine that begins by splitting the string y into two parts and then running Q on the input plus these two parts. We get that

$$x \in L_{yes} \Rightarrow \exists_y : (x, y) \in A'_{yes}$$
$$x \in L_{no} \Rightarrow \forall_y : (x, y) \in A'_{no}$$

for a new promise problem (A'_{yes}, A'_{no}) in promise-$\mathcal{C}$. So $(L_{yes}, L_{no}) \in \exists$·promise-$\mathcal{C}$. ∎

Proposition 7 $\forall\forall$·*promise-*$\mathcal{C}$ $= \forall$·*promise-*$\mathcal{C}$

Proposition 8 $\exists^+\exists^+$·*promise-*$\mathcal{C}$ $= \exists^+$·*promise-*$\mathcal{C}$.

Below we show that some of the classes that we can define using these operators coincide with the promise versions of quantum complexity classes that have already been studied under different names and different models. The first such relationship we will demonstrate is that adding classical non-determinism to a quantum machine can be characterized with the $\exists$ quantifier applied to a base class of promise-EQP. The proof idea for this theorem is based on the proof of $\exists$·P $=$ NP and relies on standard simulation. These proofs only work because we are considering promise problems rather than decision problems because we do not need to show how the new machines work on the inputs that are not considered by the definitions of the problem. The first theorem combines classical non-determinism with exact quantum computation. This class is not commonly studied, but the proof of this coincidence will form the template for the proofs in the remainder of this chapter.

Theorem 6 (Marriott) $\exists$·*promise*-EQP $=$ *promise*-QCNP

Proof: Let $(L_{yes}, L_{no}) \in \exists$·promise-EQP. Then there is some polynomial sized quantum circuit family, Q, that is EQP-consistent with a promise problem (A_{yes}, A_{no}) such that

$$x \in L_{yes} \Rightarrow \exists_y : (x, y) \in A_{yes} \Rightarrow \exists_y : Q(x, y) = 1$$
$$x \in L_{no} \Rightarrow \forall_y : (x, y) \in A_{no} \Rightarrow \forall_y : Q(x, y) = 0$$

We can construct a quantum verification procedure with classical non-determinism, M, that takes the classical witness, y, and runs the circuit Q on (x, y). Then if the correct y is provided for elements of L_{yes}, M will accept with certainty, and for elements of L_{no} every witness is rejected with certainty. So $(L_{yes}, L_{no}) \in$ promise-QCNP.

Now let $(L_{yes}, L_{no}) \in$ promise-QCNP. Then there is some quantum verification procedure for which there exist classical witnesses causing the procedure to accept yes-instances with certainty, and every witness will cause rejection for no-instances. Call the verifier's part of this procedure Q_V and note that it runs on a pair of inputs, the input string x and a classical proof, call it y. Then there is a promise problem (A_{yes}, A_{no}) such that Q_V is EQP-consistent with (A_{yes}, A_{no}) and where

$$x \in L_{yes} \Rightarrow \exists_y : (x, y) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \forall_y : (x, y) \in A_{no}$$

So, $(A_{yes}, A_{no}) \in$ promise-EQP and $(L_{yes}, L_{no}) \in \exists$·promise-EQP. ∎

We will now switch to considering a base class of promise-BQP instead of promise-EQP. We will apply the quantifiers to promise-BQP to add non-determinism to the quantum model. Again the non-determinism considered here is classical, and so the classes here are not the most commonly thought of analogues of the classical classes. First we consider an extension of MA called QCMA that can be created by adding classical non-determinism to bounded error quantum machines. The proof is very similar to the one above for promise-QCNP, only with changes to make up for the switch from the exact setting to bounded error.

Theorem 7 (Marriott) $\exists$·*promise*-BQP = *promise*-QCMA

Proof:

Let $(L_{yes}, L_{no}) \in \exists$·promise-BQP. Then there is some polynomial sized quantum circuit family, Q, that is BQP-consistent with a promise problem (A_{yes}, A_{no}) such that:

$$x \in L_{yes} \Rightarrow \exists_y : (x,y) \in A_{yes} \Rightarrow \exists_y : Pr[Q(x,y) = 1] \geq \frac{2}{3}$$
$$x \in L_{no} \Rightarrow \forall_y : (x,y) \in A_{no} \Rightarrow \forall_y : Pr[Q(x,y) = 1] \leq \frac{1}{3}$$

We can use Q to design a quantum verification procedure with classical messages for the promise problem (L_{yes}, L_{no}). On input x Merlin will send some string y and Arthur will run Q on (x, y) and answer as it does. If $x \in L_{yes}$ then there is a string y that will cause Q to accept with probability at least $\frac{2}{3}$. If $x \in L_{no}$ then there is no string that will cause Q to accept with probability greater than $\frac{1}{3}$. Thus $(L_{yes}, L_{no}) \in$

promise-QCMA.

Let $(L_{yes}, L_{no}) \in$ promise-QCMA. Then there is some polynomial-sized quantum verification procedure with classical messages, Q, that is QCMA-consistent with (L_{yes}, L_{no}). Let Q_A be Arthur's part of the verification procedure that acts on the input and a classical string provided by Merlin. Then:

$$x \in L_{yes} \Rightarrow \exists_y : Pr[Q_A(x,y) = 1] \geq \frac{2}{3}$$
$$x \in L_{no} \Rightarrow \forall_y : Pr[Q_A(x,y) = 1] \leq \frac{1}{3}$$

We can see that Q_A is BQP-consistent with a promise problem $(A_{yes},\ A_{no})$. So $(A_{yes}, A_{no}) \in$ promise-BQP, and so $(L_{yes}, L_{no}) \in \exists$·promise-BQP. ■

We will show that that adding $\exists^+$ to $\exists$·promise-BQP leads us to the anticipated promise-*QCAM*. Again this is done by showing that machines of one kind can be used to produce machines of the other kind. However since the probability of error can be caused by many factors we must carefully calculate the error of the new machines to ensure that they do in fact decide the same promise problem.

Theorem 8 (Marriott) $\exists^+\exists$·*promise*-BQP $=$ *promise*-QCAM

Proof: Let $(L_{yes}, L_{no}) \in \exists^+\exists$·promise-BQP. Then there will be a polynomial sized uniformly generated quantum circuit family, Q, that is BQP-consistent with a promise problem (A_{yes}, A_{no}), with error ϵ such that:

$$x \in L_{yes} \Rightarrow \exists^{+}_{y}\exists_{z} : (x,y,z) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \exists^{+}_{y}\forall_{z} : (x,y,z) \in A_{no}$$

We will show that $(L_{yes}, L_{no}) \in$ promise-QCAM. Let M be a quantum verification procedure that does the following. First, Arthur flips $p(|x|)$ coins, y, and sends them to Merlin. Then Arthur runs Q on the input (x, y, z) where z is the proof returned by Merlin. Below, the first probability is taken with respect to the random choice of y and where $z_{x,y}$ is assumed to be Merlin's maximally accepting proof, thus it is dependent on the strings x and y. The second probability incorporates the probability of error associated with the quantum circuit family Q.

$$\begin{aligned}
x \in L_{yes} &\Rightarrow Pr[(x,y,z_{x,y}) \in A_{yes}] \geq \frac{1}{2} + \delta \\
&\Rightarrow Pr[M(x) = 1] \geq (1-\epsilon)(\frac{1}{2} + \delta) \geq \frac{1}{2} + \delta - \epsilon \\
x \in L_{no} &\Rightarrow Pr[(x,y,z_{x,y}) \in A_{yes}] \leq \frac{1}{2} - \delta \\
&\Rightarrow Pr[M(x) = 1] \leq \frac{1}{2} - \delta + \epsilon(\frac{1}{2} + \delta) \leq \frac{1}{2} - \delta + \epsilon
\end{aligned}$$

We can reduce the error in Q so that $\epsilon < \delta$, so $(L_{yes}, L_{no}) \in$ promise-QCAM.

Let $(L_{yes}, L_{no}) \in$ promise-QCAM. We will show that $(L_{yes}, L_{no}) \in \exists^{+}\exists$·promise-BQP. Let Q be a quantum verification procedure that is QCAM-consistent with (L_{yes}, L_{no}) with error ϵ. We will create M a quantum circuit family that runs on input (x, y, z). M will treat the input as though x is the input, y is a random string, and z is a potential witness and it will run the final verification stage of Q on input

(x, y, z) and answer as it does. Consider M when run on $(x, y, z_{x,y})$ where $z_{x,y}$ is the maximally accepting proof based on x and y, where x is any string and y is a random string.

$$\begin{aligned} x \in L_{yes} &\Rightarrow Pr[Q_A(x, y, z_{x,y}) = 1] \geq 1 - \epsilon \\ &\Rightarrow Pr[M(x, y, z_{x,y}) = 1] \geq 1 - \epsilon \\ &\Rightarrow \sum_{y \in \Sigma^{p(|x|)}} Pr[M(x, y, z_{x,y}) = 1] \geq (1 - \epsilon)2^{p(|x|)} \end{aligned}$$

We want to know for how many strings y that $Pr[M(x, y, z_{x,y})] > \frac{1}{2}+\delta$. This number is maximized when every y for which this holds does so with probability 1 and the rest do not with probability exactly $\frac{1}{2}+\delta$. This is at least $\frac{(1-\epsilon)-(\frac{1}{2}+\delta)}{\frac{1}{2}-\delta}2^{p(|x|)} \geq \left(\frac{1}{2} + \delta'\right) 2^{p(|x|)}$ values of y when $0 < \epsilon < \frac{1}{2}(\frac{1}{2} - \delta)$.

$$\begin{aligned} x \in L_{no} &\Rightarrow Pr[Q_A(x, y, z_{x,y}) = 1] \leq \epsilon \\ &\Rightarrow Pr[M(x, y, z_{x,y}) = 1] \leq \epsilon \\ &\Rightarrow \sum_{y \in \Sigma^{p(|x|)}} Pr[M(x, y, z_{x,y}) = 1] \leq \epsilon 2^{p(|x|)} \end{aligned}$$

We now want to know how many values of y for which $Pr[M(x, y, z_{x,y})] < \frac{1}{2} - \delta$. So for values of y that fail this we will use as little probability as necessary to underestimate this number. For at most $\frac{\epsilon}{\frac{1}{2}-\delta}2^{p(|x|)} \leq \left(\frac{1}{2} - \delta'\right) 2^{p(|x|)}$ values of y this fact fails if $0 < \epsilon < \frac{1}{2}$.

We choose ϵ so both of the conditions above hold. This means that there are at least $\frac{1}{2} + \delta'$ values of y that are good (for which there exists a proof) when the input is in L_{yes} and there are at most $\frac{1}{2} - \delta'$ bad (for which the proof may not be bad) values of y if the input is in L_{no}. We conclude that M is BQP-consistent with a promise problem (A_{yes}, A_{no}) such that:

$$x \in L_{yes} \Rightarrow \exists_y^+ \exists_z : (x, y, z) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \exists_y^+ \forall_z : (x, y, z) \in A_{no}$$

so $(L_{yes}, L_{no}) \in \exists^+\exists$·promise-BQP. ■

3.4 Quantum Sets

Let us now turn our discussion to an extension of classical languages that takes advantage of quantum information. A classical language is often spoken of as a (potentially infinite) set of finite classical strings. The obvious extension of this concept is a quantum language or a (potentially infinite) set of finite quantum superpositions. To make the idea of a quantum language meaningful we consider the superpositions to be over elements of Σ^* but any given superposition is limited in that it must only be a superposition over strings of the same length. This idea is defined below.

Definition 32 *A* quantum set *is a set of superpositions of states of Σ^* where each superposition has a specific length. Furthermore we will say a quantum set is* computable *if there exists some quantum circuit that accepts elements of the set with* *probability at least $\frac{1}{2}$ and accepts non-elements with probability less than $\frac{1}{2}$.*

We can build classes of quantum sets similar to classical complexity classes by restricting the power of the quantum circuits that compute them. As a first step we may wish to define the class of quantum sets BQP by only considering quantum sets that have polynomial sized circuits that compute the set with bounded error. This restriction works as expected for classical sets, but for quantum sets there is an interesting side effect. We note that for a given set, if that set has any input length with some superpositions of that length in the set and some not in the set then the set is not in this class because the bounded error restriction can not be met.

To see this consider two superpositions $|\psi_{yes}\rangle$ and $|\psi_{no}\rangle$. If $|\psi_{yes}\rangle$ is a member of the quantum set then it is accepted by some quantum circuit with error greater than $\frac{2}{3}$. If $|\psi_{no}\rangle$ is not a member of the quantum set then it is rejected by the same quantum circuit with error greater than $\frac{2}{3}$. Now consider a linear combination (or rather superposition) of these two states $\alpha|\psi_{yes}\rangle + \beta|\psi_{no}\rangle$. We can choose values of α and β so that the quantum circuit when run on this superposition will accept with probability exactly $\frac{1}{2}$ and so the quantum circuit could not decide the quantum set with bounded error. This side effect is due to the continuous nature of quantum superpositions in contrast to the discrete nature of classical strings.

However, because of this interesting side effect, when considering quantum sets it is natural to consider quantum promise problems. A quantum promise problem is defined in an analogous way to promise problems already considered. If we have two disjoint quantum sets, A_{yes} and A_{no}, then we can say that (A_{yes}, A_{no}) is a quantum promise problem. As in the classical study of promise problems the alternate definition is equivalent.

We note that by considering the class promise-BQP of quantum sets we avoid the side effect we encountered when we considered the class BQP of quantum sets because we need not consider the problematic superpositions that exist between members of the set and strictly defined non-members. However we encounter a second problem when considering quantum sets and bounded error computation. The quantum nature of the input poses a problem similar to the one studied in the last section when considering quantum certificates. Bounded error classes of quantum sets do not share the robustness feature of their classical counterparts.

It should be clear that the standard amplification by repetition is ruled out because quantum information cannot be copied. The procedure used in the last section will also not be sufficient, since it works because we are only interested in the performance of the maximally accepting input. Consider a promise problem $L_{yes} = \{|\psi\rangle|$ the first bit of $|\psi\rangle$ is 1 with probability greater than $\frac{2}{3}\}$ and $L_{no} = \{|\psi\rangle|$ the first bit of $|\psi\rangle$ is 1 with probability less than $\frac{1}{3}\}$. There exists a simple circuit that decides this promise problem with error bound $\frac{1}{3}$ by observing the first bit and answering what is seen. The error of this circuit cannot be amplified to be exponentially small because that would mean developing a procedure to distinguish the two states $|\psi_1\rangle = \frac{\sqrt{2}}{\sqrt{3}}|1\rangle + \frac{1}{\sqrt{3}}|0\rangle$ and $|\psi_2\rangle = \frac{1}{\sqrt{3}}|1\rangle + \frac{\sqrt{2}}{\sqrt{3}}|0\rangle$ with probability greater than $\frac{1}{2}|||\psi_1\rangle\langle\psi_1| - |\psi_2\rangle\langle\psi_2|||_{tr}$. This value is a constant so we are assured that this language cannot be decided with arbitrary precision. So when considering classes of quantum sets with bounded error we recognize that we have a hierarchy of classes corresponding to different error bounds. We will be careful to note this fact as it becomes important in the discussions of the next section.

Some may wish to distinguish between the complexity classes or promise classes that only contain classical sets, and those that contain quantum sets. However, to avoid cluttering the next section with extra notation we will use the same names. This will be unimportant except when considering bounded error classes, and then we will take extra care everything works as expected.

3.5 Quantum Quantifiers

To link classical languages to quantum sets we can consider a quantum quantifier with characteristics similar to the classical quantifiers. We will define a quantum quantifier $\exists_Q$ that is an operator that is applied to class of quantum promise problems. Quantum quantifiers will be shown to characterize quantum non-determinism. A quantum quantifier is taken with respect to polynomial sized quantum states rather than polynomial length classical states or rather strings. These quantifiers are the obvious extension of the classical quantifiers and their application to base classes characterized by quantum models of computation lead to natural classes. Of course we see that if these quantifiers are applied to classical classes there is no benefit beyond the classical quantifiers.

Definition 33 *We will say that $(L_{yes}, L_{no}) \in \exists_Q$·promise-C if and only if:*

$$x \in L_{yes} \Rightarrow \exists_{|\psi\rangle} : (x, |\psi\rangle) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \forall_{|\psi\rangle} : (x, |\psi\rangle) \in A_{no}$$

for quantum states $|\psi\rangle$ of polynomial length in x, and for some promise problem

$(A_{yes}, A_{no}) \in$ *promise-$\mathcal{C}$.*

Definition 34 *We will say that $(L_{yes}, L_{no}) \in \forall_Q$·promise-$\mathcal{C}$ if and only if:*

$$x \in L_{yes} \Rightarrow \forall_{|\psi\rangle} : (x, |\psi\rangle) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \exists_{|\psi\rangle} : (x, |\psi\rangle) \in A_{no}$$

for quantum states $|\psi\rangle$ of polynomial length in x, and for some promise problem $(A_{yes}, A_{no}) \in$ promise-C.

We notice that these quantum quantifiers exist in many of the same relationships as their classical brethren. Below we prove this for the relationships considered previously.

Theorem 9 (Marriott) *co-$\exists_Q$·promise-$\mathcal{C}$ $= \forall_Q$·promise-$\mathcal{C}$ for any $\mathcal{C}$ that is closed under complement.*

Proof: Let $(L_{yes}, L_{no}) \in \exists_Q$·promise-$\mathcal{C}$. Let $(A_{yes}, B_{no}) \in$ promise-$\mathcal{C}$ be the implied quantum promise problem associated with (L_{yes}, L_{no}) and let Q be a circuit family that is $\mathcal{C}$-consistent with (A_{yes}, A_{no}). We will show that $(L_{no}, L_{yes}) \in \forall_Q$·promise-$\mathcal{C}$. Let $\overline{Q}$ be the circuit family that runs Q then answers the opposite. Then since $\mathcal{C}$ is closed under complement $\overline{Q}$ is $\mathcal{C}$-consistent with (A_{no}, A_{yes}). So $(A_{no}, A_{yes}) \in$ promise-$\mathcal{C}$. Now:

$$x \in L_{no} \Rightarrow \forall_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{no}$$
$$x \in L_{yes} \Rightarrow \exists_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{yes}$$

so $(L_{no}, L_{yes}) \in \forall_Q$·promise-C.

For the other direction we can show that co-$\forall_Q$·promise-C $\subseteq \exists_Q$·promise-C which follows by an identical argument. ∎

Now we will apply the new quantum quantifiers in the same way that we have applied the classical quantifiers. These will lead to the complete quantum analogues of the classical classes, where not only is quantum information processing allowed, but now the non-determinism is quantum as well. First we will look at the exact setting and apply our quantum quantifier to the base class of promise-EQP and show that this characterizes quantum non-determinism. The proof follows the familiar pattern, and is similar to the ones above except for changes made to accommodate the quantum sets and the quantum non-determinism. We also note here that promise-EQP with quantum sets can be thought of as a special member of the bounded error promise-BQP hierarchy.

Theorem 10 (Marriott) $\exists_Q$·*promise*-EQP = *promise*-QNP

Proof: Let $(L_{yes}, L_{no}) \in \exists_Q$·promise-EQP. Then there is some polynomial sized quantum circuit family, Q, that is EQP-consistent with a quantum promise problem (A_{yes}, A_{no}) such that:

$$x \in L_{yes} \Rightarrow \exists_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{yes} \Rightarrow \exists_{|\Psi\rangle} : Q(x, |\Psi\rangle) = 1$$

$$x \in L_{no} \Rightarrow \forall_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{no} \Rightarrow \forall_{|\Psi\rangle} : Q(x, |\Psi\rangle = 0$$

We can construct a quantum verification procedure with quantum non-determinism, M, that takes the quantum witness, $|\Psi\rangle$, and runs the circuit Q on $(x, |\Psi\rangle)$. Then if the correct $|\Psi\rangle$ is provided for elements of L_{yes} M will accept with certainty, and for elements of L_{no} every witness is rejected with certainty. So $(L_{yes}, L_{no}) \in$ promise-QNP.

Now let $(L_{yes}, L_{no}) \in$ promise-QNP. Then there is some quantum verification procedure for which there exist quantum witnesses causing the procedure to accept "yes"-instances with certainty, and every witness will cause rejection for "no"-instances. Call the verifier's part of this procedure Q_V. Then there is a quantum promise problem (A_{yes}, A_{no}) such that Q_V is EQP-consistent with (A_{yes}, A_{no}) and where:

$$x \in L_{yes} \Rightarrow \exists_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \exists_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{no}$$

So, $(A_{yes}, A_{no}) \in$ promise-EQP and $(L_{yes}, L_{no}) \in \exists_Q$·promise-EQP. ■

We now consider the remainder of the bounded error cases. We will apply the new quantum existential quantifier to the base class of promise-BQP with quantum promise problems and show that it characterizes the class promise-QMA. This can be considered from two angles, and our proof will demonstrate them both. First if the quantum promise problems are only considered in the base class promise-BQP, and $\exists_Q$·promise-BQP and promise-QMA are considered as classical promise classes only, then it does not matter what error bound we choose for the base class promise-BQP since the classical promise classes are robust themselves. Secondly we can consider when quantum sets are allowed in all the classes considered. In this case it will be clear from the proof that the error bound must be chosen to be the same for all classes if the theorem is to be true. The proof is identical to the one above regarding promise-QNP with modifications only to make up for the bounded error, for which we use the bound $\frac{1}{3}$ arbitrarily.

Theorem 11 (Marriott) $\exists_Q$·*promise*-BQP $=$ *promise*-QMA

Proof:

Let $(L_{yes}, L_{no}) \in \exists_Q$·promise-BQP. Then there is some polynomial sized quantum circuit family, Q, that is BQP-consistent with a quantum promise problem (A_{yes}, A_{no}) such that:

$$\begin{aligned} x \in L_{yes} &\Rightarrow \exists_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{yes} \\ &\Rightarrow \exists_{|\Psi\rangle} : Pr[Q(x, |\Psi\rangle) = 1] \geq \frac{2}{3} \\ x \in L_{no} &\Rightarrow \forall_{|\Psi\rangle} : (x, |\Psi\rangle) \in A_{no} \\ &\Rightarrow \forall_{|\Psi\rangle} : Pr[Q(x, |\Psi\rangle) = 1] \leq \frac{1}{3} \end{aligned}$$

We can use Q to design a quantum verification procedure with quantum messages for the promise problem (L_{yes}, L_{no}). On input x Merlin will send some quantum state $|\Psi\rangle$ and Arthur will run Q on $(x, |\Psi\rangle)$ and answer as it does. If $x \in L_{yes}$ then there is a quantum state $|\Psi\rangle$ that will cause Q to accept with probability at least $\frac{2}{3}$. If $x \in L_{no}$ then there is no quantum state that will cause Q to accept with probability greater than $\frac{1}{3}$. Thus $(L_{yes}, L_{no}) \in$ promise-QMA.

Let $(L_{yes}, L_{no}) \in$ promise-QMA. Then there is some polynomial-sized quantum verification procedure with quantum messages, Q, that is QMA-consistent with (L_{yes}, L_{no}). Let Q_A be Arthur's part of the verification procedure that acts on the input and a quantum state provided by Merlin. Then:

$$x \in L_{yes} \Rightarrow \exists_{|\Psi\rangle} : Pr[Q_A(x, |\Psi\rangle) = 1] \geq \frac{2}{3}$$
$$x \in L_{no} \Rightarrow \forall_{|\Psi\rangle} : Pr[Q_A(x, |\Psi\rangle) = 1] \leq \frac{1}{3}$$

We can see that Q_A is BQP-consistent with a quantum promise problem (A_{yes}, A_{no}). So $(A_{yes}, A_{no}) \in$ promise-BQP, and so $(L_{yes}, L_{no}) \in \exists_Q$·promise-BQP. ■

Our final theorem of this section is perhaps the one of greatest interest because in it we combine the classical quantifiers with the new quantum quantifiers to characterize another well studied class. The class promise-QAM can be characterized by adding only classical randomness to the base class promise-QMA. Again this proof is similar to one from earlier in the section dealing only with classical quantifiers.

Again we must consider that the base class promise-BQP is not robust. As above if we only considered the quantum sets in the base class then the choice of error bound is not relevant and we use ϵ arbitrarily. Different from above however, the proof below does not imply that the hierarchy of bounded error classes coincides if quantum promise problems are considered for all classes. This is because the robustness is an important part of the following proof. Thus the alternate interpretation of this theorem, where quantum promise problems are considered remains open.

Theorem 12 (Marriott) $\exists^+\exists_Q$·*promise-BQP = promise-QAM*

Proof: Let $(L_{yes}, L_{no}) \in \exists^{+}\exists_{Q}$·promise-BQP. Then there will be a polynomial sized uniformly generated quantum circuit family, Q, that is BQP-consistent (error ϵ) with a quantum promise problem (A_{yes}, A_{no}) such that:

$$x \in L_{yes} \Rightarrow \exists_y^{+}\exists_{|\Psi\rangle} : (x, y, |\Psi\rangle) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \exists_y^{+}\forall_{|\Psi\rangle} : (x, y, |\Psi\rangle) \in A_{no}$$

We will show that $(L_{yes}, L_{no}) \in$ promise-QAM. Let M be a quantum verification procedure that does the following. First Arthur flips $p(|x|)$ coins, y, and sends them to Merlin. Then Arthur runs Q on the input $(x, y, |\Psi_{x,y}\rangle)$ where $|\Psi_{x,y}\rangle$ is the quantum proof provided by Merlin. Below, the first probability is taken with respect to the random choice of y and where $|\Psi_{x,y}\rangle$ is assumed to be Merlin's maximally accepting proof, thus it is dependent on the strings x and y, which is indicated by the subscripts. The later probability incorporates the probability of error associated with the quantum circuit family Q.

$$\begin{aligned} x \in L_{yes} &\Rightarrow Pr[(x, y, |\Psi_{x,y}\rangle) \in A_{yes}] \geq \frac{1}{2} + \delta \\ &\Rightarrow Pr[M(x) = 1] \geq (1 - \epsilon)(\frac{1}{2} + \delta) \geq \frac{1}{2} + \delta - \epsilon \\ x \in L_{no} &\Rightarrow Pr[(x, y, |\Psi_{x,y}\rangle) \in A_{yes}] \leq \frac{1}{2} - \delta \\ &\Rightarrow Pr[M(x) = 1] \leq \frac{1}{2} - \delta + \epsilon(\frac{1}{2} + \delta) \leq \frac{1}{2} - \delta + \epsilon \end{aligned}$$

Since δ is chosen arbitrarily, $(L_{yes}, L_{no}) \in$ promise-QAM.

Let $(L_{yes}, L_{no}) \in$ promise-QAM. We will show that $(L_{yes}, L_{no}) \in \exists^{+}\exists_{Q}$·promise-BQP. Let Q be a quantum verification procedure that is QAM-consistent with (L_{yes}, L_{no}) with error δ. We will create M a quantum circuit family that runs on the quantum input $(x, y, |\Psi\rangle)$. M will treat the input as though x is the input, y is a random classical string, and $|\Psi\rangle$ is a potential quantum witness and it will run the verification stage of Q, call it Q_A, on input $(x, y, |\Psi\rangle)$ and answer as it does.

$$\begin{aligned}
x \in L_{yes} \quad &\Rightarrow \quad Pr[Q_A(x, y, |\Psi_{x,y}\rangle) = 1] \geq 1 - \delta \\
&\Rightarrow \quad Pr[M(x, y, |\Psi_{x,y}\rangle) = 1] \geq 1 - \delta \\
&\Rightarrow \quad \sum_{y \in \Sigma^{p(|x|)}} Pr[M(x, y, |\Psi_{x,y}\rangle) = 1] \geq (1 - \delta) 2^{p(|x|)}
\end{aligned}$$

We want to know for how many strings y that $Pr[M(x, y, |\Psi_{x,y}\rangle)] > 1 - \epsilon$. This number is maximized when every y for which this holds does so with probability 1 and the rest do not with probability exactly $1 - \epsilon$. This is at least $\frac{\epsilon - \delta}{\epsilon} 2^{p(|x|)} \geq \left(\frac{1}{2} + \delta'\right) 2^{p(|x|)}$ values of y for $0 < \delta < \frac{\epsilon}{2}$.

$$\begin{aligned}
x \in L_{no} \quad &\Rightarrow \quad Pr[Q_A(x, y, |\Psi_{x,y}\rangle) = 1] \leq \delta \\
&\Rightarrow \quad Pr[M(x, y, |\Psi_{x,y}\rangle) = 1] \leq \delta \\
&\Rightarrow \quad \sum_{y \in \Sigma^{p(|x|)}} Pr[M(x, y, |\Psi_{x,y}\rangle) = 1] \leq \delta 2^{p(|x|)}
\end{aligned}$$

We now want to know how many values of y for which $Pr[M(x, y, |\Psi_{x,y}\rangle)] < \epsilon$. So for values of y that fail this we will use as little probability as necessary to underestimate this number. For at most $\frac{\delta}{\epsilon} 2^{p(|x|)} \leq \left(\frac{1}{2} - \delta'\right) 2^{p(|x|)}$ values of y this fact holds if $0 < \delta < \frac{\epsilon}{2}$.

Since the error δ can be made smaller than any constant we can conclude that M is BQP-consistent (error ϵ) with a promise problem (A_{yes}, A_{no}) such that:

$$x \in L_{yes} \Rightarrow \exists_y^+ \exists_{|\Psi\rangle} : (x, y, |\Psi\rangle) \in A_{yes}$$
$$x \in L_{no} \Rightarrow \exists_y^+ \forall_{|\Psi\rangle} : (x, y, |\Psi\rangle) \in A_{no}$$

so $(L_{yes}, L_{no}) \in \exists^+ \exists_Q$·promise-BQP. ∎

Chapter 4

Quantum Oracle Separations

We now turn our attention to relativised separation, in an attempt to see if these relationships as well are inherited from the classical classes. Relativised separation is a weak type of separation used by complexity theorists in the stead of strong separation proofs that are notoriously difficult to come by except in a few rare cases. In fact the study of relativised separation was first used to better understand the relationship between the paradigmatic classes P and NP.

A relativised separation, or oracle separation, is a separation that is true when all machines are given access to the same oracles. It was soon recognized that oracle separations were a weaker separation when it was discovered that not only was there an oracle that would separate P from NP, but also one that made them equal. Despite not solving the famous problem, oracle results are often pursued because they are some indication of the difficulty of proving their equality. When two classes can be separated by an oracle it means that "black box" attempts at simulation will not be successful, and a new strategy that does not "relativise" must be found. In the absence of the new strategy the oracle separation can be interpreted as "proving these classes are equal requires a novel approach." Oracle separations are proved by constructing, step by step, an oracle that will fool one class of oracle machines, but be solvable by another. This requires defining an oracle dependent problem with the specific structure to separate the powers of the two classes. Sometimes these problems are simple, we will use common problems like OR and MAJORITY, and some are exceedingly complex.

4.1 Prior Work in Oracle Separations

Below we state known oracle separations for the classical classes we are studying. We use the notations $\mathcal{C}^A$ to denote the class $\mathcal{C}$ relative to an oracle A.

Theorem 13 (Baker, Gill, Solovay[4]) *For each of the following relations there exists an oracle, A, relative to which the relation holds:*

1. $\mathrm{P}^A = \mathrm{NP}^A$
2. $\mathrm{P}^A \subsetneq \mathrm{NP}^A$
3. $\mathrm{NP}^A \neq co - \mathrm{NP}^A$

Following this paper, many theorists explored what other results could be demonstrated through the use of relativization. Later the probabilistic complexity classes were studied by Rackoff [16] as well as many others. The following relationships between the probabilistic classes were demonstrated.

Theorem 14 *There exists an oracle, A, relative to which the all of these relations holds:*

1. $co - \mathrm{R}^A \not\subseteq \mathrm{NP}^A$
2. $\mathrm{BPP}^A \not\subseteq \mathrm{NP}^A$
3. $\mathrm{R}^A \neq co - \mathrm{R}^A$

Here R is the one-sided probabilistic class and is used because it is a subset of BPP and will generate corollaries for all of its super-classes, like the last two relations in this theorem. The proof of the first result provides an oracle dependent language and an oracle where the language is in the class co-R^A but is not in the class NP^A. The first corollary follows since co-R $\subseteq$ BPP. The second corollary follows from the fact that the set that is in co-R^A is not in NP^A and since R^A is in NP^A the set must not be in R^A either. Furthermore other corollaries can be shown.

Corollary 2 *There exists an oracle, A, relative to which both of the following hold*

1. $P^A \subsetneq R^A \subsetneq BPP^A$
2. $P^A \subsetneq co\text{-}R^A \subsetneq BPP^A$

To complete the study of these simple classes the following relationship was also demonstrated.

Theorem 15 *There exists an oracle, A, relative to which* $NP^A \not\subseteq BPP^A$.

It is worth noting that the combination of these two oracle results provide worlds in which there are sets in BPP^A that are not in NP^A and vice versa. These theorems and their respective corollaries settle all the relativized pairwise inclusion relationships that are consistent with known inclusions among the classes, P, R, co-R, NP, and BPP. Of course more complex relationships can and have been established dealing with the classes discussed. For a simple presentation of these theorems and their proofs see Balcazar, Diaz, and Gabarro [6].

Moving into the realm of quantum complexity theory one would be motivated to ask what relationships can be shown regarding the most commonly discussed classes

EQP and BQP. These classes have also been studied extensively in the relativized setting.

Theorem 16 *For each separation below it has been demonstrated that there exists an oracle, A, relative to which the separation holds:*

1. $\mathrm{P}^A \subsetneq \mathrm{EQP}^A$
2. $\mathrm{BPP}^A \subsetneq \mathrm{BQP}^A$

The first separation in this theorem comes via an adaptation of the Deutsch-Josza problem cast as an oracle dependent language by Berthiume and Brassard [10]. The second separation was first demonstrated by Berstein and Vazarani [9] and used a complex problem called the recursive Fourier sampling problem. Since then simpler proofs of this result have emerged. Regarding further the class BQP we can note that another corollary can be added to the list provided above.

Corollary 3 *There exists an oracle, A, relative to which* $\mathrm{BQP}^A \not\subseteq \mathrm{NP}^A$.

This can be combined with a result of Bennett, Brassard, Bernstein, and Vazirani [8] to complete the comparison of BQP and NP.

Theorem 17 (Bennett, Brassard, Bernstein, Vazirani) *There exists an oracle, A, relative to which* $\mathrm{NP}^A \not\subseteq \mathrm{BQP}^A$.

This result is actually extended in the paper to provide a separation between NP $\cap$ co-NP and BQP. We can again notice that the same relationship that was identified between BPP and NP can be identified here with respect to the classes BQP and NP.

Some have begun to study the classes representing the combination of non-determinism and quantum machines. Watrous[18] has established the following separations.

Theorem 18 *For each separation below it has been demonstrated that there exists an oracle, A, relative to which the separation holds:*

1. $\mathrm{MA}^A \subsetneq \mathrm{QMA}^A$
2. $\mathrm{BQP}^A \not\subseteq \mathrm{MA}^A$

This final separation is also a corollary of a stronger claim, namely that EQP is not contained in MA relative to an oracle, which was claimed by Berstein and Vazarani, yet for which no proof was provided.

4.2 New Oracle Separations

We will now turn our attention back to standard complexity classes to extend and add results to the known oracle results in the field to better complete the picture the quantum classes are beginning to form. First let us revisit the concept of oracles. Below we will also consider an oracle, A, to be a set of indexes, $A \subseteq \Sigma^*$. We can see that if the oracle is a sequence of bits, then we can say that a particular index is in the oracle if the bit that it indexes is 1. Likewise if the bit is 0 then we can say that the index is not in the oracle. This way we can easily discuss concepts like the *empty oracle*, the oracle with all 0s, or adding (removing) indices to an oracle by changing the corresponding bits from 0 to 1 (1 to 0). It is also common to write the indices as binary integers although one might also use the decimal values when convenient.

We will switch now from the quantum circuit model to the quantum Turing machine model because some of the earlier result we draw upon use this model. We will also notice that the difficulties associated with programming a quantum Turing machine will not pose a problem for us in this section since we will not have to provide complex programs. An important property of quantum oracle Turing machines is that they can enter the query state with a superposition of indices written on the query tape. This will allow the machine to query many places of the oracle in one query, although the bits queried will also be in superposition and must either be carefully manipulated to retrieved the desired information, or can be observed to provide a bit chosen at random from the queried set. See [9] and [8] for a discussion of how quantum oracle Turing machines still evolve unitarily and thus obey the properties of quantum mechanics.

A concept regarding the error of an oracle quantum Turing machine will be useful for a number of the results below. Let M be a quantum oracle Turing machine, A an oracle, and x a word. We will say that M is an *ϵ-error machine under A on x* when the error probability of M operating with oracle A on word x is bounded by $\epsilon < 1/2$. To say that M is an *ϵ-error machine under A* is to say that it is an ϵ-error machine under A for every word x. It should be noticed that this definition is oracle dependent. A machine that is ϵ-error under an oracle A may have unbounded error if even one word is added to or removed from A, and in fact this is an important property that enables the proofs in the next section. Since the classes we consider are robust the choice of ϵ is arbitrary in the range $0 < \epsilon < \frac{1}{2}$.

We also note that the set of quantum oracle Turing machines is enumerable. A quantum oracle Turing machine is defined by a triplet (Σ, Q, δ) where Σ is a finite alphabet with an identified blank symbol, Q is a finite set of internal states with identified states q_o (the initial state), q_f (the final state), q_q (the query state), and q_a (the answer query state), no two of which are the same, and δ is the quantum transition function

$$\delta : Q \times \Sigma \rightarrow \tilde{\mathbb{C}}^{\Sigma \times Q \times \{L,R\}}$$

Here $\tilde{\mathbb{C}}$ is an appropriately chosen set of $\alpha \in \mathbb{C}$ that are sufficient for quantum computation. For our purposes we must limit the chosen amplitudes to those that are efficiently computable. We will note that it has been shown that the set $\{0, \pm\frac{3}{5}, \pm\frac{4}{5}, 1\}$ is sufficient[1]. The number of transition functions δ is enumerable and so the number of quantum oracle Turing machines is enumerable.

We can restrict our discussion to clocked quantum oracle Turing machines without loss of generality, which will allow us to consider an enumeration that is in increasing running time which is utilized by our proofs below. The dovetailing lemma in [9] can be used to construct a clocked quantum oracle Turing machine out of any quantum oracle Turing machine and any time constructible function expressing the running time of that machine.

The lemmas that are provided in this section are used to argue that a particular set of words exists that can be sufficiently hidden in an oracle for the purpose of diagonalizing against a set of machines. The arguments used to demonstrate these lemmas rely on results of Beals, Buhrman, Cleve, Mosca, and de Wolf [7]. In this paper the authors demonstrate lower bounds on the number of queries required to compute particular symmetric boolean functions of variables stored in an oracle. The important facts that we will use regard the symmetric boolean functions **OR** and

MAJORITY. In both cases the lower bound of the number of queries required was polynomial in the number of variables. Since we will be concerned with computing these functions over exponential sized domains the number of queries required will also be exponential. More precise results are demonstrated in the paper and are used below.

Once we have the lemma to rely upon we will use a standard proof technique to demonstrate our first oracle separation. The technique used begins by defining an oracle dependent problem that has the unique properties necessary to separate the classes considered. Once the problem is chosen we will step by step construct an oracle that will diagonalize against every machine of a particular model, by forcing each machine to err on at least one input. Thus with respect to this oracle the oracle dependent language will be in one class and not the other, effectively separating them.

Lemma 1 *Let M be a quantum oracle machine with running time $p(n)$ a polynomial. Let n_0 be chosen so that $p(n_0) < 2^{\frac{n_0-4}{2}}$. Let A be an oracle with no words of length n_0. If M is an exact (0-error) machine with respect to A and the input 0^{n_0} and if M accepts input 0^{n_0} with oracle A, then there exists a non-empty set of words $B \subseteq 1 \cdot \Sigma^{n_0-1}$, such that M is either not an exact (0-error) machine with respect to the oracle $A \cup B$ and the input 0^{n_0} or M accepts input 0^{n_0} with oracle $A \cup B$.*

Proof: Assume for the sake of a contradiction that there does not exist such a B. Then for every such B the machine M is an exact machine and it rejects 0^{n_0}. If this is the case then we can construct an exact quantum oracle machine that can compute the bitwise or of 2^{n_0-1} oracle entries with fewer than $p(n_0) < 2^{\frac{n_0-4}{2}}$ queries to the oracle. Beals et. al. [7] have demonstrated that this is not possible. Therefore there must exist at least one set B such that M is either not an exact machine with respect to the oracle $A \cup B$ and the input 0^{n_0} or M accepts input 0^{n_0} with oracle $A \cup B$. ∎

The result presented below shows that one of the common properties of non-determinism carries over into the quantum setting. That is, the question of whether NP is closed under complement is almost as primary as the question of whether P = NP. Since there is an oracle that separates NP from its complement we believe that demonstration of this particular closure to be very difficult. We extend those oracle results to the quantum non-deterministic classes to show that this relationship is constant in the new quantum model as well.

Theorem 19 (Marriott) *There exists an oracle, A, such that* $\mathrm{QNP}^A \neq$*co-*QNP^A.

Proof: Consider the oracle dependent language

$$L(A) = \{0^n | \exists_{w,|w|=n}, A(w) = 1\}$$

The language $L(A) \in NP^A \subseteq \mathrm{QNP}^A$ for all oracles A. We will construct an oracle A such that $L(A) \notin$co-EQNP^A or equivalently $\overline{L(A)} \notin \mathrm{QNP}^A$.

Consider the construction given in Figure 4.1. First we must demonstrate that the construction can indeed be performed. First we should note that in step n there always exists a $k(n)$ that meets the conditions. Secondly the lemma assures us that the set of words $B(n)$ exists if needed.

Now we argue that the construction provides us with an oracle, A, for which

```
Step 0:
    1. $k(0) = 0$
    2. $A = \phi$
Step $n$:
    1. let $k(n)$ be the smallest number such that $p_n(k(n)) < 2^{\frac{k(n)-4}{2}}$ and $p_{n-1}(k(n-1)) < k(n)$
    2. if $0^{k(n)} \in L(M_n, A)$ then
        (a) fix a computation of $M_n$ on $0^{k(n)}$ and $A$ that accepts
        (b) let $B(n)$ be the set of words guaranteed by the Lemma and let $A = A \cup B(n)$
       else $A = A$
```

where M_n is the nth polynomial time exact non-deterministic quantum oracle machine in the standard enumeration where $p_n(i) \leq p_{n+1}(i)$ for all i and n.

Figure 4.1: Construction of an oracle A for Theorem 19

no polynomial time exact non-deterministic quantum oracle machine decides the language $L(A)$. The construction is designed to diagonalize against the ith polynomial time exact non-deterministic quantum oracle machine showing that $\overline{L(A)} \neq L(M_i, A)$. If the "then" branch is taken then M_i is either not an exact machine with respect to oracle $A \cup B(i)$ and input $0^{k(i)}$ or the computation that is fixed in the construction at step i will accept the input, thus showing that $\overline{L(A)} \neq L(M_i, A)$. If the "else" branch is taken then $\overline{L(A)} \neq L(M_i, A)$ is already true. Every such machine is ruled out by the construction so the resulting oracle A is such that $\overline{L(A)} \neq L(M_i, A)$ for any i. Therefore $L(A) \notin$co-QNP^A. ∎

We have shown that non-determinism acts as suspected in the quantum model for the exact setting. Now we do the same for the bounded error setting. We utilize the same strategy and the proofs of the following lemma and theorem are closely related to the last two presented.

Lemma 2 *Let M be a quantum oracle machine with running time $p(n)$ a polynomial. Let n_0 be chosen so that $p(n_0) < 2^{\frac{n_0-5}{2}}$. Let A be an oracle that contains no words of length n_0. If M is an ϵ-error machine with respect to A and the input 0^{n_0} and if M accepts input 0^{n_0} with oracle A, then there exists a non-empty set of words $B \subseteq 1 \cdot \Sigma^{n_0-1}$, such that M is either not an ϵ-error machine with respect to the oracle $A \cup B$ and the input 0^{n_0} or M accepts input 0^{n_0} with oracle $A \cup B$.*

Proof: Assume for the sake of a contradiction that there does not exist such a B. Then for every such B the machine M is an ϵ-error machine and it rejects 0^{n_0}. If this is the case then we can construct an ϵ-error quantum oracle machine that can compute the bitwise or of 2^{n_0-1} oracle entries with fewer than $p(n_0) < 2^{\frac{n_0-5}{2}}$ queries

Step 0:

1. $k(0) = 0$
2. $A = \phi$

Step n:

1. let $k(n)$ be the smallest number such that $p_n(k(n)) < 2^{\frac{k(n)-5}{2}}$ and $p_{n-1}(k(n-1)) < k(n)$
2. **if** $0^{k(n)} \in L(M_n, A)$ **then**
   (a) fix a computation of M_n on $0^{k(n)}$ and A that **accepts**
   (b) let $B(n)$ be the set of words guaranteed by the Lemma and let $A = A \cup B(n)$

   else $A = A$

where M_n is the nth polynomial time ϵ-error non-deterministic quantum oracle machine in the standard enumeration where $p_n(i) \leq p_{n+1}(i)$ for all i and n.

Figure 4.2: Construction of an oracle A for Theorem 20

to the oracle. Beals et. al. [7] have demonstrated that this is not possible. Therefore there must exist at least one set B such that M is either not an ϵ-error machine with respect to the oracle $A \cup B$ and the input 0^{n_0} or M accepts input 0^{n_0} with oracle $A \cup B$. ■

Theorem 20 (Marriott) *There exists an oracle, A, such that* $\text{QMA}^A \neq co-\text{QMA}^A$.

Proof: Consider the oracle dependent language

$$L(A) = \{0^n | \exists_{w,|w|=n}, A(w) = 1\}$$

The language $L(A) \in \mathrm{NP}^A \subseteq \mathrm{QMA}^A$ for all oracles A. We will construct an oracle A such that $L(A) \notin co-\mathrm{QMA}^A$ or equivalently $\overline{L(A)} \notin \mathrm{QMA}^A$.

Consider the construction given in Figure 4.2. First we must demonstrate that the construction can indeed be performed. First we should note that in step n there always exists a $k(n)$ that meets the conditions. Secondly the lemma assures us that the set of words $B(n)$ exists if needed.

Now we argue that the construction provides us with an oracle, A, for which no polynomial time ϵ-error non-deterministic quantum oracle machine decides the language $L(A)$. The construction is designed to diagonalize against the ith polynomial time ϵ-error non-deterministic quantum oracle machine showing that $\overline{L(A)} \neq L(M_i, A)$. If the "then" branch is taken then M_i is either not an ϵ-error machine with respect to oracle $A \cup B(i)$ and input $0^{k(i)}$ or the computation that is fixed in the construction at step i will accept the input, thus showing that $\overline{L(A)} \neq L(M_i, A)$. If the "else" branch is taken then $\overline{L(A)} \neq L(M_i, A)$ is already true. Every such machine is ruled out by the construction so the resulting oracle A is such that $\overline{L(A)} \neq L(M_i, A)$ for any i. Therefore $L(A) \notin co-\mathrm{QMA}^A$. ∎

Finally we will switch gears a little to show that the quantum classes are still properly contained by the largest of the polynomial time probabilistic classes in the weak sense. We use a similar method as the other proofs in this section, but we change our choice of oracle dependent problem. Now we consider the problem MAJORITY and apply it to the various lengths in the oracle.

Lemma 3 *Let M be a quantum oracle machine with running time $p(n)$ a polynomial. Let n_0 be chosen so that $p(n_0) < 2^{\frac{n_0-6}{2}}$. Let A be an oracle that contains no words* *of length n_0. If M is an ϵ-error machine with respect to A and the input 0^{n_0} and if M rejects input 0^{n_0} with oracle A, then there exists a non-empty set of words $B \subseteq 1 \cdot \Sigma^{n_0 - 1}$, such that M is either not an ϵ-error machine with respect to the oracle $A \cup B$ and the input 0^{n_0} or M accepts input 0^{n_0} with oracle $A \cup B$ and $|B| \leq 2^{n_0-2}$ or M rejects input 0^{n_0} with oracle $A \cup B$ and $|B| > 2^{n_0-2}$.*

Proof: Assume for the sake of a contradiction that there does not exist such a B. Then for every such B the machine M is an ϵ-error machine and it rejects 0^{n_0} if $|B| \leq 2^{n_0-2}$ and it accepts 0^{n_0} if $|B| > 2^{n_0-2}$. If this is the case then we can construct an ϵ-error quantum oracle machine that can compute the majority function of 2^{n_0-1} oracle entries with fewer than $p(n_0) < 2^{\frac{n_0-6}{2}}$ queries to the oracle. Beals et. al. [7] have demonstrated that this is not possible. Therefore there must exist at least one set B such that M is either not an ϵ-error machine with respect to the oracle $A \cup B$ and the input 0^{n_0} or M accepts input 0^{n_0} with oracle $A \cup B$ and $|B| \leq 2^{n_0-2}$ or M rejects input 0^{n_0} with oracle $A \cup B$ and $|B| > 2^{n_0-2}$. ∎

Theorem 21 (Marriott) *There exist an oracle, A, such that* $\mathrm{QMA}^A \subsetneq \mathrm{PP}^A$.

Proof: Consider the oracle dependent language

$$L(A) = \left\{ 0^n \,\middle|\, \sum_{|w|=n} A(w) > 2^{n-2} \right\}$$

The language $L(A) \in \mathrm{PP}^A$ for all oracles A. We will construct an oracle A such that $L(A) \notin \mathrm{QMA}^A$.

Consider the construction given in Figure 4.3. First we must demonstrate that the construction can indeed be performed. First we should note that in step n there

Step 0:

1. $k(0) = 0$
2. $A = \phi$

Step n:

1. let $k(n)$ be the smallest number such that $p_n(k(n)) < 2^{\frac{k(n)-6}{2}}$ and $p_{n-1}(k(n-1)) < k(n)$
2. **if** $0^{k(n)} \in L(M_n, A)$ **then** $A = A$
   else
   (a) fix a computation of M_n on $0^{k(n)}$ and A that **accepts**
   (b) let $B(n)$ be the set of words guaranteed by the Lemma and let $A = A \cup B(n)$

where M_n is the nth polynomial time ϵ-error non-deterministic quantum oracle machine in the standard enumeration where $p_n(i) \leq p_{n+1}(i)$ for all i and n.

Figure 4.3: Construction of an oracle A for Theorem 21

always exists a $k(n)$ that meets the conditions. Secondly the lemma assures us that the set of words $B(n)$ exists if needed.

Now we argue that the construction provides us with an oracle, A, for which no polynomial time ϵ-error non-deterministic quantum oracle machine decides the language $L(A)$. The construction is designed to diagonalize against the ith polynomial time ϵ-error non-deterministic quantum oracle machine showing that $L(A) \neq L(M_i, A)$. If the "then" branch is taken then $L(A) \neq L(M_i, A)$ is already true. If the "else" branch is taken then M_i is either not an ϵ-error machine with respect to oracle $A \cup B$ and input 0^{n_0} or the computation of M_i that is fixed at step i accepts input 0^{n_0} with oracle $A \cup B$ and $|B| \leq 2^{n_0-2}$ or rejects input 0^{n_0} with oracle $A \cup B$ and $|B| > 2^{n_0-2}$, thus showing that $L(A) \neq L(M_i, A)$. Every such machine is ruled out by the construction so the resulting oracle A is such that $L(A) \neq L(M_i, A)$ for any i. Therefore $L(A) \notin \mathrm{QMA}^A$. ■

Chapter 5

Discussion and Open Questions

The characterization of non-determinism by quantifiers is one of the favored characterizations in complexity theory because of its ease of use. Despite the fact that this characterization works well when applying these quantifiers to exact classes, we find that the characterization is questionable when we try to apply these quantifiers to bounded error classes. This problem is demonstrated by the problem of proving that $\exists\cdot$BPP = MA. Though intuitively MA is considered a non-deterministic extension of BPP and so it would seem that this relationship should hold, the quantifiers as they are currently used are not known to characterize non-determinism in these cases.

In the previous sections we discussed how we can extend the study of these quantifiers into the realm of promise problems and their associated promise classes. In effect we have defined a new set of quantifiers that not only succeed where the old ones succeeded, but also where they failed. We note that in the realm of promise classes, promise-MA = $\exists\cdot$promise-BPP = $\exists\exists^{+}\cdot$promise-P. This result shows that this extension is not susceptible to the same problems as its successor, and as such is a more precise characterization of non-determinism. Currently it is not known if this characterization fails when applied to any promise class, but it seems to succeed in all the cases that we have considered in this project.

As was noted in earlier sections the coincidence of promise classes implies the coincidence of the related complexity classes. Thus in the previous sections when we demonstrate that the quantifier definition of non-determinism in the promise class setting results in identical classes, we have also shown that using this definition of non-determinism works for the standard complexity classes as well. Thus we can consider an upgrade to the current theory of quantifiers. Below we suggest a redefinition of what it means to apply these quantifiers to a complexity class. Though notationally these quantifiers will be identical to the quantifiers used in prior work we stress that results using this definition imply nothing about the previous definition.

Definition 35 *Let Q be a quantifier, classical or quantum. We will say that a language, $L \subseteq \Sigma^*$, is in the class $Q \cdot \mathcal{C}$ if and only if $(L, \overline{L}) \in Q$·promise-$\mathcal{C}$.*

We will note that if a language is in a class defined with the old quantifiers then it will necessarily be in the class defined with the new quantifiers. This property can be thought of as a type of containment. The new quantifiers work whenever the old ones did, but they also work in other cases as well. Below we state and prove this property.

Proposition 9 *Let Q be a quantifier, classical or quantum, and let $\overline{Q}$ be its complement. If $L \in Q \cdot \mathcal{C}$ then $(L, \overline{L}) \in Q$·promise-$\mathcal{C}$.*

Proof: Let L be in $Q \cdot \mathcal{C}$. Then there is a machine, M, which operates under the model defining the class $\mathcal{C}$ for which:

$$x \in L \Rightarrow Q_y : M \text{ accepts } (x, y)$$
$$x \notin L \Rightarrow \overline{Q}_y : M \text{ rejects } (x, y)$$

We can use the same machine M for the promise problem $(L, \overline{L})$ and we see that we get:

$$x \in L \Rightarrow Q_y : M \text{ accepts } (x, y)$$
$$x \in \overline{L} \Rightarrow \overline{Q}_y : M \text{ rejects } (x, y)$$

Clearly we can see that M is $\mathcal{C}$-consistent with the implied promise problem (A_{yes}, A_{no}) and thus $(L, \overline{L}) \in Q$·promise-$\mathcal{C}$.□

Using this new definition of what it means to apply a quantifier to a complexity class we can now eliminate the problematic cases encountered with the originals. Below we state the known results regarding this new definition that are collected from a number of sources via the proposition above and from work earlier in this project. First in the classical setting we have:

Theorem 22 *1.* $\mathrm{NP} = \exists \cdot \mathrm{P}$

2. $\mathrm{BPP} = \exists^+ \cdot \mathrm{P}$

3. $\mathrm{MA} = \exists \cdot \mathrm{BPP} = \exists\exists^+ \cdot \mathrm{P}$

4. $\mathrm{AM} = \exists^+ \cdot \mathrm{MA} = \exists^+\exists \cdot \mathrm{BPP} = \exists^+\exists\exists^+ \cdot \mathrm{P}$

We have also found that the original definitions of the quantifiers when applied to the quantum classes with error either bounded or disallowed do not lead to the expected classes, due to the inherent randomness in quantum mechanics. The new definitions of these quantifiers allows us to overcome these problems and extend these results into the quantum domain.

Theorem 23 *1.* $\mathrm{QCNP} = \exists \cdot \mathrm{EQP}$

2. $\mathrm{QCMA} = \exists \cdot \mathrm{BQP}$

3. $\mathrm{QCAM} = \exists^+ \cdot \mathrm{QCMA} = \exists^+\exists \cdot \mathrm{BQP}$

We note that it is not known how to reduce the last two of these results further to the class EQP as was done in the classical setting. Several questions can be raised with respect to this that may have connections to derandomization. First we can ask if BQP $=\exists^{+}\cdot$EQP, or if these classes can be separated by an oracle. If it turns out that these classes can be separated by an oracle then perhaps there is another quantifier that can be defined, related to the $\exists^{+}$ quantifier for which this is true, or even such that Q·P = BQP. We will note that the obvious extension of $\exists^{+}$ into the quantum world, that is for most superpositions, is potentially powerless over the classical quantifier since it seems it could be simulated by constructing a purely mixed state. Thus it is an open question is if bounded error quantum computation can be reduced to deterministic quantum computation or even deterministic classical computation via an appropriately defined quantifier. Next we show the known results using the new quantum quantifiers defined in the previous sections.

Theorem 24 *1.* $\mathrm{QNP} = \exists_Q \cdot \mathrm{EQP}$

2. $\mathrm{QMA} = \exists_Q \cdot \mathrm{BQP}$

3. $\mathrm{QAM} = \exists^{+} \cdot \mathrm{QMA} = \exists^{+}\exists_Q \cdot \mathrm{BQP}$

The obvious overlap between these two definitions for quantifiers is due to the fact that a shift from languages to promise problems is minor, but in this case we find that it has made the difference. Since quantum information's infancy we have seen several significant results that have depended on a notion of promise problems, in a way because of the nature of quantum mechanics. However promise problems have been also studied in the classical setting prior to these motivations. It seems not altogether shocking that they should once again prove useful to both quantum and classical complexity theory. Upon which with our discussion of quantum sets we note that standard bounded error complexity classes are largely inferior to their promise class counterparts. Further study of these classes may lead to other breakthroughs in the field. The vast similarities of the classes $\mathcal{C}$, $\mathcal{C}$ with quantum sets, and promise-$\mathcal{C}$, invite an abuse of notation by calling all the classes, or rather their union, by the name $\mathcal{C}$. We find that in this case we already know much about the world of these unions.

Now regarding the quantum Arthur-Merlin hierarchies, it has been shown by Kitaev and Watrous[13] that the quantum message hierarchy collapses to the third level, unlike the strictly classical hierarchy that collapses to the second level. Currently it is not known if this result can be improved upon. Two primary questions arise from these results. First, it has not been determined to which level the classical message quantum hierarchy collapses. It seems logical to conclude that it must collapse to either the second or third level, but the level to which it does collapse will shed another revealing fact concerning these models. The second question is whether the results of this project in the application of the quantifiers can be used to demonstrate the collapse of these hierarchies as the original quantifiers were used for the classical Arthur-Merlin hierarchy.

Furthermore we have verified that all the basic oracle separations in the quantum world exist in the same way as the classical world. This comes perhaps as expected as the quantum world has a great tendency to act classically which is evident in the nature of how the theory evolved. However, as is also the case with the quantum world this vast similarity comes with a twist. A uniqueness of the quantum world is the apparent benefit of quantum computation in the exact setting. Classical randomness provides no benefit to a deterministic algorithm in the exact setting; however, there is evidence to suggest that quantum properties do provide such a benefit. As such we have an added layer to the error hierarchy in the quantum world. These classes exists between the classical classes P and NP, and the bounded error quantum classes BQP and QMA, in an area vacant with respect to classical randomness. An open question is whether BQP is contained in PH, or perhaps easier, a well defined QPH.

Bibliography

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

[2] D. Aharonov and T. Naveh. Quantum np - a survey. *arXiv:quant-ph/0210077*, 2002.

[3] L. Babai. Trading group theory for randomness. *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 421–429, 1985.

[4] T. Baker, J. Gill, and R. Solovay. Relativizations of the p =? np question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[5] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, 1988.

[6] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* Springer-Verlag, 1990.

[7] R. Beals, H. Burhman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 351 – 362, 1998.

[8] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510 – 1523, 1997.

[9] E. Berstein and U. Vazirai. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411 – 1473, 1997.

[10] A. Berthiaume and G. Brassard. Oracle quantum computing. *Proceedings of the 7th Annual IEEE Conference on Structure in Complexity*, 1993.

[11] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61:159–173, 1984.

[12] A. Kitaev, A. Shen, and M. Vyalyi. *Classical and quantum computation.* American Mathematical Society, 2002.

[13] A. Kitaev and J. Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. *Proceedings of the 32th Annual Symposium on Theory of Computing*, pages 608–617, 2000.

[14] C. Marriott and J. Watrous. Quantum arthur-merlin games. *Manuscript*, 2003.

[15] M. Nielsen and L. Chuang. *Quantum Computation and Quantum Information.* Cambridge Press, 2000.

[16] C. Rackoff. Relativized questions involving probabilistic algorithms. *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing*, pages 338–342, 1978.

[17] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484 – 1509, 1997.

[18] J. Watrous. Succint quantum proofs for properties of finite groups. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 537–546, 2000.

[19] S. Zachos. Probabilistic quantifiers and games. *Journal of Computer and System Sciences*, 36:433–451, 1988.