Quantum Computation

Nicholas T. Ouellette

March 18, 2002

Abstract

The theory of quantum computation, a new and promising field, is discussed. The mathematical background to the field is presented, drawing ideas both from quantum mechanics and the theory of classical computation. The quantum circuit model of computation is discussed, and a universal set of quantum logic gates is investigated. Quantum algorithms are examined, focusing particularly on Grover's database search algorithm and Shor's factoring algorithm, the two best known quantum algorithms. Finally, an implementation of quantum computation using linear optics is developed.

Contents

1 Introduction

The theory of quantum mechanics developed in the early part of the twentieth century is a theory of the very small. A very mathematical theory, it nevertheless makes predictions that agree to fantastic precision with experimentally determined results. In order to make the theory agree with the physical world, however, scientists have been forced to part with the determinism of classical physics in favor of probabilistic quantum mechanical predictions. This sacrifice of determinism has been ameliorated, however, by the identification of the quantum mechanical properties of superposition, interference, and entanglement as powerful resources. The theory of quantum computation exploits these tools to perform computation exponentially faster in some cases than is possible with any classical computer.

The theory of classical computation stands in stark contrast to quantum theory. It is not a theory describing the physical world; rather, it is purely mathematical, designed to provide a framework for studying algorithms. Much of computational theory is centered around the idea of the Turing machine, named after Alan Turing, the British mathematician who pioneered the field. A Turing machine is a simple theoretical construct that has proved useful in studying computation, and one of the great triumphs of computer science has been to prove that there is no possible computation that cannot be performed on this simple apparatus. This idea is embodied in the Church-Turing thesis, which states that the class of functions computable by a Turing machine is exactly the class of functions computable by algorithm [1]. Thus, the study of the theory of computation may be reduced to the study of the Turing machine. While the full theory of Turing machines and computation is beyond the scope of this paper, we will on occasion make use of some of the concepts involved.

On the surface, then, it would appear that the fields of classical computation and quantum mechanics have no relation to one another. Indeed, the connection between the two is subtle, and was not recognized until the 1980s. Appropriately enough, the first hints of this link grew out of information theory, a subject that also links together abstract mathematical notions with the physical world. One fundamental result from information theory is Landauer's Principle, which states that for every bit of information erased by a computing device, an amount of energy equal to $kT \ln 2$ is released into the environment, where k is Boltzmann's constant and T is the temperature of the environment [2]. As heat dissipation is generally an unwanted side effect of computation, researchers began to study the possibility of computation without erasure. This led to the development of reversible computation, where every computation may be run backwards and thus loses no information. This result in turn led some scientists to wonder if quantum mechanical systems could be used to compute, since quantum systems undergo unitary evolution, which is by its very nature reversible. Finally, in 1985, David Deutsch at Oxford University published a paper containing a full theory of a quantum Turing machine, making explicit the link between computational theory and quantum mechanics [3].

Deutsch showed that the theory of computation developed by Turing and others

is not free from physical assumptions. A classical Turing machine is assumed to be in a definite state at the conclusion of each step of computation. But a quantum mechanical system is subject to no such constraint; the result of each step of the computation may be a superposition of computational states. Using this principle, a quantum Turing machine may, in a sense that will be explained more fully in Section 4, explore all possible computational paths while performing only a single computation. It is this principle of quantum parallelism that gives quantum computers their power. It is important to note at this point that a quantum Turing machine, and therefore a quantum computer, is no more powerful than a classical Turing machine, in the sense that there is no function that may be computed by one that may not be by the other. It appears, however, that quantum Turing machines may be able to compute some functions *more efficiently* than classical Turing machines.

The two most famous and useful quantum algorithms known are Shor's factoring algorithm and Grover's database search algorithm. Shor's algorithm can factor numbers faster than any known classical algorithm (though it has not yet been proved that factoring is an inherently inefficient problem for a classical computer), while Grover's algorithm can speed up the search of an unstructured database by a factor of a square root of the number of elements in the database over the best classical algorithm. Both of these quantum algorithms, though especially Shor's algorithm, have important applications to cryptography. The most widespread cryptosystem in use today is the RSA encryption scheme, which relies on the difficulty of factoring large numbers for its security. A quantum computer could decode an RSA encrypted message in seconds. Thus, research in quantum computation has also spurred research in the related field of quantum cryptography, in which quantum mechanical properties are used to create unbreakable codes.

Despite the obvious benefits of quantum computers, none yet exists. Many physical realizations have been proposed, but none has yet produced large quantum computers, even though the requirements for quantum computation seem minimal. Any quantum system possessing two nondegenerate states can be used as a quantum bit, or qubit, which is the basic unit of quantum information, just as the bit is the basic unit of classical information. A quantum computer would consist of many qubits and some mechanism for changing their state. Additionally, it must be possible to entangle different qubits to create quantum registers; this notion will be explained in a later section.

To date, only small quantum computers running a single algorithm have been built, despite the simple requirements. Though no solution has yet been found that will scale easily to large numbers of qubits, some implementations have been quite successful on a small scale as proof-of-principle experiments. The most advanced of these small quantum computers are based on nuclear magnetic resonance (NMR). In an NMR quantum computer, nuclear spins are used as qubits, and radio frequency waves are used to change the state of the qubits. A simpler implementation, discussed extensively in a later section of this paper, is based on linear optics, where combinations of photon spatial modes and polarization states are used as qubits.

Experiments have also been done using quantum dots, cavity QED systems, and trapped ions, to name a few. The reader is directed to Nielsen and Chuang's book [1] for an overview of some of these implementations.

There are problems with each of these implementations, however, that have stymied the progress towards a working universal quantum computer. These problems can be separated into implementation-specific issues and issues of general difficulty. The most problematic of these general difficulties is decoherence, or the tendency of the delicate superpositions necessary for quantum computation to decay due to interactions with their environment. Several schemes have been developed for dealing with this problem, most notably the theories of decoherence-free subspaces and quantum error correcting codes. While these schemes do reduce the adverse effects caused by decoherence, they also require extra qubits that will slow the development of working quantum computers.

In this paper, we present an overview of the field of quantum computation, along with a complete treatment of linear optical quantum computation as an example of a possible experimental realization of the theory. Section 2 develops the theory of quantum mechanics from the viewpoint of quantum computation and introduces some useful ideas from the theory of computation. Section 3 gives an overview of the quantum circuit model, and determines a set of quantum logic gates that is universal for quantum computation. Section 4 discusses several quantum algorithms, beginning with the relatively simple Deutsch and Deutsch-Jozsa algorithms and also covering Grover's and Shor's algorithms. Finally, Section 5 provides a complete description of a hypothetical quantum computer implemented using linear optics.

2 Mathematical Tools

2.1 Quantum Mechanics

The theory of quantum mechanics is built on top of the theory of linear algebra. In order to describe the physical world, we must define some mapping between the mathematical concepts and observable physical dynamics. This is accomplished by defining a set of postulates. We can say that the postulates are valid if the resultant theory agrees with experiment, and indeed the predictions of quantum mechanics agree impeccably with observations.

In this section, we develop the theory of quantum mechanics from the viewpoint of quantum computation. The following discussion is adapted from Boccio [4] and Nielsen and Chuang [1].

2.1.1 States and Qubits

To begin with, we must define a framework within which to work; to that end, we make the following postulate:

Postulate 1 *A physical system is represented as a Hilbert space, and a state of the system is given by a unit vector* $|\psi\rangle$ *in that space. In particular, a qubit is a unit vector in a two dimensional Hilbert space.*

A Hilbert space is simply a complex linear vector space with an inner product defined. Thus it is clear that linear algebra will be the primary mathematical formalism used in our theory. In quantum computation, we are primarily interested in qubits; thus we will be most interested in simple two-dimensional vector spaces.

Qubits are the quantum analog of classical bits. As two dimensional vectors, they have two discrete states, which we label $|0\rangle$ and $|1\rangle$. We also here define the so-called standard computational basis, in which

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.1}$$

We will often make use of this basis, and unless otherwise stated, it is the basis used throughout this paper.

In addition to being in either of the basis states $|0\rangle$ or $|1\rangle$, a qubit may exist in any superposition of these states, given by

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.2}$$

with

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.3}$$

The fact that qubits can be in definite superpositions is the root of the increased computational power of quantum computers relative to classical computers.

2.1.2 Operators and Evolution

Now that we have developed a mathematical representation of a physical system and its states, we must specify how states evolve. We thus make our second postulate:

Postulate 2 *Closed systems evolve via unitary transformations. Thus, quantum logic gates, which change the state of qubits, are represented as unitary operators.*

A unitary operator is one whose Hermitian conjugate is its inverse. Thus, if U is a unitary operator,

$$U^\dagger U = UU^\dagger = I, \tag{2.4}$$

where I is the identity operator. This allows us to state and prove a lemma concerning the eigenvalues of a unitary operator that we shall use later in this paper:

Lemma 2.1 *A unitary operator has eigenvalues of the form $e^{i\theta}$, and any operator with eigenvalues of this form is unitary.*

Proof To prove the first part of the lemma, consider the matrix representation of a unitary operator in its eigenbasis. In this basis, it will be diagonal with its eigenvalues v_i down the diagonal, where the v_i are in general complex. The product $U^\dagger U$ in this basis will then be a diagonal matrix with the values $v_i^* v_i$ down the diagonal. To fulfill the unitary condition, we must have $v_i^* v_i = 1$; thus, the real part of the v_i must be 1, and therefore the v_i will be of the form $e^{i\theta_i}$. Examining the product $UU^\dagger$ gives the same result.

The second part of the lemma is proved in much the same way. We define some operator Q with eigenvalues $e^{i\phi_i}$, and work in its eigenbasis. In this basis, the product $Q^\dagger Q$ has a matrix representation consisting of $e^{-i\phi_i} e^{i\phi_i} = 1$ down the diagonal, and thus is the identity matrix. The same result is obtained given the product $QQ^\dagger$, and so Q fulfills the unitarity condition and the lemma is proved.

□

Notice that postulate 2 specifies only that closed systems undergo unitary evolution. No specification is made for systems that are not closed; presumably, the dynamics are far more complex. Many of the difficulties encountered in the effort to construct a working quantum computer stem from the interactions of the computer with its environment; quantum computers are only approximately closed systems. In this paper, however, we consider only ideal, closed quantum computational systems, and so are concerned only with purely unitary evolution.

Postulate 2 associates the class of unitary operators with state evolution. We now associate a second class of operators with physical observables, which will lead us to a representation of quantum measurement:

Postulate 3 *A physical dynamical variable is represented by a Hermitian operator, and the possible outcomes of a measurement of that observable are the eigenvalues of the operator.*

A Hermitian operator H is one that is equal to its Hermitian conjugate, satisfying the relation $H = H^\dagger$. Just as the unitarity condition restricts the eigenvalues of a unitary operator, the hermiticity condition imposes a restriction on the eigenvalues of a Hermitian operator:

Lemma 2.2 *The eigenvalues of a Hermitian operator are real.*

Proof Consider a Hermitian operator H, and again consider its matrix representation in its eigenbasis, where it is diagonal with its eigenvalues η_i down the diagonal and the η_i are in general complex. To satisfy the hermiticity condition, $H = H^\dagger$, the relation $\eta = \eta^*$ must hold. Thus, the η_i must be real.

□

The condition proved in lemma 2.2 makes Hermitian operators a good choice to model observables, since, of course, we can never measure a complex number.

As we show below, the eigenvectors of a Hermitian operator are orthogonal, a fact we shall often use:

Theorem 2.1 *The eigenvectors of a Hermitian operator corresponding to distinct eigenvalues are orthogonal.*

Proof Let Q be a Hermitian operator, and let $|\psi\rangle$ and $|\phi\rangle$ be eigenvectors of Q with eigenvalues ψ and ϕ, respectively, such that $\psi \neq \phi$. The, we have

$$\phi \langle\psi|\phi\rangle = \langle\psi|Q|\phi\rangle = (Q|\psi\rangle)^\dagger |\phi\rangle = (\psi|\psi\rangle)^\dagger |\phi\rangle = \psi \langle\psi|\phi\rangle, \tag{2.5}$$

where the last equality follows from lemma 2.2. Therefore,

$$\phi \langle\psi|\phi\rangle = \psi \langle\psi|\phi\rangle, \tag{2.6}$$

and so $\langle\psi|\phi\rangle = 0$, which means that $|\psi\rangle$ and $|\phi\rangle$ are orthogonal.

□

2.1.3 Composite States and Registers

So far in our discussion of quantum mechanics, we have dealt only with single systems; from the standpoint of quantum computation, we have only discussed single qubits. A quantum computer with the capacity to operate on only a single qubit is useless, and so we must extend our discussion to systems of many qubits. We therefore introduce the following postulate:

Postulate 4 *The Hilbert space describing a composite physical system is the tensor product of the spaces describing the individual systems, and the state vector of the system is the tensor product of the individual state vectors. A quantum register is the tensor product of individual qubits.*

The tensor product, denoted by the $\otimes$ operator, is associative and distributive. In terms of matrices and vectors, the tensor product is reduced to the so-called Kronecker product. The Kronecker product of an $m \times n$ matrix A and a $p \times q$ matrix B is given by

$$A \otimes B = \overbrace{\left.\begin{pmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \cdots & \cdots & \vdots & \cdots \\ A_{m1}B & A_{m2}B & \cdots & A_{mn}B \end{pmatrix}\right\}}^{nq} mp. \tag{2.7}$$

In equation (2.7), each entry is a submatrix of the dimensions of B, with each entry in B multiplied by a component of A as shown. The Kronecker product of a vector follows directly from equation (2.7) by simply setting n and q to 1. Let us illustrate the tensor product with a simple example.

Consider the matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.8}$$

We will see in Section 3 that this operator is of fundamental importance in quantum computation. Using the definition in equation (2.7), we now calculate $H \otimes H$:

$$H \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} H & H \\ H & -H \end{pmatrix}. \tag{2.9}$$

Expanding this out, we have

$$H \otimes H = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \tag{2.10}$$

In a similar way, we can define multiqubit states in terms of the tensor product. Consider a simple quantum register consisting of a qubit in the $|0\rangle$ state and a qubit in the $|1\rangle$ state. Using postulate 4, we can write this register as $|0\rangle \otimes |1\rangle$. The representation of this state in the computational basis is quickly given by equation (2.7):

$$|0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \tag{2.11}$$

This process generalizes simply to larger quantum registers. We shall often use a shorthand notation to describe such multiqubit states. We define $|0\rangle \otimes |1\rangle \equiv |01\rangle$. Whenever we write a qubit with more than one label in the ket, a tensor product is implied.

Before leaving this section, let us introduce one more piece of notation we will use throughout this paper. Consider an operator Q that operates on a single qubit. To extend this operation to a register of n qubits, we take its tensor product with itself n times. We represent this operation by $Q^{\otimes n}$, in analogy with exponentiation. We use the same notation for qubits; in our notation, the state $|0\rangle^{\otimes n}$ represents a quantum register n qubits long with every qubit in the $|0\rangle$ state.

2.2 The Theory of Computation

The theory of computation may broadly be divided into two sections: computability and complexity. Computability theory has led to a knowledge of the kinds of problems that can and cannot be solved by a computing device. It is straightforward to show that there must be functions that cannot be computed: the set of all possible functions is uncountably infinite, while the set of all possible programs is countably infinite, so that there are more functions than programs. But while computability theory is a rich field and interesting in its own right, we will not consider it here. It cannot shed any light on the difference between classical and quantum computers, because the set of functions computable by each type of computer is identical.

The notion that quantum computers are more powerful than their classical counterparts stems from another branch of the theory of computation, namely complexity theory. Early in the development of the theory of computation, it was realized that simply classifying functions as computable or not was too coarse grained, and not useful for practical purposes; there are many useful computable functions that nevertheless cannot be computed in a reasonable amount of time. Classifying functions (and therefore the algorithms that implement them) by the exact amount of running time they required is too fine grained an approach, leading to needless extra taxonomy. Complexity theory thus defines broad complexity classes, and states that algorithms running in a time proportional to a polynomial in their input size are feasible, and algorithms running in exponential time are infeasible.

We will clarify these notions in a moment; first, however, we take a detour to discuss classical and quantum Turing machines, which are the basis for all of the theory of computation.

2.2.1 Classical Turing Machines

While a computing device may be thought of as a function evaluator, it must also be an information processor. Turing machines[1] are defined from this perspective, and build on the theory of formal languages. We define an alphabet Σ to be a finite set of symbols (which may always be the set of binary numbers, $\{0, 1\}$). A string is a finite sequence of symbols from Σ. Predictably, a language is defined to be a

[1]Though we refer to Turing "machines," the reader should understand that a Turing machine does not refer to a physical device, but rather to a mathematical object. Whenever we refer to machines or devices in this section, we are referring simply to mathematical constructs.

set of strings over an alphabet. We also introduce the notion of an automaton or language acceptor. An automaton is a device that is paired with a given language. Suppose we have an automaton M and a language L. We now feed M with strings. If M returns "true" for all strings that are in L and "false" for all strings not in L, we say that M accepts L.

Formal languages fall into several different categories, and each class of languages is paired with a corresponding class of automaton. A full discussion of the so-called Chomsky hierarchy [5], which ranks the different classes of formal languages based on their generality, is beyond the scope of this paper; we simply state that the most general language classes are the recursive and recursively enumerable languages[2], both of which have Turing machines as their automata. Because Turing machines are related to the most general formal languages, we regard them as a good model of computation. Indeed, no alternate model of computation has yet been proposed that cannot be proven to be equivalent to the Turing machine model.

A Turing machine may be thought of as consisting of three elements, as shown in Figure 1. The first of these is an infinite tape marked off into segments. Each

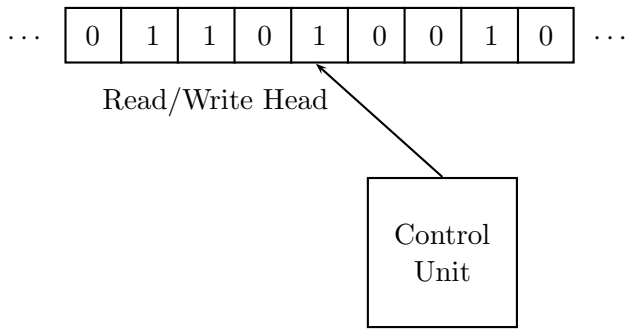

Figure 1: A representation of a Turing machine.

segment may either contain some symbol or be blank. The Turing machine also has a read/write head that can move along the length of the tape, reading in symbols and possibly erasing or overwriting them. Finally, it has a finite control unit that may be in some state. The action of the Turing machine will be determined by both

[2]The precise definitions of recursive and recursively enumerable languages are unimportant for the present discussion. Formally, the recursive languages are the class of languages decided by a Turing machine, while the recursively enumerable languages are those languages semidecided by a Turing machine. A Turing machine is said to decide a language if it halts on all strings, and accepts strings in the language and rejects strings not in the language. A Turing machine semidecides a languages if it halts only on strings in the language and does not halt on other strings.

its state and the currently read symbol on the tape. It is important to note that at the end of every computational step, the machine will be in a definite state.

Formally, then, a Turing machine $M$is a quintuple $(K, \Sigma, \delta, s, H)$, where K is a finite set of states, Σ is an alphabet, s is the start state, H is the set of halting states, and δ is a function from $(K - H) \times \Sigma$ to $K \times \Sigma$, where $\times$ denotes the Cartesian product. δ specifies the action of the Turing machine based on its current state and the symbol it is currently reading. For the class of deterministic Turing machines, δ is a true function, and must be single valued and must contain an entry for every pair from $K - H$ and Σ. For a nondeterministic Turing machine, the transition function need only be a relation: it may be multiply valued, and need not contain entries for all possible elements of $(K - H) \times \Sigma$. We represent such a nondeterministic transition function by Δ. A computation of a nondeterministic Turing machine proceeds along all possible paths simultaneously; however, one must remember that the nondeterministic Turing machine is simply a mathematical construct and cannot, even in principle, be built.

Remarkably, although many alternate models of computing devices have been proposed, none has ever been shown to be more powerful than a Turing machine. Even augmenting the standard Turing machine described above with multiple tapes or multiple read/write heads does not allow the computation of any new functions; indeed, any computation on such an extended Turing machine may be efficiently translated into a computation on a standard Turing machine. It should be mentioned, however, that it has not been proved that the action of a nondeterministic Turing machine can be simulated efficiently on a standard deterministic Turing machine, and indeed most computer scientists believe that it cannot be. We will return to this question in our discussion of computational complexity below.

2.2.2 Quantum Turing Machines

The theory of the quantum Turing machine was first published by Deutsch [3]. Deutsch's quantum Turing machine is similar to a classical Turing machine in that it consists of a finite control unit and an infinite tape. Instead of being able to record only classical information, however, the tape is now capable of holding a qubit at each location; that is, the tape is made up of an infinite sequence of two-level observables. The control unit is likewise a quantum device, characterized by some number of two-level observables. Deutsch also introduces a pointer to the current position of the read/write head on the tape, characterized by yet another observable. A state of the machine is given by a vector in the Hilbert space constructed from the space of the head pointer, the tape slots, and the control unit.

As a quantum device, the quantum Turing machine evolves via unitary dynamics. Deutsch specified that only a single tape square can participate in a given operational step of the machine. This ensures that the machine will operate by finite means, a requirement on any reasonable model of computation.

While the quantum Turing machine is running, it must not be observed; other-

wise, any superposition on the tape or in the control unit will be collapsed. This requirement, however, makes it difficult to determine when the machine has halted. Deutsch escaped this problem by specifying that the quantum Turing machine should keep one extra qubit with which it can signal the halt condition. This qubit may be periodically observed without disturbing the rest of the computation.

Though we will not refer back to the concept of the quantum Turing machine, it is important to understand that it is a fully developed theoretical model that can be proved to be as powerful as a classical Turing machine.

2.2.3 Computational Complexity

Computational complexity provides a framework for determining the efficiency of algorithms. As discussed above, quantifying the exact running time of an algorithm is often too fine-grained to be useful; we would like, however, some measure of the running time. To this end, we define the order of a function $f(n)$ as follows: we say that $f(n)$ is $\mathcal{O}(g(n))$ if, for some constants c and n_0, $f(n) \leq cg(n)$ for all n greater than n_0. This is often referred to as 'big-O' notation; we also refer to it as the time complexity of an algorithm. We note that the time complexity defined in this way specifies an upper bound on the running time; it is the running time of the algorithm in its worst case. One can also define lower asymptotic bounds, but since big-O notation is more standard, we use it.

We can now use this concept of time complexity to define what is meant by an efficient algorithm. Algorithms whose time complexity is polynomial in their argument size are deemed tractable and efficient. Algorithms with exponential time complexities are deemed inefficient. Algorithms that fall into these two categories are grouped together in large complexity classes. The class $\mathcal{P}$ is the set of all polynomial time algorithms, while the class $\mathcal{EXP}$ is the set of all exponential time algorithms. It should be clear that $\mathcal{P} \subset \mathcal{EXP}$. These classes can also be defined in terms of Turing machines; for example, $\mathcal{P}$ is the set of algorithms that run in polynomial time on a deterministic Turing machine.

The major outstanding problem in computer science today concerns a third complexity class, the class $\mathcal{NP}$. $\mathcal{NP}$ is generally defined to be the class of algorithms that run in polynomial time on a nondeterministic Turing machine. $\mathcal{NP}$ may also be defined to be the class of problems with solutions that may be checked in polynomial time by a deterministic Turing machine. The most important unsolved question in computer science is whether or not $\mathcal{P} = \mathcal{NP}$. It is widely believed that the two classes are not equal, though proving this fact has turned out to be very difficult. While important in its own right, proving $\mathcal{P} \neq \mathcal{NP}$ also has implications for quantum computation; though we do not give a proof here, such a proof could be used to show that quantum computers are provably more efficient than classical computers.

The field of computational complexity is much broader than we can treat here. The interested reader is directed to Lewis and Papadimitriou's book [5] for more information.

3 Quantum Gates and Circuits

3.1 Quantum Logic

A classical computer is, in essence, a device that processes classical information, encoded in bits. While this high level description is good for dealing with a computer as a mathematical entity, it is useful when considering a computer to be a physical device to model a computer from a more physically motivated perspective. Classical digital computers are built from logic gates and wires that are grouped together to form more complex circuits. Data moves along the wires and is changed by the logic gates in such a way as to implement the desired computation. Logic gates themselves are defined by their action when fed with different input bits, usually codified in a truth table.

Analogously, since a quantum computer is a processor of quantum information, we define a quantum circuit to be a network of quantum logic gates and quantum wires that act on qubits. To be consistent with the postulates of quantum mechanics listed in Section 2, we define a quantum logic gate to be a unitary transformation on one or more qubits. A quantum wire is simply a theoretical construct to denote movement of qubits, either through space or time. In this way, our quantum circuit model applies equally well to any implementation of a quantum computer.

In this section, we first present several single qubit gates, which are some of the most important quantum gates. Next, we consider multi-qubit controlled operations, and important extensions. Finally, we consider a discrete set of gates that is universal for quantum computation.

Before moving on to our discussion of single qubit gates, let us introduce some notation. Figure 2 shows our representation of a quantum circuit. The lines

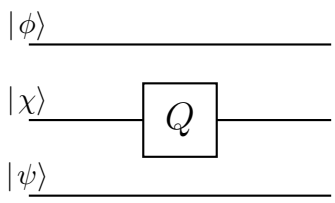

Figure 2: A generic quantum circuit. The lines represent quantum wires while the box signifies a quantum gate Q. Information flows from left to right.

represent quantum wires and the box signifies a quantum logic gate Q. Each wire carries a single qubit, and we assume that all the qubits in a particular circuit are a part of the same system, and that the state of the circuit at any time may be

described by a tensor product of the individual qubits. Any quantum gate in the circuit will act only on the qubits carried by the wires that pass through it; thus, quantum gates will often act on subspaces of the Hilbert space of the full circuit. Additionally, we define the direction of information flow in a quantum circuit to be from left to right. Using these rules, the circuit in Figure 2 represents the operation

$$(I \otimes Q \otimes I)(|\phi\rangle \otimes |\chi\rangle \otimes |\psi\rangle) = |\phi\rangle \otimes Q|\chi\rangle \otimes |\psi\rangle \,. \tag{3.1}$$

Note that we have here introduced the convention that when a tensor product of operators acts on a tensor product of qubits, the first operator acts on the first qubit, the second operator on the second qubit, and so on. Thus, in equation (3.1), the operator Q acts only on the qubit $|\chi\rangle$, while the other two qubits are acted upon by the identity operator.

3.2 Single Qubit Gates

The building blocks of any quantum circuit are operations on single qubits. In this section, we introduce several single qubit gates that we will use frequently.

We begin with the well-known Pauli matrices X, Y, and Z. In the computational basis, these gates have the matrix representations

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{3.2}$$

In addition to specifying their matrix forms, it is also instructive to express the Pauli matrices in terms of projectors in the computational basis:

$$\begin{aligned} X &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ Y &= i|0\rangle\langle 1| - i|1\rangle\langle 0| \\ Z &= |0\rangle\langle 0| - |1\rangle\langle 1| \end{aligned} \tag{3.3}$$

Finally, we also specify the action of the Pauli matrices on the general qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$:

$$\begin{aligned} X|\psi\rangle &= \alpha|1\rangle + \beta|0\rangle \\ Y|\psi\rangle &= i\alpha|1\rangle - i\beta|0\rangle \\ Z|\psi\rangle &= \alpha|0\rangle - \beta|1\rangle \end{aligned} \tag{3.4}$$

The Pauli matrices have many interesting properties. Simply by inspecting their matrix forms, it is easy to show that the Pauli matrices all square to the identity matrix. Using this property, we can prove a useful identity. Consider $\exp(ix\sigma)$ for some real number x and matrix σ such that $\sigma^2 = I$. Then, expanding the exponential, we have

$$e^{ix\sigma} = \sum_{n=0}^{\infty} \frac{(ix\sigma)^n}{n!}. \tag{3.5}$$

Breaking this sum into its even and odd components, we have

$$e^{ix\sigma} = \sum_{n\ even} \frac{(ix\sigma)^n}{n!} + \sum_{m\ odd} \frac{(ix\sigma)^m}{m!}. \tag{3.6}$$

We know that $\sigma^2 = I$; thus, σ raised to any even power will be I and to any odd power will be σ. Using this fact, the summations become

$$e^{ix\sigma} = \sum_{n\ even} \frac{(i)^n x^n}{n!} + \sum_{m\ odd} \frac{(i)^m x^m \sigma}{m!}. \tag{3.7}$$

Now, we let $n = 2k$ and $m = 2\ell + 1$, so that we can rewrite the summations as

$$e^{ix\sigma} = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!} + i\sigma \sum_{\ell=0}^{\infty} \frac{(-1)^\ell x^{2\ell+1}}{(2\ell+1)!}. \tag{3.8}$$

But these are simply the expansions of the sine and the cosine; thus,

$$e^{ix\sigma} = \cos x + i\sigma \sin x. \tag{3.9}$$

We can now use this identity to define the following useful rotation operators about the x, y, and z axes, given by

$$\begin{aligned}
R_x(\theta) &\equiv e^{i\theta X/2} = \cos\frac{\theta}{2} - iX\sin\frac{\theta}{2} = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \\
R_y(\theta) &\equiv e^{i\theta Y/2} = \cos\frac{\theta}{2} - iY\sin\frac{\theta}{2} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \\
R_z(\theta) &\equiv e^{i\theta Z/2} = \cos\frac{\theta}{2} - iZ\sin\frac{\theta}{2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}.
\end{aligned} \tag{3.10}$$

Additionally, we can define a rotation operator about some arbitrary axis n as

$$R_n(\theta) \equiv e^{-i\theta\hat{n}\cdot\boldsymbol{\sigma}/2} = \cos\frac{\theta}{2} - i\left(n_x X + n_y Y + n_z Z\right)\sin\frac{\theta}{2}, \tag{3.11}$$

where $\hat{n} = (n_x, n_y, n_z)$ is a unit vector in the n direction and $\boldsymbol{\sigma} = (X, Y, Z)$ is a vector of the Pauli matrices. This general rotation follows because

$$(\hat{n}\cdot\boldsymbol{\sigma})^2 = I, \tag{3.12}$$

and so equation (3.10) defines a rotation is accordance with our earlier findings for operators that square to unity. To see why equation (3.12) holds, let us expand it out:

$$\begin{aligned}
(\hat{n}\cdot\boldsymbol{\sigma})^2 &= (n_x X + n_y Y + n_z Z)^2 \\
&= (n_x^2 + n_y^2 + n_z^2)I + n_x n_y \{X, Y\} + n_x n_z \{X, Z\} + n_y n_z \{Y, Z\}
\end{aligned} \tag{3.13}$$

where $\{\cdot,\cdot\}$ denotes the anticommutator. It is becomes apparent simply by writing it out that for any two Pauli matrices σ_i and σ_j,

$$\sigma_i\sigma_j = i\epsilon_{ijk}\sigma_k, \tag{3.14}$$

where we have invoked the Einstein summation convention and ϵ_{ijk} is the Levi-Civita totally antisymmetric tensor. This in turn shows that $\{\sigma_i, \sigma_j\} = 0$, or that the Pauli matrices anticommute. This fact combined with the requirement that $\hat{n}$ be a unit vector means that equation (3.13) reduces to the identity operator, so that equation (3.11) holds.

The Pauli matrices are a basis for the space of two by two matrices, and as such, any single qubit operator can be expressed in terms of them. We now make use of this fact to prove a theorem that will be useful in our study of controlled operations. First, we state a small lemma that will be necessary for proving the theorem:

Lemma 3.1 *Suppose U is a single qubit quantum gate. Then there exist real numbers α, β, γ, and δ such that*

$$U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta). \tag{3.15}$$

Proof Since U is a single qubit gate, it must be unitary, as we have postulated that all quantum dynamics are unitary. Since U is unitary, its rows and columns must be orthonormal. Suppose we write U as

$$U = \begin{pmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}\cos\frac{\gamma}{2} & -e^{i(\alpha-\frac{\beta}{2}+\frac{\delta}{2})}\sin\frac{\gamma}{2} \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}\sin\frac{\gamma}{2} & e^{i(\alpha+\frac{\beta}{2}+\frac{\delta}{2})}\cos\frac{\gamma}{2} \end{pmatrix} \tag{3.16}$$

for real numbers α, β, γ, and δ. We claim that this form of U ensures the orthonormality of the rows and columns, and thus the unitarity of U. Labelling the columns of U as $\mathbf{u}_{c1}$ and $\mathbf{u}_{c2}$ and the rows as $\mathbf{u}_{r1}$ and $\mathbf{u}_{r2}$, it is simple to see that

$$\begin{aligned} \mathbf{u}_{c1}\cdot\mathbf{u}_{c2} &= -e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{i(\alpha-\frac{\beta}{2}+\frac{\delta}{2})}\cos\frac{\gamma}{2}\sin\frac{\gamma}{2} + e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}e^{i(\alpha+\frac{\beta}{2}+\frac{\delta}{2})}\sin\frac{\gamma}{2}\cos\frac{\gamma}{2} \\ &= -2\sin\gamma e^{i(2\alpha+\beta)} + 2\sin\gamma e^{i(2\alpha+\beta)} \\ &= 0. \end{aligned} \tag{3.17}$$

Likewise, $\mathbf{u}_{r1}\cdot\mathbf{u}_{r2} = 0$. Thus, writing U in this form ensures its unitarity, and then equation (3.15) follows from matrix multiplication.

□

Now, we can state the theorem:

Theorem 3.1 *Let U be a single qubit gate. Then there exist single qubit gates A, B, and C such that $ABC = I$ and $U = e^{i\alpha}AXBXC$, where α is an overall phase factor.*

Proof Let us define

$$\begin{aligned} A &\equiv R_z(\beta)R_y\left(\frac{\gamma}{2}\right) \\ B &\equiv R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\delta+\beta}{2}\right) \\ C &\equiv R_z\left(\frac{\delta-\beta}{2}\right). \end{aligned} \tag{3.18}$$

Then we have

$$\begin{aligned} ABC &= R_z(\beta)R_y\left(\frac{\gamma}{2}\right)R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\delta+\beta}{2}\right)R_z\left(\frac{\delta-\beta}{2}\right) \\ &= R_z(\beta)R_z(-\beta) = I. \end{aligned} \tag{3.19}$$

Now, note that, using equation (3.14),

$$XYX = iZX = i(iY) = -Y. \tag{3.20}$$

Therefore,

$$\begin{aligned} XR_y(\theta)X &= X(\cos\theta + iY\sin\theta)X = \cos\theta - iY\sin\theta \\ &= R_y(-\theta), \end{aligned} \tag{3.21}$$

where we have used $X^2 = I$. Similarly,

$$XR_z(\theta)X = R_z(-\theta). \tag{3.22}$$

With these identities in mind, we can show that

$$\begin{aligned} XBX &= XR_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\delta+\beta}{2}\right) \\ &= XR_y\left(-\frac{\gamma}{2}\right)XXR_z\left(-\frac{\delta+\beta}{2}\right) \\ &= R_y\left(\frac{\gamma}{2}\right)R_z\left(\frac{\delta+\beta}{2}\right). \end{aligned} \tag{3.23}$$

Thus,

$$\begin{aligned} AXBXC &= R_z(\beta)R_y\left(\frac{\gamma}{2}\right)R_y\left(\frac{\gamma}{2}\right)R_z\left(\frac{\delta+\beta}{2}\right)R_z\left(\frac{\delta-\beta}{2}\right) \\ &= R_z(\beta)R_y(\gamma)R_z(\delta), \end{aligned} \tag{3.24}$$

and thus, by lemma 3.1, we have

$$U = e^{i\alpha}AXBXC. \tag{3.25}$$

□

Before leaving this section, we introduce three additional useful single qubit gates: the Hadamard gate H, phase gate S, and $\pi/8$ gate T. These three gates have matrix and projector representations given by

$$\begin{aligned} H &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\langle 0| + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\langle 1| \\ S &= \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = |0\rangle\langle 0| + i\,|1\rangle\langle 1| \\ T &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} = |0\rangle\langle 0| + e^{i\pi/4}\,|1\rangle\langle 1|\,. \end{aligned} \tag{3.26}$$

Of these three gates, we shall use the Hadamard gate most frequently, and so let us investigate it further. Consider its projector representation, and notice that

$$\begin{aligned} H\,|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H\,|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{3.27}$$

The Hadamard gate thus takes either of the computational basis states and puts it into a superposition state. Consider now a tensor product of two Hadamard gates acting on the two qubit state $|00\rangle = |0\rangle \otimes |0\rangle$. Explicitly calculating this operation, we have

$$H^{\otimes 2}\,|0\rangle^{\otimes 2} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle). \tag{3.28}$$

Thus, we see that this combination of two Hadamard gates acting on the $|00\rangle$ state gives an even superposition of all the computational basis states for a two qubit system. Let us now generalize this result.

Look again at equation (3.27). It is clear that for an arbitrary single qubit state $|\phi\rangle$, we can write

$$H\,|\phi\rangle = \sum_{k=0}^{1} \frac{(-1)^{\phi k}\,|k\rangle}{\sqrt{2}}. \tag{3.29}$$

Let us now generalize this to the case of n qubits. We have

$$H^{\otimes n}\,|\phi_1, \phi_2, \ldots, \phi_n\rangle = \sum_{k_1,\ldots,k_n} \frac{(-1)^{\phi_1 k_1 + \phi_2 k_2 + \ldots + \phi_n k_n}\,|k_1, k_2, \ldots, k_n\rangle}{\sqrt{2^n}}, \tag{3.30}$$

which we can summarize as

$$H^{\otimes n}\,|\phi\rangle = \sum_{k} \frac{(-1)^{\phi \cdot k}\,|k\rangle}{\sqrt{2^n}}. \tag{3.31}$$

Here, $\phi \cdot k$ represent the bitwise inner product of ϕ and k; that is, the sum modulo 2 of the products of the individual digits of ϕ and k when both are written in binary. Note also that we term such an n-fold tensor product of Hadamard gates the Hadamard transform. Let us now consider the case where $|\phi\rangle = |0\rangle^{\otimes n}$. In that case, $\phi \cdot k = 0$, and so

$$H^{\otimes n} |0\rangle^{\otimes n} = \sum_k \frac{|k\rangle}{\sqrt{2^n}}, \tag{3.32}$$

which is again an even superposition of computational basis states. We will often make use of this result.

3.3 Controlled Operations

Now that we have discussed single qubit dynamics and have determined how to construct any arbitrary single qubit gate using the Pauli matrices, let us move on to multiqubit gates. The most important class of multiqubit gates, and as we shall see the only class we need concern ourselves with, is the class of controlled operations.

A controlled operation is a single qubit operation performed on a target qubit predicated on the values of one or more control qubits. The most important of the controlled operations, and indeed also the simplest, is the control-NOT gate. This gate flips the value of the target qubit if the control qubit is in the $|1\rangle$ state, and it is the quantum analog of the classical XOR gate (see Appendix A). In the computational basis, the control-NOT gate is given by

$$\begin{aligned} CN &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\ &= |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|, \end{aligned} \tag{3.33}$$

where we have abbreviated the two-qubit tensor product states by using two labels in the kets. Figure 3 shows the representation of both the control-NOT gate as well as a general controlled-U gate.

We now show how to construct a general controlled operation. Recall from theorem 3.1 that an arbitrary single qubit gate U can be written $U = e^{i\alpha} AXBXC$. The first step in our construction is to perform a controlled phase shift by α; that is, we want to rotate the phase of the target qubit if the control qubit is in the state $|1\rangle$. This two-qubit transformation is given by

$$\begin{aligned} CP &= |00\rangle\langle 00| + |01\rangle\langle 01| + e^{i\alpha}|10\rangle\langle 10| + e^{i\alpha}|11\rangle\langle 11| \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\alpha} & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{pmatrix}. \end{aligned} \tag{3.34}$$

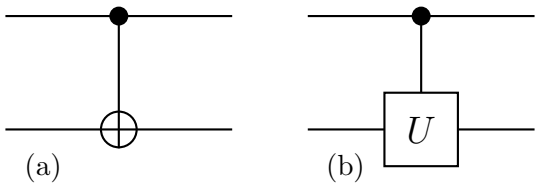

Figure 3: (a) A control-NOT gate. The dark dot represents the control qubit, while the open circle represents the target qubit. (b) A more general controlled-U operation.

We note, however, that this transformation can be decomposed into two single qubit operations, since

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix} \otimes I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\alpha} & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{pmatrix}. \tag{3.35}$$

Here, the leftmost operator in the tensor product acts only on the control qubit and the rightmost only on the target qubit. Thus, since target qubit is acted upon by the identity in this transformation, the controlled phase shift can be replaced with a single qubit transformation on the control qubit. Now, consider the circuit in Figure 4; we claim that this circuit implements a general controlled operation. To see why

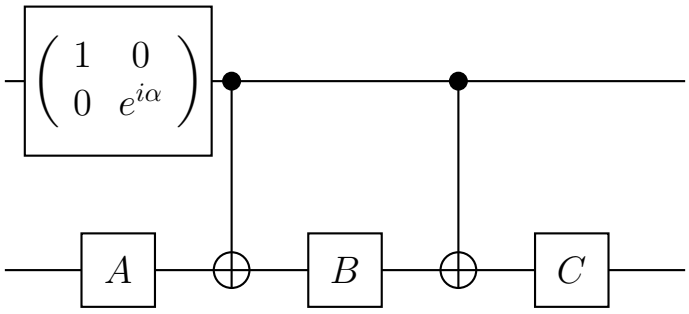

Figure 4: A circuit implementing an arbitrary controlled U operation, where $U = e^{i\alpha}AXBXC$.

this should be, first notice that a control-NOT is exactly a controlled X operation, which makes sense given that the X gate is the quantum analog of a classical NOT gate, as is easy to show using the definition in equation (3.3):

$$\begin{aligned} X|0\rangle &= (|0\rangle\langle 1| + |1\rangle\langle 0|)|0\rangle = |1\rangle \\ X|1\rangle &= (|0\rangle\langle 1| + |1\rangle\langle 0|)|1\rangle = |0\rangle . \end{aligned} \tag{3.36}$$

We now have two cases to consider: first, suppose the control qubit is in the state $|0\rangle$. In this case, the two control-NOT gates do not change the state of the target qubit, and the transformation ABC is applied to the target qubit. But, as part of theorem 3.1, $ABC = I$, and so if the control qubit is not set the state of the target qubit does not change. Now, suppose the control qubit is in the state $|1\rangle$. In this case, the transformation $e^{i\alpha}AXBXC$ is applied to the target qubit, which is exactly U. The circuit in Figure 4 does indeed then implement a general controlled U operation.

Finally, we introduce one more piece of notation before moving on. We have thus far only considered controlled operations that perform some transformation if a control qubit is in the $|1\rangle$ state. We can just as easily perform the transformation if the control qubit is in the $|0\rangle$ state. We use an open circle to denote a control of this type, as shown in Figure 5. We also note that since the X operator is

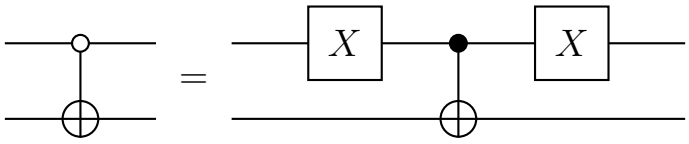

Figure 5: A controlled operation conditioned on the control qubit being in the $|0\rangle$ state and its implementation using a control-NOT gate.

the analog of a classical NOT gate, we may use it to map this new kind of control operation into the type we have already discussed.

3.4 Universal Gate Sets

Now that we have discussed single and double qubit gates, we are ready to move on to the subject of universal sets. A universal gate set is a finite set of logic gates that can be used to implement any arbitrary circuit. As an example, the NAND gate, an AND gate with its output inverted, is universal for classical computation. In this section, we develop a quantum gate set that is universal for quantum computation.

Ours will be an approximate universal set, but it can approximate any unitary transformation to arbitrary accuracy. Following Nielsen and Chuang[1], we first prove the universality of two-level unitary matrices and successively refine this set until we have a discrete universal set.

3.4.1 Two-Level Unitary Matrices

Consider a quantum gate U that acts on n qubits. Letting $d = 2^n$, this gate acts on a d-dimensional Hilbert space, and so can be written in the computational basis as a $d \times d$ matrix. We define a two-level unitary operation to be one that acts nontrivially on only a single qubit. We claim that we can find two level unitary matrices U_{d-1} to U_1 such that $U_{d-1}U_{d-2}\ldots U_2U_1U = V_1$ is a matrix with a one as the top left element and all zeros in the rest of the first row and column [6]. We repeat this process so that

$$V_kV_{k-1}\ldots V_1U = I, \tag{3.37}$$

and so

$$U = V_1^\dagger V_2^\dagger \ldots V_k^\dagger. \tag{3.38}$$

Rather than prove this in general, we explicitly construct the 3×3 case. Even though quantum gates can never have an odd number of rows and columns, the 3×3 case is simpler than the 4×4 case, and is the first nontrivial example of the decomposition process we are demonstrating. Thus, while our example is not a realistic one for quantum computation, it contains the same fundamental steps and is easier to follow.

Consider the matrix

$$U = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & j \end{pmatrix} \tag{3.39}$$

where the elements of U satisfy the unitarity condition and are in general complex. We claim that we can write $U_3U_2U_1U = I$ for two level unitary matrices U_3, U_2, and U_1. Let us first consider U_1. Our goal is to make the entry $[U_1U]_{21}$ zero; if $b = 0$, then, we can simply set $U_1 = I$. Otherwise, let us choose a two level unitary matrix of the following form:

$$U_1 = \begin{pmatrix} \alpha & \beta & 0 \\ \gamma & \delta & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.40}$$

Forming the product U_1U, we get

$$\begin{aligned} U_1U &= \begin{pmatrix} \alpha & \beta & 0 \\ \gamma & \delta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & j \end{pmatrix} \\ &= \begin{pmatrix} (\alpha a + \beta b) & (\alpha d + \beta b) & (\alpha g + \beta h) \\ (\gamma a + \delta b) & (\gamma d + \delta e) & (\gamma g + \delta h) \\ c & f & j \end{pmatrix} \\ &= \begin{pmatrix} a' & d' & g' \\ b' & e' & h' \\ c' & f' & j' \end{pmatrix}. \end{aligned} \tag{3.41}$$

Now, we want $b' = \gamma a + \delta b = 0$. Observe that we can accomplish this by setting

$$\begin{aligned} \gamma &= \frac{b}{\sqrt{|a|^2 + |b|^2}} \\ \delta &= \frac{-a}{\sqrt{|a|^2 + |b|^2}}. \end{aligned} \tag{3.42}$$

Since U_1 must also be unitary, these choices for γ and δ force

$$\beta = \frac{b^*}{\sqrt{|a|^2 + |b|^2}}. \tag{3.43}$$

For reasons that will become clear shortly, we also choose to set

$$\alpha = \frac{a^*}{\sqrt{|a|^2 + |b|^2}}. \tag{3.44}$$

Next, we consider U_2U_1U. We would like the lower left corner entry of this matrix product to be zero. If it is already, we simply set

$$U_2 = \begin{pmatrix} a'^* & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.45}$$

If not, we choose a two level unitary matrix of the form

$$U_2 = \begin{pmatrix} \alpha' & 0 & \beta' \\ 0 & 1 & 0 \\ \gamma' & 0 & \delta' \end{pmatrix}. \tag{3.46}$$

Multiplying the matrices out, we obtain, in much the same way as above,

$$\begin{aligned}
\alpha' &= \frac{a'^*}{\sqrt{|a'|^2 + |c'|^2}} \\
\beta' &= \frac{c'^*}{\sqrt{|a'|^2 + |c'|^2}} \\
\gamma' &= \frac{c'}{\sqrt{|a'|^2 + |c'|^2}} \\
\delta' &= \frac{-a'}{\sqrt{|a'|^2 + |c'|^2}}
\end{aligned} \tag{3.47}$$

Thus, we obtain

$$U_2 U_1 U = \begin{pmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{pmatrix}. \tag{3.48}$$

Since all of U_2, U_1, and U are unitary, their product is as well. In order to fulfill the unitarity condition, $d'' = g'' = 0$, so that

$$U_2 U_1 U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{pmatrix}. \tag{3.49}$$

Note that our somewhat arbitrary choice of α and α' above was contrived to give us a one in the upper left hand corner of this product. Now, we need simply set

$$U_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e''^* & h''^* \\ 0 & f''^* & j''^* \end{pmatrix} \tag{3.50}$$

and we will obtain, as desired, $U_3 U_2 U_1 U = I$, so that $U = U_1^\dagger U_2^\dagger U_3^\dagger$. We have therefore decomposed a general 3×3 unitary matrix into a product of two level unitary matrices. In terms of the more general expressions in equations (3.37) and (3.38), we have found that $V_1 = U_2 U_1$ and $V_2 = U_3$. Recall that V_1 is a matrix such that the product $V_1 U$ has a one in the upper left entry and zeros in the rest of the first row and column. Looking at equation (3.49), this is clearly the case. With this choice of V_2, we also have that $U = V_1^\dagger V_2^\dagger$, as desired. This algorithm clearly generalizes to higher dimensions. For example, in the 4×4 case, the matrix $V_1 U$ would again have a one in the upper left entry and zeros in the rest of the first row and column. Acting recursively, we would the find a matrix V_2 such that the non-trivial 3×3 submatrix of $V_1 U$ would take on the same structure in the product $V_2 V_1 U$, and so on until we have fully decomposed the original matrix.

3.4.2 Single Qubit and Control-NOT Gates

Now that we have shown that any arbitrary quantum gate can be implemented by two level unitary matrices, we continue on towards a discrete universal gate set by showing that an arbitrary two level unitary operation can be implemented using only single qubit gates and the control-NOT gate.

Suppose U is a two level unitary operation. Let $\tilde{U}$ be the nontrivial 2×2 submatrix of U. We will prove that U can be implemented by single qubit gates and the control-NOT gate by explicit construction. Let $|s\rangle$ and $|t\rangle$ be the two computational basis states one which U acts nontrivially . Let us define $|g_1\rangle = |s\rangle$ and $|g_m\rangle = |t\rangle$. The states $|g_2\rangle$ and $|g_{m-1}\rangle$ are defined such that any two consecutive states $|g_i\rangle$ and $|g_{i+1}\rangle$ differ only by one bit; in other words, the states $\{|g_i\rangle\}$ form a Gray code from $|s\rangle$ to $|t\rangle$. As an example of this, let us form a Gray code from $|0110\rangle$ to $|1000\rangle$. We have

$$\begin{aligned} |g_1\rangle &= |0110\rangle \\ |g_2\rangle &= |1110\rangle \\ |g_3\rangle &= |1100\rangle \\ |g_4\rangle &= |1000\rangle \end{aligned} \tag{3.51}$$

The idea of our construction is this: begin with $|g_1\rangle$. By assumption, it differs from $|g_2\rangle$ by a single bit. We thus swap these two states by using a controlled bit flip with its target being the bit where $|g_1\rangle$ and $|g_2\rangle$ differ. The bit flip will be conditioned by the requirement that all the other bits of $|g_1\rangle$ and $|g_2\rangle$ must be the same. We iterate this process until we have transformed $|g_{m-2}\rangle$ into $|g_{m-1}\rangle$. At this point, we performed a controlled $\tilde{U}$ operation with its target being the qubit where $|g_{m-1}\rangle$ and $|g_m\rangle$ differ conditioned on all the other qubits being set. We then proceed to undo all the swaps we have made.

As an example of this procedure, consider the three-qubit transformation

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 & 0 & 0 & b \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & c & 0 & 0 & 0 & 0 & d \end{pmatrix}, \tag{3.52}$$

where

$$\tilde{U} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{3.53}$$

is an arbitrary unitary matrix. We note that U acts nontrivially on the basis states

$|010\rangle$ and $|111\rangle$, on which it has the action

$$\begin{aligned} U|010\rangle &= a|010\rangle + c|111\rangle \\ U|111\rangle &= b|010\rangle + d|111\rangle \,. \end{aligned} \tag{3.54}$$

The Gray code states for this gate are given by

$$\begin{aligned} |g_1\rangle &= |010\rangle \\ |g_2\rangle &= |011\rangle \\ |g_3\rangle &= |111\rangle \,. \end{aligned} \tag{3.55}$$

Based on our procedure defined above, the circuit implementing U is shown in Figure 6. Consider the action of this circuit on a register in the state $|010\rangle$, so

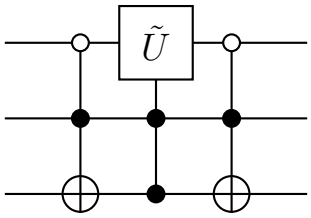

Figure 6: A circuit implementing the three-qubit gate U. Each of the three controlled operations is conditioned on two control qubits.

that the top wire in Figure 6 carries a qubit in the $|0\rangle$ state, the middle wire a $|1\rangle$, and the bottom wire a $|0\rangle$. The first operation is a control-NOT gate conditioned on the top qubit being a $|0\rangle$ and the middle qubit being a $|1\rangle$. Since both the top and middle qubits do in fact have these values, we invert the state of the bottom qubit, leaving us in the state $|011\rangle$.

The second gate in Figure 6 is a controlled $\tilde{U}$ operation on the top qubit, conditioned on the both middle and bottom qubits being $|1\rangle$. Since both are in the $|1\rangle$, state, the $\tilde{U}$ operation is applied to the top qubit, so that after two operations we are in the state $a|011\rangle + c|111\rangle$.

Finally, the third operation is another control-NOT gate with the same conditioning as before. Thus, we finally obtain the state $a|010\rangle + c|111\rangle$, which is exactly what we found before in equation (3.54).

A similar line of reasoning can be used to find the action of the circuit in Figure 6 on the state $|111\rangle$:

$$|111\rangle \rightarrow |111\rangle \rightarrow b|011\rangle + d|111\rangle \rightarrow b|010\rangle + d|111\rangle \,. \tag{3.56}$$

Likewise, it is straightforward to show that any other state passes through this circuit unchanged. Thus, the circuit in Figure 6 reproduces the operation of U.

We have now shown how to implement an arbitrary two level unitary gate using only controlled operations. Recall, though, that in Section 3.3, we showed that an arbitrary controlled operation may be implemented using only single qubit gates and the control-NOT gate; thus, we have now proved these gates to be universal.

3.4.3 A Discrete Universal Set

We are now finally in a position to provide a discrete set of quantum gates that is universal for quantum computation. Our set will consist of the control-NOT gate, the Hadamard gate H, and the $\pi/8$ gate T. It is also common to include the phase gate in this set since its inclusion is natural when considering fault-tolerant gate constructions; however, since the phase gate S is simply T^2, we do not include it here.

Our set of gates will be an approximate universal set. Since the space of all unitary transformations is continuous, we obviously cannot truly construct it with a finite number of transformations. However, we will see that our universal set can approximate any unitary transformation to arbitrary accuracy.

Consider the T gate and the combination HTH. From the definition of the rotation operators in equation (3.10), it is clear that the T gate is, up to a phase factor, a rotation by $\pi/4$ radians about the z axis. Similarly, HTH is a rotation by $\pi/4$ radians about the x axis, up to a phase, as shown below:

$$\begin{aligned} HTH &= \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} (1+e^{i\pi/4}) & (1-e^{i\pi/4}) \\ (1-e^{i\pi/4}) & (1+e^{i\pi/4}) \end{pmatrix} \end{aligned} \tag{3.57}$$

Since

$$\begin{aligned} 1+e^{i\pi/4} &= 1+\cos\frac{\pi}{4}+i\sin\frac{\pi}{4} = 2\cos^2\frac{\pi}{8}+2i\sin\frac{\pi}{8}\cos\frac{\pi}{8} \\ &= 2e^{i\pi/8}\cos\frac{\pi}{8} \end{aligned} \tag{3.58}$$

and

$$\begin{aligned} 1-e^{i\pi/4} &= 1-\cos\frac{\pi}{4}-i\sin\frac{\pi}{4} = 2\sin^2\frac{\pi}{8}-2i\sin\frac{\pi}{8}\cos\frac{\pi}{8} \\ &= -2ie^{i\pi/8}\sin\frac{\pi}{8}, \end{aligned} \tag{3.59}$$

we have

$$HTH = e^{i\pi/8}\begin{pmatrix} \cos\frac{\pi}{8} & -i\sin\frac{\pi}{8} \\ -i\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix} = R_x\left(\frac{\pi}{4}\right). \tag{3.60}$$

Now, let us consider the composition of these two rotations:

$$\begin{aligned} e^{-i\pi Z/8}e^{-i\pi X/8} &= \left(\cos\frac{\pi}{8} - iZ\sin\frac{\pi}{8}\right)\left(\cos\frac{\pi}{8} - iX\sin\frac{\pi}{8}\right) \\ &= \cos^2\frac{\pi}{8} - i\left[(X+Z)\cos\frac{\pi}{8} + Y\sin\frac{\pi}{8}\right]\sin\frac{\pi}{8}. \end{aligned} \tag{3.61}$$

Comparison with equation (3.11) shows that this is a rotation about the axis $\hat{n} = (\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8})$ through an angle ϕ defined by $\cos\frac{\phi}{2} = \cos^2\frac{\pi}{8}$. Though we do not present the proof, this ϕ can be shown to be an irrational multiple of 2π [7].

We would like to be able to represent a rotation through any angle θ by repeated application of $R_n(\phi)$. We note that since ϕ is an irrational multiple of 2π, $\phi = 2\lambda\pi$ where λ is irrational. For any arbitrary θ, there is an integer k such that

$$e^{ik\pi\lambda} \approx e^{i\theta}, \tag{3.62}$$

since phase factors are defined modulo 2π. Thus, $R_n(\phi)$ can be used to approximate a rotation of any angle. Now, consider the product $HR_n(\phi)H$. Using the fact that $H = \frac{1}{\sqrt{2}}(X+Z)$ and that H is Hermitian as well as unitary, we have

$$HR_n(\phi)H = \cos^2\frac{\pi}{8} - i\left[(X+Z)\cos\frac{\pi}{8} + HYH\sin\frac{\pi}{8}\right]\sin\frac{\pi}{8}. \tag{3.63}$$

Since

$$\begin{aligned} HYH &= \frac{1}{2}(X+Z)Y(X+Z) = \frac{1}{2}(X+Z)(-iZ+iX) \\ &= \frac{1}{2}(-Y-Y) = -Y, \end{aligned} \tag{3.64}$$

we have

$$HR_n(\phi)H = \cos^2\frac{\pi}{8} - i\left[(X+Z)\cos\frac{\pi}{8} - Y\sin\frac{\pi}{8}\right]\sin\frac{\pi}{8}, \tag{3.65}$$

which is a rotation about the $\hat{m} = (\cos\frac{\pi}{8}, -\sin\frac{\pi}{8}, \cos\frac{\pi}{8})$ axis, where $\hat{m}$ and $\hat{n}$ are not parallel. Up to a phase shift, any single qubit gate can be represented as a product of rotations about two nonparallel axes, and so any single qubit gate can be represented by an appropriate product of $R_n(\theta)$ and $R_m(\theta)$. Since we can approximate these two general rotations using only the Hadamard and $\pi/8$ gates, we need only H and T to implement any arbitrary single qubit gate. With the results of the previous sections in mind, then, we have constructed a universal set of gates. Any quantum circuit can be approximated to arbitrary accuracy by the Hadamard, $\pi/8$, and control-NOT gates.

4 Quantum Algorithms

4.1 Introduction

The power of quantum computation becomes most evident in the study of quantum algorithms. Strictly speaking, a quantum computer is no more powerful than a classical computer; there is no function computable by a quantum computer that is not also computable by a classical computer. There are, however, tasks which a quantum computer can perform significantly faster than a classical computer is thought to be able to, using what is known as quantum parallelism.

The idea behind quantum parallelism is simple. Since a register of n qubits can be prepared in a superposition of all its basis states, and can therefore represent simultaneously all numbers between 0 and $2^n - 1$, any unitary transformation acting on this register must transform every one of these basis states with a single operation. After a single application of the unitary transformation, the quantum register is left in a superposition of all possible answers. This seems too good to be true, and in many ways it is; a measurement of the register must, of course, yield only a single result, and in the process destroys the superposition. This collapse would seem to argue that nothing has been gained over a classical computation of the same function. The known quantum algorithms, however, cleverly take advantage of the fact that the amplitudes of the superposition of the result register may be adjusted using quantum interference in order to produce their remarkable results.

The general pattern for quantum algorithms is thus clear: first, a quantum register is prepared in an even superposition, in which every state has the same coefficient, of its basis states. Next, some function is evaluated on this superposition using a unitary transformation. Finally, the register is interfered by means of a Fourier or Hadamard transform, and is then measured to obtain a result.

In this section, we present the major known quantum algorithms. Deutsch's algorithm and its extension, the Deutsch-Jozsa, algorithm have no known applications but are useful as introductions to the more complicated quantum algorithms. Grover's search algorithm speeds up the search of an unstructured database of N elements by a factor of $\sqrt{N}$ over a classical search, and has many applications. Finally, Shor's factoring algorithm, an application of the more general quantum order-finding algorithm, can determine the prime factors of a composite number much faster than the best known classical algorithm, and has important ramifications for cryptography.

4.2 Deutsch's Algorithm

Deutsch's problem, first proposed and solved in Deutsch's seminal paper [3], was the first example of a problem that could be solved faster on a quantum computer than on a classical computer. While it has no known applications, both Deutsch's algorithm and the Deutsch-Jozsa algorithm are important from a historical standpoint,

and as simple introductions to quantum algorithms. Here, we present a simplified version of the algorithm presented in Cleve *et al.* [8] and Nielsen and Chuang [1].

Deutsch's problem supposes the existence of a black box that evaluates some function $f : \{0, 1\} \mapsto \{0, 1\}$. There are four such functions: two constant functions,

$$f(0) = f(1) = 0 \quad \text{and} \quad f(0) = f(1) = 1, \tag{4.1}$$

and two so-called balanced functions,

$$f(0) = 0, f(1) = 1 \quad \text{and} \quad f(0) = 1, f(1) = 0. \tag{4.2}$$

We would like to determine which of these two categories of functions a given black box falls into. Classically, the only way to tell would be to evaluate both $f(0)$ and $f(1)$; even though we are not asking for the values explicitly, both evaluations are necessary to determine the global function property. A quantum computer, however, can determine this global property with only a single function evaluation, using Deutsch's algorithm. A quantum network depiction of Deutsch's algorithm is shown in Figure 7.

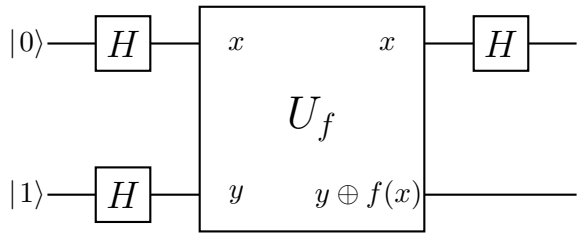

Figure 7: A quantum circuit to implement Deutsch's algorithm.

As shown in the figure, we begin with a two-qubit register initialized to the state $|01\rangle$. Each of these qubits is passed through a Hadamard gate, putting the register into the state

$$|\psi\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right). \tag{4.3}$$

Next, we must evaluate the function f. This evaluation is accomplished by means of the U_f operator, which some have called an f-control-NOT gate. This two-qubit gate inverts the second of its input qubits only if the value of f evaluated on its first input qubit is 1; thus, it performs the transformation

$$|x\rangle\,|y\rangle \xrightarrow{U_f} |x\rangle\,|y \oplus f(x)\rangle\,, \tag{4.4}$$

where $\oplus$ denotes addition modulo 2.[3] We also note that

$$\begin{aligned} U_f\left[|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right] &= \frac{1}{\sqrt{2}}|x\rangle\left[|0\oplus f(x)\rangle-|1\oplus f(x)\rangle\right] \\ &= \begin{cases} |x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(x)=0 \\ |x\rangle\left(\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right), & f(x)=1 \end{cases} \\ &= (-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right). \end{aligned} \tag{4.5}$$

After passing through the U_f gate, then, the circuit is in the state

$$\begin{aligned} U_f|\psi\rangle &= (-1)^{f(x)}\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \\ &= \begin{cases} \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)=f(1)=0 \\ \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)=0,\ f(1)=1 \\ -\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)=f(1)=1 \\ -\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)=1,\ f(1)=1 \end{cases}, \end{aligned} \tag{4.6}$$

which can be more concisely summarized as

$$U_f|\psi\rangle = \begin{cases} \pm\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)=f(1) \\ \pm\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)\neq f(1) \end{cases}. \tag{4.7}$$

Now, the first qubit is passed through another Hadamard gate, which produces the state

$$\begin{cases} \pm|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)=f(1) \\ \pm|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right), & f(0)\neq f(1) \end{cases}. \tag{4.8}$$

Now, noting that

$$f(0)\oplus f(1) = \begin{cases} 0, & f(0)=f(1) \\ 1, & f(0)\neq f(1) \end{cases}, \tag{4.9}$$

we can rewrite the final state of the circuit as

$$|\psi_{out}\rangle = \pm|f(0)\oplus f(1)\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right). \tag{4.10}$$

[3] Addition modulo 2 produces the following results:

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

Thus, a measurement of the first qubit will give the value of $f(0) \oplus f(1)$; if this value is 0, f is constant, and otherwise is balanced. Thus, with only a single evaluation, we have determined a global property of f.

Deutsch's algorithm, while giving a 50% speedup over the fastest classical algorithm for the same task, is not a particularly stunning result. It contains the seeds,however, from which the other known quantum algorithms grow. Deutsch's algorithm first throws its quantum register into an even superposition and then proceeds to evaluate a function and interfere the outcomes to obtain its final result. We will see this pattern again in each of the other algorithms presented in this section. Next, we present the n-qubit extension of Deutsch's algorithm, which, while similar to Deutsch's algorithm, has a more impressive result.

4.3 The Deutsch-Jozsa Algorithm

The Deutsch-Jozsa algorithm is the n-qubit generalization of Deutsch's algorithm, first published in a paper by Deutsch and Jozsa in 1992 [9]. Again, we present here an improved version of the algorithm from Cleve *et al.* [8] and Nielsen and Chuang [1].

Suppose we have a function f, now defined such that $f : \{0,1\}^n \mapsto \{0,1\}$ for even n, that is guaranteed to be either constant or balanced[4]. Our goal is to determine with certainty which category f falls into. Classically, this would require, in the worst case, $2^{n-1}+1$ evaluations of f, since we could get 2^{n-1} 0's before finally getting a 1. Quantum mechanically, however, this global property of f can once again be determined with only a single evaluation of f.

The algorithm proceeds much as Deutsch's algorithm did; a quantum circuit diagram for the algorithm is shown in Figure 8. As before, the input state of $|0\rangle^{\otimes n}|1\rangle$ is first sent through a Hadamard transform. As we have shown before in equation (3.32), a Hadamard transform on a quantum register in the $|0\rangle^{\otimes n}$ state produces an even superposition of all the basis states of the register, so after the application of the Hadamard transform we have the state

$$|\psi\rangle = \sum_{x=0}^{2^n-1} \frac{|x\rangle}{\sqrt{2^n}} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right), \tag{4.11}$$

where we have acted with the Hadamard transform on the $|1\rangle$ qubit as well. Now, once again, we use the U_f operator to evaluate the function f, which gives

$$U_f|\psi\rangle = \sum_{x=0}^{2^n-1} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right), \tag{4.12}$$

[4]A constant n-qubit function is one that gives the same result for any input, and a balanced n-qubit function is one that returns one result for exactly half of its inputs and a second result for the other half of its inputs.

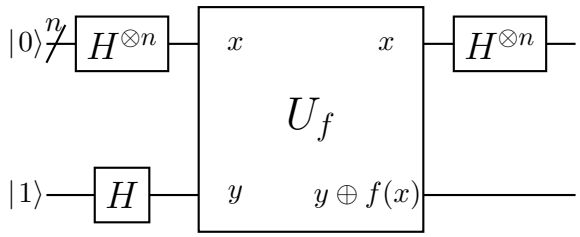

Figure 8: A quantum circuit to implement the Deutsch-Jozsa algorithm. The wire with a / represents an n-qubit bus and is a shorthand notation for n wires.

as we showed before. The result of the evaluation of f on *every* argument from 0 to $2^n - 1$ is now stored in the first register. A direct measurement of this register would, of course, produce only a single result. However, by interfering the qubits, we may determine the global property of f we desire. To that end, we perform a second Hadamard transform on the first register. Recall from equation (3.31) that the Hadamard transform on some register $|\phi\rangle$ in general produces the state

$$H^{\otimes n} |\phi\rangle = \sum_k \frac{(-1)^{\phi \cdot k} |k\rangle}{\sqrt{2^n}}. \tag{4.13}$$

Using this, we can write the state of our circuit after the second Hadamard transform as

$$\sum_k \sum_x \frac{(-1)^{x \cdot k \oplus f(x)} |k\rangle}{2^n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right). \tag{4.14}$$

Now, we measure the first register. It turns out that if $k = 0$, we know with certainty that f is constant; if $k \neq 0$, then f is balanced. To see why this is true, consider equation (4.12). If f is constant for all values of its argument, this equation can be rewritten as

$$U_f |\psi\rangle = \pm H^{\otimes n} |0\rangle^{\otimes n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right). \tag{4.15}$$

Since the Hadamard transform is Hermitian as well as unitary (it being both symmetric and real), the second Hadamard transform on this state produces

$$|0\rangle^{\otimes n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right). \tag{4.16}$$

Thus, when every qubit in the first register is 0, f is constant. If f is balanced, we cannot make the simplification in equation (4.15), and at least one qubit in the first register will be nonzero.

Amazingly, then, we have achieved an exponential speedup over the best classical algorithm for solving this problem, reducing the worst case number of function evaluations from $2^{n-1}+1$ to a single evaluation.

4.4 Grover's Algorithm

Now that we have seen some examples of simple quantum algorithms, we move on to an algorithm with practical application. Suppose we have an n qubit database of unsorted data, and we wish to find elements that match a certain criterion. Since we are assuming that there is no structure to this database, the best we can do classically is simply to search through the entire database until we come across the element we seek. For an $N = 2^n$ element database, this will require $\mathcal{O}(N)$ queries to the database. However, the quantum algorithm due to Grover can speed up the search for this "needle in a haystack" [10], allowing us to find the desired element in only $\mathcal{O}(\sqrt{N})$ steps. We begin by presenting the algorithm itself, and follow with a discussion of why the algorithm works. Finally, we present an alternate generalized form of the algorithm.

4.4.1 Procedure

As with the Deutsch-Jozsa algorithm, Grover's algorithm begins by applying a Hadamard transform to the $|0\rangle^{\otimes n}$ state, giving us the familiar even superposition

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle\ . \tag{4.17}$$

We next apply several instances of the so-called Grover operator G.

The first part of the Grover operator is a black box known as an oracle that can recognize the desired elements of our database.[5]This oracle O marks such states by giving them local phase shifts of π radians; the internal workings of the oracle are not specified by the algorithm. We can write the action of the oracle as

$$O|x\rangle = (-1)^{\delta_{x\tau}}|x\rangle\ , \tag{4.18}$$

where $|\tau\rangle$ represents a solution of the search problem. We also note that

$$O = I - 2|\tau\rangle\langle\tau| \tag{4.19}$$

[5]It may seem as if the existence of an oracle that knows that answer to the search problem obviates the need for the rest of the search algorithm. There are instances, however, where it is possible to recognize the solution of a search problem without knowing the solution, especially when using quantum parallelism. Consider what happens when the oracle operates on a quantum register and one of the states in the register is the target state of search problem. The oracle will mark this state, but without the rest of the search algorithm, there is no way to tell which state has been marked since all the basis states of the register are entangled. In this case, even though the oracle can recognize the solution, it cannot be used to find the solution by itself.

since

$$\begin{aligned}(I - 2\,|\tau\rangle\,\langle\tau|\,)\,|x\rangle &= |x\rangle - 2\,|\tau\rangle\,\langle\tau\,|x\rangle = |x\rangle - 2\delta_{x\tau}\,|\tau\rangle \\ &= \begin{cases} |\tau\rangle - 2\,|\tau\rangle = -\,|\tau\rangle\,, & x = \tau \\ |\tau\rangle\,, & x \neq \tau \end{cases} \\ &= (-1)^{\delta_{x\tau}}\,|x\rangle\,. \end{aligned} \tag{4.20}$$

The next part of the Grover operator is a Hadamard transform $H^{\otimes n}$. We then give every element of the database except for the $|0\rangle$ element a π phase shift. Just as we can write the action of the oracle as in equation (4.19), we can write this step as an application of the operator

$$2\,|0\rangle\,\langle 0| - I, \tag{4.21}$$

where the signs of the two halves of the operator are reversed since we are shifting the phase of every element *except* the 0^{th} element; that is, we are performing the transformation $|x\rangle \rightarrow -(-1)^{\delta_{x0}}\,|x\rangle$. Finally, the Grover operator contains a second Hadamard transform. The full Grover operator thus can be written

$$G = H^{\otimes n}(2\,|0\rangle\,\langle 0| - I)H^{\otimes n}O. \tag{4.22}$$

In the next section, we discuss exactly what the operator does; we also calculate how many times it must be applied.

After applying the Grover operator, all we must do to complete the algorithm is measure the register; with an arbitrarily high probability, the measured state will be one of the desired target states.

4.4.2 Inversion about the Mean

The question remains as to what the Grover iteration actually does, and why it performs a database search. Consider that the essential operation of Grover's algorithm is to take some initial state $|\gamma\rangle$ and transform it into some target state $|\tau\rangle$. For the algorithm to work, then, repeated applications of the Grover operator must change the one vector into the other. Let us define a two dimensional space spanned by vectors $|\alpha\rangle$ and $|\beta\rangle$ where $|\beta\rangle$ is an even superposition of the solutions to our search problem and $|\alpha\rangle$ is an even superposition of the nonsolutions; let us furthermore suppose that there are N states in our database and M solutions. Normalizing these two vectors, we have

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{nonsolutions} |x\rangle \tag{4.23}$$

and

$$|\beta\rangle = \frac{1}{\sqrt{M}} \sum_{solutions} |x\rangle\,. \tag{4.24}$$

We can reexpress the initial even superposition in terms of these new vectors (for convenience, we make the definition $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$). Since

$$|\psi\rangle = \frac{1}{\sqrt{N}} \left(\sum_{nonsolutions} |x\rangle + \sum_{solutions} |x\rangle \right), \tag{4.25}$$

we have

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{N}} \left(\sqrt{N-M}\,|\alpha\rangle + \sqrt{M}\,|\beta\rangle \right) \\ &= \sqrt{\frac{N-M}{N}}\,|\alpha\rangle + \sqrt{\frac{M}{N}}\,|\beta\rangle\,. \end{aligned} \tag{4.26}$$

Now we can see what the Grover operator does to the vector $|\psi\rangle$ in this space. The first part of the Grover operator is an application of the oracle. Recall that the oracle gives a π phase shift to all solutions of the search problem; thus, by definition,

$$O\,|\psi\rangle = O\left(\sqrt{\frac{N-M}{N}}\,|\alpha\rangle + \sqrt{\frac{M}{N}}\,|\beta\rangle \right) = \sqrt{\frac{N-M}{N}}\,|\alpha\rangle - \sqrt{\frac{M}{N}}\,|\beta\rangle\,. \tag{4.27}$$

This amounts to a reflection of $|\psi\rangle$ about the $|\alpha\rangle$-axis. The other part of the Grover operator, $H^{\otimes n}(2\,|0\rangle\,\langle 0| - I)H^{\otimes n}$, can be rewritten in our current notation as $2\,|\psi\rangle\,\langle\psi| - I$. Comparing this form to the form of the oracle in equation (4.19), we can see that this second part of the Grover operator is also a reflection in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$, this time about the original vector $|\psi\rangle$. This operation is often called inversion about the mean; consider its action on a general state $\sum_k \alpha_k\,|k\rangle$:

$$\begin{aligned} (2\,|\psi\rangle\,\langle\psi| - I)\left(\sum_k \alpha_k\,|k\rangle \right) &= \sum_k \left[2\alpha_k\,|\psi\rangle\,\langle\psi|\,k\rangle - \alpha_k\,|k\rangle \right] \\ &= \sum_k \left[\frac{2\alpha_k}{N} \sum_i \sum_j |i\rangle\,\langle j|\,k\rangle - \alpha_k\,|k\rangle \right] \\ &= \sum_k \left[\frac{2\alpha_k}{N} \sum_i \sum_j \delta_{jk}\,|i\rangle - \alpha_k\,|k\rangle \right] \\ &= \sum_k \left[\frac{2\alpha_k}{N} \sum_i |i\rangle \right] - \sum_k \left[\alpha_k\,|k\rangle \right]. \end{aligned} \tag{4.28}$$

We now swap the summations and define the average value of α to be $\langle\alpha\rangle \equiv \sum_k \alpha_k/N$ to obtain

$$\sum_i \left[\sum_k \frac{2\alpha_k}{N} \right] |i\rangle - \sum_k \alpha_k\,|k\rangle = \sum_i 2\langle\alpha\rangle\,|i\rangle - \sum_k \alpha_k\,|k\rangle\,. \tag{4.29}$$

Finally, we relabel the dummy index i by k, obtaining a final state of

$$\sum_k \left[-\alpha_k + 2\langle\alpha\rangle\right] |k\rangle \ . \tag{4.30}$$

The two halves of the Grover operator thus each perform a reflection in the same plane, and since the product of two reflections is a rotation, the Grover operator effectively rotates the starting state towards the vector of solutions, as is shown in Figure 9. Continued applications of the Grover operator rotate the state closer and

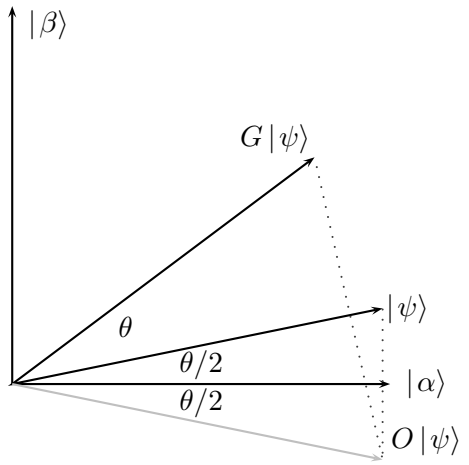

Figure 9: The action of a single application of the Grover operator on the state $|\psi\rangle$.

closer to the solution vector, while remaining in the same space. So how many times must we apply the Grover operator to have a good chance of finding a solution?

Let us make the definition that

$$\cos\frac{\theta}{2} \equiv \sqrt{\frac{N-M}{N}}. \tag{4.31}$$

This defines the triangle in Figure 10, such that

$$\sin\frac{\theta}{2} = \sqrt{\frac{M}{N}}. \tag{4.32}$$

We can therefore reexpress $|\psi\rangle$ as

$$|\psi\rangle = \cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle \ . \tag{4.33}$$

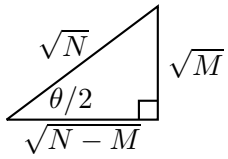

Figure 10: The triangle defined by equation (4.31).

Now, as can be seen from Figure 9, the Grover operator rotates $|\psi\rangle$ by exactly θ radians; the oracle's reflection effectively rotates the state by $-\theta$ radians and the subsequent reflection about $|\psi\rangle$ rotates the state by 2θ radians.

We can now quantify the number of times we must apply the Grover operator to our system to rotate $|\psi\rangle$ close to $|\beta\rangle$. Consider again equation (4.26); our goal is to rotate this state close to the vector $|\beta\rangle$. We now show that rotating through an angle $\phi = \text{Cos}^{-1}\sqrt{M/N}$ will take $|\psi\rangle$ to $|\beta\rangle$.

First, we note that ϕ is the complement of $\theta/2$. Thus, we can easily write down a rotation operator in the $\{|\alpha\rangle, |\beta\rangle\}$ basis. We have

$$U_R = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{M}{N}} & -\sqrt{\frac{N-M}{N}} \\ \sqrt{\frac{N-M}{N}} & \sqrt{\frac{M}{N}} \end{pmatrix}. \tag{4.34}$$

We now operate on $|\psi\rangle$, which in this basis is

$$|\psi\rangle = \begin{pmatrix} \sqrt{\frac{N-M}{N}} \\ \sqrt{\frac{M}{N}} \end{pmatrix} \tag{4.35}$$

to obtain

$$U_R|\psi\rangle = \begin{pmatrix} \sqrt{\frac{M}{N}}\sqrt{\frac{N-M}{N}} - \sqrt{\frac{M}{N}}\sqrt{\frac{N-M}{N}} \\ \left(\frac{N-M}{N}\right) + \frac{M}{N} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |\beta\rangle, \tag{4.36}$$

as claimed.

Since every application of the Grover operator rotates $|\psi\rangle$ through θ radians, we must apply it

$$R = \frac{\text{Cos}^{-1}\sqrt{\frac{M}{N}}}{\theta} \tag{4.37}$$

times to rotate $|\psi\rangle$ to $|\beta\rangle$. In actuality, we require R to be an integer, and so R is actually the integer closest to the value in equation (4.37). We now simplify this

rather unwieldy expression by noting that, since the maximum value of Cos^{-1} is $\pi/2$, the upper bound on R is

$$R \leq \left\lceil \frac{\pi}{2\theta} \right\rceil . \tag{4.38}$$

For $M \leq N/2$ (we will explain this restriction shortly), we have, using equation (4.32),

$$\frac{\theta}{2} \geq \sin \frac{\theta}{2} = \sqrt{\frac{M}{N}} \tag{4.39}$$

so that $\theta \geq 2\sqrt{M/N}$ and

$$R \leq \left\lceil \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rceil . \tag{4.40}$$

Thus, as stated earlier, Grover's algorithm will run in $\mathcal{O}(\sqrt{N})$ time.

Now, to see why the above expression was limited to the case of $M \leq N/2$, we derive an expression for $\sin\theta$. First, let us apply a half-angle identity to equation (4.32), giving us

$$\sin \frac{\theta}{2} = \sqrt{\frac{1 - \cos\theta}{2}} = \sqrt{\frac{M}{N}}. \tag{4.41}$$

Simplifying, this equation describes a triangle such that

$$\cos\theta = \frac{N - 2M}{N}. \tag{4.42}$$

Using this triangle, we have

$$\sin\theta = \frac{2\sqrt{M(N-M)}}{N}. \tag{4.43}$$

Examining this expression, we can see that as M increases from $N/2$ to N, θ becomes smaller, and thus R increases. Thus we have the strange behavior that Grover's algorithm performs worse for databases where more than half of the elements are solutions. Odd as this behavior is, we may escape from it simply by adding another N nonsolutions to our database. In this case, there will definitely be more than twice as many nonsolutions as solutions, and Grover's algorithm will perform optimally, with the same complexity as derived above.

4.4.3 An Alternate Approach

The above discussion of Grover's algorithm seems intimately connected with the properties of the Hadamard transform, as it appears twice in the Grover operator which is the heart of the algorithm. Grover, however, also published a second, more general version of his algorithm that shows that almost any unitary operator may be used to search a quantum database [11].

Recall that the goal of Grover's algorithm is to take some initial state $|\gamma\rangle$ to a target state $|\tau\rangle$. We have shown how the Grover operator G is able to perform this operation, using the Hadamard transform. Now consider an operator Q, defined as

$$Q = -I_\gamma U^\dagger I_\tau U, \tag{4.44}$$

where

$$I_\phi = I - 2\,|\phi\rangle\,\langle\phi| \tag{4.45}$$

and U is some arbitrary unitary transformation. We now operate with Q on $|\gamma\rangle$:

$$\begin{aligned} Q\,|\gamma\rangle &= -I_\gamma U^\dagger I_\tau U\,|\gamma\rangle \\ &= (2\,|\gamma\rangle\,\langle\gamma| - I)\,U^\dagger\,(I - 2\,|\tau\rangle\,\langle\tau|)\,U\,|\gamma\rangle \\ &= \Big[2\,|\gamma\rangle\,\langle\gamma| - 4\,|\gamma\rangle\,\langle\gamma|\,U^\dagger\,|\tau\rangle\,\langle\tau|\,U - I + 2U^\dagger\,|\tau\rangle\,\langle\tau|\,U\Big]\,|\gamma\rangle\,. \end{aligned} \tag{4.46}$$

Defining

$$U_{\tau\gamma} = \langle\tau\,|U|\,\gamma\rangle\,, \tag{4.47}$$

we have

$$Q\,|\gamma\rangle = \left(1 - 4\,|U_{\tau\gamma}|^2\right)|\gamma\rangle + 2U_{\tau\gamma}\left(U^\dagger\,|\tau\rangle\right). \tag{4.48}$$

This result suggests that we examine Q in the $\left\{|\gamma\rangle\,, U^\dagger\,|\tau\rangle\right\}$ basis. To that end, let us calculate $Q\left(U^\dagger\,|\tau\rangle\right)$:

$$\begin{aligned} Q\left(U^\dagger\,|\tau\rangle\right) &= \left(-I_\gamma U^\dagger I_\tau U\right)\left(U^\dagger\,|\tau\rangle\right) \\ &= (2\,|\gamma\rangle\,\langle\gamma|)\,U^\dagger\,(I - 2\,|\tau\rangle\,\langle\tau|)\,UU^\dagger\,|\tau\rangle \\ &= \Big[2\,|\gamma\rangle\,\langle\gamma|\,U^\dagger - 4\,|\gamma\rangle\,\langle\gamma|\,U^\dagger\,|\tau\rangle\,\langle\tau| - U^\dagger + 2U^\dagger\,|\tau\rangle\,\langle\tau|\Big]\,|\tau\rangle \\ &= \left(U^\dagger\,|\tau\rangle\right) - 2U^*_{\tau\gamma}\,|\gamma\rangle\,. \end{aligned} \tag{4.49}$$

Thus, we see that Q preserves the space spanned by $\left\{|\psi\rangle\,, U^\dagger\,|\tau\rangle\right\}$, and we can write it in this basis as

$$Q\begin{pmatrix} |\gamma\rangle \\ U^\dagger\,|\tau\rangle \end{pmatrix} = \begin{pmatrix} 1 - 4\,|U_{\tau\gamma}|^2 & 2U_{\tau\gamma} \\ -2U^*_{\tau\gamma} & 1 \end{pmatrix}\begin{pmatrix} |\gamma\rangle \\ U^\dagger\,|\tau\rangle \end{pmatrix}. \tag{4.50}$$

Now, we know that $|U_{\tau\gamma}|^2$, being a normalized probability, is less than one. Assuming that both it and its associated amplitudes $U_{\tau\gamma}$ and $U^*_{\tau\gamma}$ are small, we can view Q as a rotation of roughly $2\,|U_{\tau\gamma}|$ in this space, from $|\gamma\rangle$ to $U^\dagger\,|\tau\rangle$. Thus, using similar logic as above, Q can also search a database, using any convenient unitary transformation.

So how many times must we apply Q to attain a high probability of finding a solution? In any situation of interest, we know that $|\gamma\rangle$ and $U^\dagger\,|\tau\rangle$ are close to orthogonal. If they were not, then the bracket $\langle\tau\,|U|\,\gamma\rangle$ would not be small, and

so there would be a nonnegligible probability that an observation of $U|\gamma\rangle$ would produce $|\tau\rangle$. If this were the case, then by simply conducting several repeated experiments, $|\tau\rangle$ could be readily determined. Thus, the angle between $|\gamma\rangle$ and $U^\dagger|\tau\rangle$ must be roughly $\pi/2$. Since every application of Q rotates the state by roughly $2|U_{\tau\gamma}|$, we must apply Q

$$\begin{aligned} r &= \left(\frac{\pi}{2}\right)\left(\frac{1}{2|U_{\tau\gamma}|}\right) = \frac{\pi}{4|U_{\tau\gamma}|} \\ &= \mathcal{O}\left(\frac{1}{|U_{\tau\gamma}|}\right) \end{aligned} \tag{4.51}$$

times. Thus, we have that

$$|\tau\rangle \approx UQ^{\otimes r}|\gamma\rangle\,. \tag{4.52}$$

Finally, let us recreate the original Grover operator G from Q. Recall that

$$G = H^{\otimes n}\left(2|0\rangle\langle 0| - I\right)H^{\otimes n}\left(I - 2|\tau\rangle\langle\tau|\right), \tag{4.53}$$

which in our present notation is

$$G = U(-I_0)U^\dagger I_\tau, \tag{4.54}$$

using $H^{\otimes n} = U = U^\dagger$. Now, we note that

$$\begin{aligned} U^\dagger G U &= U^\dagger U(-I_0)U^\dagger I_\tau U \\ &= -I_0 U^\dagger I_\tau U \\ &= Q. \end{aligned} \tag{4.55}$$

Thus, the Grover operator G may be written as simply a unitary transformation of the generalized Grover operator Q. We may also recover the complexity of Grover's algorithm, as follows. We have shown that the generalized version of Grover's algorithm is of order $\mathcal{O}(1/|U_{\tau\gamma}|)$. Thus, to recover the standard Grover's algorithm complexity of $\mathcal{O}(\sqrt{N})$. We want

$$\left\langle\tau\left|H^{\otimes n}\right|0\right\rangle = \frac{1}{\sqrt{N}}. \tag{4.56}$$

This is simple to show. First, we have

$$\left\langle\tau\left|H^{\otimes n}\right|0\right\rangle = \langle\tau|\left(\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle\right). \tag{4.57}$$

Since $|\tau\rangle$ is orthonormal to exactly one of the kets in the sum (assuming a single solution to the search problem), this bracket collapses simply to $1/\sqrt{N}$, and we recover the complexity of the original Grover's algorithm.

4.5 Shor's Algorithm

Even though the field of quantum computation was born in 1985 with Deutsch's seminal paper [3], it was not until 1994, when Shor first published fast quantum algorithms for factoring and finding discrete logarithms, that quantum computation became a major field of research. Classically, factoring is a difficult problem; while it has not been proved to be an exponential time problem, the best known classical algorithm, known as the Number Field Sieve, runs in a time of $\mathcal{O}(\exp((\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}))$ for a number N [12]. Amazingly, Shor's algorithm can factor a number in only $\mathcal{O}((\log N)^2(\log\log N)(\log\log\log N))$ steps [13]. The vast difference between these two time complexities can be seen in Figure 11, in which each of these functions is plotted against the logarithm base 2 of N, which gives the number of bits in N when N is represented in binary.

The intense interest generated by the publishing of Shor's factoring algorithm sprang not only from the fact that his algorithm was exponentially faster than the best known classical algorithm, an impressive result in and of itself, but also from a more practical perspective. Since factoring has long been assumed to be a difficult problem for large numbers, it was used as the basis for the RSA cryptosystem, one of the most widespread cryptosystems in use today. Thus, a working quantum computer running Shor's algorithm could quickly and efficiently gain access to huge amounts of encrypted data. Needless to say, this prospect generated widespread interest.

4.5.1 Number Theoretic Background

Shor's algorithm relies on a trick from number theory for its efficiency. It is well known that the problem of factoring a composite number may be mapped into the equivalent problem of finding the order of a periodic function. Following Nielsen and Chuang [1] and Ekert and Jozsa [14], we now present this reduction.

Suppose N is a positive integer and x is some integer relatively prime to N such that $1 \leq x < N$. We define the order of $f(x) = x \mod N$ to be the least r such that $x^r \equiv 1 \mod N$. Finding such an r given x and N is referred to as the order-finding problem, for which there is no known efficient classical algorithm. It is this order-finding problem for which Shor determined an efficient quantum algorithm.

The heart of the reduction of factoring to order finding is the following theorem:

Theorem 4.1 *Let N be a composite number and let x be a nontrivial solution to*

$$x^2 \equiv 1 \mod N \tag{4.58}$$

in the range $1 \leq x < N$, with $\gcd(x, N) = 1$. We define a trivial solution to be $x \equiv \pm 1 \mod N$. Then, at least one of $\gcd(x+1, N)$ or $\gcd(x-1, N)$ is a nontrivial factor of N.

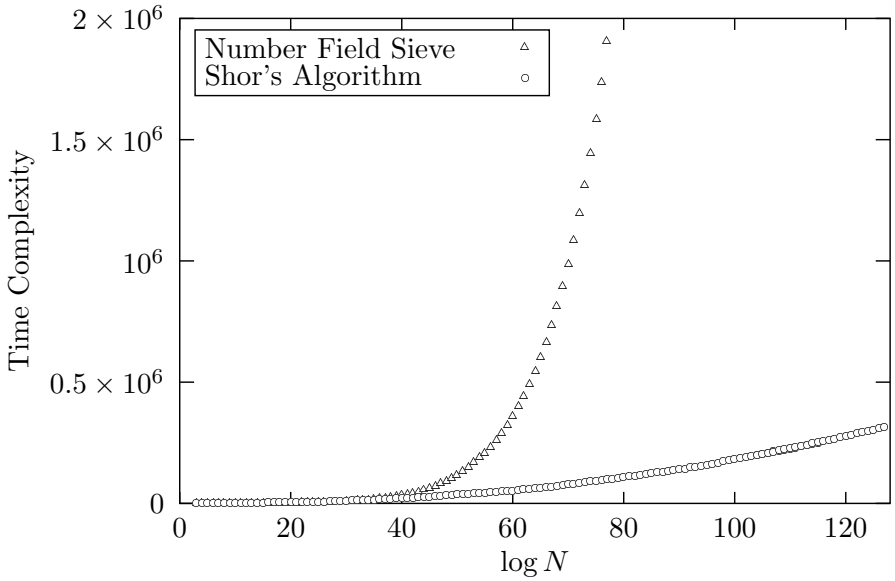

Figure 11: Here, we plot the time complexities of the Number Field Sieve algorithm and Shor's algorithm versus the number of bits in the number we wish to factor. Note that the numbers on the vertical axis have little absolute meaning; in some sense they measure the number of computational steps necessary to complete the algorithms, but the computing device on which these steps are performed is not specified, and so neither is the absolute number of steps. Much more important is the shape of the curves and the relative function values. Comparing these, it is clear that Shor's algorithm is vastly faster than the Number Field Sieve. Most computers today use 128 bit numbers in their encryption schemes; it is clear from this plot that numbers of this length are more than adequate to protect information from today's computers. A quantum computer running Shor's algorithm, however, could easily break such encryption.

Proof Let us begin by rewriting $x^2 \equiv 1 \mod N$ as $x^2 - 1 \equiv 0 \mod N$. It is then clear that N must divide $x^2 - 1 = (x+1)(x-1)$. Thus, N must have a common factor with either $(x+1)$ or $(x-1)$. Recognizing that $-1 \mod N = N - 1$, the condition that x is not a trivial solution becomes

$$1 < x < N - 1, \tag{4.59}$$

which in turn leads to

$$x - 1 < x + 1 < N, \tag{4.60}$$

and so the common factor cannot be N itself. Thus, either $\gcd(x+1, N)$ or $\gcd(x-1, N)$ must be a factor of N.

□

Now let $x \equiv y^{r/2} \mod N$ where r is the order of $y \mod N$. Squaring this expression, we have

$$x^2 \equiv y^r \equiv 1 \mod N, \tag{4.61}$$

where the second equality follows from the definition of the order. It is then clear that one of $\gcd(y^{r/2}+1, N)$ or $\gcd(y^{r/2}-1, N)$ is a nontrivial factor of N. Thus, by finding the order modulo N of a number relatively prime to N, we can find a factor of N. Since there exists a fast classical algorithm to find greatest commons divisors (see Appendix B), an efficient algorithm for order finding can easily be mapped into an efficient algorithm for factoring. In the following section, we present Shor's quantum order finding algorithm.

4.5.2 Order Finding

As shown above, the factoring problem is easily reduced to the problem of finding the order of a periodic function. The heart of Shor's factoring algorithm is thus an efficient order finding algorithm that depends crucially on the properties of the quantum Fourier transform, as we will now show.

Let us define a unitary transformation U_f such that

$$U_f \left|x\right\rangle \left|0\right\rangle = \left|x\right\rangle \left|f(x)\right\rangle, \tag{4.62}$$

where

$$f(x) = y^x \mod N, \tag{4.63}$$

where y is chosen such that $\gcd(y, N) = 1$ so that y and N are relatively prime. This modular exponentiation operator can be implemented efficiently;for our purposes, however, we treat it simply as a black box. Now, let us begin the order finding algorithm by initializing two quantum registers to $\left|0\right\rangle^{\otimes q}$, where q is a power of two satisfying $N^2 \leq q \leq 2N^2$:

$$\left|\psi_0\right\rangle = \left|0\right\rangle^{\otimes \log q} \left|0\right\rangle^{\otimes \log q}. \tag{4.64}$$

This choice of q ensures that the quantum Fourier transform we use momentarily may be computed efficiently. Note that $|\psi_0\rangle$ is a tensor product of $\log q$ qubits because we are here representing numbers in binary, and $\log q$ is the number of bits in the number q. We then apply a Hadamard transform to the first register, obtaining

$$H^{\otimes \log q} |\psi_0\rangle = \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle |0\rangle^{\otimes \log q} = |\psi_1\rangle . \tag{4.65}$$

Now, we apply the U_f operator defined above to this state:

$$U_f |\psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle |f(x)\rangle . \tag{4.66}$$

Now, we measure the second register, obtaining some value y_0. Because the two registers were entangled by the U_f operator, this measurement projects the first register into a superposition of all x such that $f(x) = y_0$. Let us define x_0 to be the smallest such x. We know that $y_0 = y^{x_0} \mod N$. Letting r be the order of f modulo N, we also know that $y_0 = y^{x_0+kr} \mod N$ for $k \in [0, K]$ where K is defined as

$$K = \left\lceil \frac{q - x_0}{r} \right\rceil . \tag{4.67}$$

Incorporating these ideas, the state of the system after the measurement is

$$|\phi\rangle = \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} |x_0 + kr\rangle |y_0\rangle . \tag{4.68}$$

At this point, the second register has served its purpose, and we drop it from our discussion. Now, let us apply an q dimensional quantum Fourier transform to the first register, where the Fourier transform operator is

$$F_q = \frac{1}{\sqrt{q}} \sum_{b,c=0}^{q-1} |c\rangle \, e^{\frac{2\pi i bc}{q}} \, \langle b| . \tag{4.69}$$

This gives us the state

$$\begin{aligned} F_q |\phi\rangle &= \frac{1}{\sqrt{qK}} \sum_{k=0}^{K-1} \sum_{b,c=0}^{q-1} |c\rangle \, e^{\frac{2\pi i bc}{q}} \, \langle b | x_0 + kr \rangle \\ &= \sum_{c=0}^{q-1} \left[\frac{1}{\sqrt{qK}} \sum_{k=0}^{K-1} e^{\frac{2\pi i (x_0+kr)c}{q}} \right] |c\rangle \\ &= \sum_{c=0}^{q-1} \left[\frac{1}{\sqrt{qK}} \left(\sum_{k=0}^{K-1} e^{\frac{2\pi i krc}{q}} \right) e^{\frac{2\pi i x_0 c}{q}} \right] |c\rangle . \end{aligned} \tag{4.70}$$

Using the orthonormality condition for complex Fourier series,

$$\sum_{q=0}^{Q-1} e^{\frac{2\pi iaq}{Q}} e^{\frac{2\pi ibq}{Q}} = Q\delta_{ab}, \tag{4.71}$$

and the fact that $K \approx q/r$, the term in parentheses in equation (4.70) reduces to [14]

$$\frac{q}{r}\delta_{c,0 \mod (q/r)}, \tag{4.72}$$

which says that c must be a multiple of q/r. Let $c = j\frac{q}{r}$. Then we have

$$F_q \left|\phi\right\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{2\pi ijx_0}{r}} \left| j\frac{q}{r} \right\rangle . \tag{4.73}$$

A measurement of this register will now give us some c such that

$$\frac{c}{q} = \frac{\lambda}{r} \tag{4.74}$$

where c and q are known. By using a continued fractions expansion (see Appendix B), we can now determine the period r.

In our discussion of the order finding algorithm above, we have assumed (equation (4.72)) that the Fourier transform generates constructive interference only for those c that are multiples of q/r. This, however, is not strictly the case. In fact, for the algorithm to produce the order of $f(x)$ with high probability, c need only meet the condition [13]

$$-\frac{r}{2} \leq rc \mod q \leq \frac{r}{2}, \tag{4.75}$$

which is equivalent to

$$|rc - dq| \leq \frac{r}{2} \tag{4.76}$$

for some d. Rearranging this expression, we have

$$\left|\frac{c}{q} - \frac{d}{r}\right| \leq \frac{1}{2q}. \tag{4.77}$$

Again, since c and q are known, we may find the period r by again using a continued fractions expansion. As long as this period is even, we know that at least one of $\gcd(y^{r/2}+1, N)$ or $\gcd(y^{r/2}-1, N)$ is a nontrivial factor of N. If r is odd, the algorithm fails, and we must try again with a different choice of y.

4.5.3 An Example: Factoring 15

Here, we illustrate Shor's algorithm with a relatively simple example. We choose to factor $N = 15$, as this is the smallest composite number that is neither even nor the square of a prime. Thus, $N = 15$ is the first nontrivial number to factor.

First, we need to pick some y relatively prime to 15; let $y = 7$. We also need a q that is a power of two and satisfies $15^2 \leq q \leq 2(15^2)$; let $q = 256$. Using the notation above, we begin with the state

$$|\psi_0\rangle = |0\rangle^{\otimes 8} |0\rangle^{\otimes 8}. \tag{4.78}$$

The first step of the algorithm is to apply the Hadamard transform to the first register, giving us

$$H^{\otimes 8} |\psi_0\rangle = \frac{1}{\sqrt{256}} \sum_{x=0}^{255} |x\rangle |0\rangle^{\otimes 8} = |\psi_1\rangle . \tag{4.79}$$

We next apply the U_f operator to the second register, giving us

$$\begin{aligned} U_f |\psi_1\rangle &= \frac{1}{\sqrt{256}} \sum_{x=0}^{255} |x\rangle |7^x \mod 15\rangle \\ &= \frac{1}{\sqrt{256}} [|0\rangle |1\rangle + |1\rangle |7\rangle + |2\rangle |4\rangle + |3\rangle |13\rangle + |4\rangle |1\rangle + \ldots] . \end{aligned} \tag{4.80}$$

It is clear from this expression that $r = 4$. Despite this, we continue with the algorithm, since in general the order finding problem will not be as simple. The next step is the measure the second register to obtain some y_0; suppose we find $y_0 = 13$. The first register will then be in a superposition of all the x values that give $f(x) = 13$. We can see from equation (4.80) that x_0, the smallest of these x values, is 3. Dropping the second register, just as we did above, the state of the first register after the measurement is given by

$$|\phi\rangle = \frac{1}{\sqrt{64}} [|3\rangle + |7\rangle + |11\rangle + |15\rangle + \ldots] . \tag{4.81}$$

Now, we must take the Fourier transform of $|\phi\rangle$. Using equation (4.73), we have

$$F_{256} |\phi\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{6\pi i j}{r}} \left| j \frac{256}{r} \right\rangle . \tag{4.82}$$

A measurement of the register now will give either 0, 64, 128, or 192, each with the same probability. Now, using equation (4.74), quickly see that, indeed, $r = 4$. Now, since r is even, we know that at least one of

$$\gcd(7^2 + 1, 15) = \gcd(50, 15) = 5 \tag{4.83}$$

or

$$\gcd(7^2 - 1, 15) = \gcd(48, 15) = 3 \tag{4.84}$$

is a nontrivial factor of 15. In fact, each of these is a nontrivial factor; thus, we have successfully used Shor's algorithm to factor 15.

5 Linear Optics Quantum Computation

5.1 Why Linear Optics?

Of the proposed physical realizations of quantum computation, none is simpler than linear optics quantum computation (LOQC) [15]. While not a feasibly scalable implementation, there are many advantages to LOQC. First and foremost, the subject of quantum optics is well understood both theoretically and experimentally. A wide range of experimental techniques allowing the manipulation of single photons, which in LOQC are used as qubits, are known. Indeed, single photons provide several choices for the definition of qubits. Quantum information can be encoded in a photon's two orthogonal polarization states, in different spatial modes of a photon, or in some combination of the two [16]. In this section, we focus more on using photon spatial modes as our qubits, since, in theory, a single photon may then be used to represent an entire quantum register when sent through a multiport beam splitter, as shown below.

There are also, unfortunately, some major disadvantages to LOQC. The primary issue, and the one that prevents LOQC from being a fully realizable solution for quantum computation, is that the number of optical devices necessary to build quantum circuits tends to scale exponentially with the number of qubits. Also, when using free space optics, component alignment can be very challenging. Since a linear optics quantum computer is at its heart a complex interferometer, all the issues in interferometer design and construction carry over to any experimental implementation of LOQC. Finally, some multi-qubit gates are much simpler to implement experimentally using nonlinear optical effects; for the most part, though, we avoid such constructions due to the additional (and somewhat unnecessary) theoretical complexity.

Even though LOQC is not scalable, it can be used to implement small quantum circuits relatively simply, and serves as an excellent test bed for proof of principle experiments. In the remainder of this section, we present implementations of a universal set of gates, as well as compact implementations of the Fourier and Hadamard transforms. Finally, we present some sample quantum circuits implemented with LOQC.

5.2 Optical Realizations of Universal Quantum Logic Gates

Recall from section 3.4.3 that a universal set of quantum logic gates is given by the single qubit operations H and T coupled with a control-NOT gate. We consider here optical implementations of these gates. We first present realizations of these gates using the spatial modes of photons as qubits, followed by implementations using photon polarization.

5.2.1 Spatial Mode Gates

As we mentioned briefly above, LOQC circuits are in essence interferometers. As such, LOQC circuits are engineered so that photons travel in one of a number of discrete optical paths. Classically, a photon can exist in only a single path at a time; quantum mechanically, however, a photon can have some amplitude of being in a number of different paths simultaneously. Thus, we can use the location of a given photon as quantum information, and can use this 'which-path' information to define qubits. We term qubits of this type spatial mode qubits, since the existence of a photon in a particular path is an excitation of one of the optical modes of that path. The basis states of a spatial mode qubit correspond to the existence of a photon in one of two paths.

As an example of a spatial mode gate, consider the $\pi/8$ gate T, the simplest gate to implement in our universal set. The spatial mode LOQC implementation of the T gate is shown in Figure 12. The interpretation of the right-hand side of this

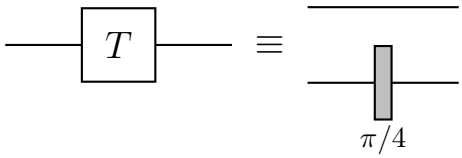

Figure 12: The $\pi/8$ gate operating on a spatial qubit. The shaded box is a phase shifter.

figure is slightly different from our earlier quantum circuit diagrams. Here, the solid lines represent not abstract quantum wires, but instead different paths a photon may take through the circuit. Thus, even though the T gate is a single qubit gate, there are two solid lines in Figure 12; a single spatial mode qubit is formed by two optical paths. To make the T gate, we simply insert a $\pi/2$ phase shifter into one of the two optical paths. Identifying the top path in Figure 12 with the $|0\rangle$ state and the bottom path with the $|1\rangle$, this optical setup will perform the transformations

$$\begin{aligned} |0\rangle &\rightarrow |0\rangle \\ |1\rangle &\rightarrow e^{i\pi/4}, \end{aligned} \tag{5.1}$$

which, in matrix form, is

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{5.2}$$

This, of course, is exactly the transformation performed by the T gate, and so the circuit of Figure 12 does indeed implement the T gate.

The Hadamard gate is also simple to implement in LOQC. Recall that in the standard computational basis, its matrix representation is

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{5.3}$$

A beam splitter surrounded by two $-\pi/2$ phase shifters in one of its modes performs this operation on a spatial qubit, as is shown in Figure 13 In general, a beam splitter will perform the transformation

$$B = \begin{pmatrix} R_{31} & T_{32} \\ T_{41} & R_{42} \end{pmatrix}, \tag{5.4}$$

where R and T are reflection and transmission coefficients for the paths labelled by their subscripts [17]. For a $50-50$ lossless beam splitter, R and T have equal magnitude, and since the relation

$$|R|^2 + |T|^2 = 1 \tag{5.5}$$

must hold for reflection and transmission coefficients, this magnitude is $1/\sqrt{2}$. Note, however, that reflections introduce a $\pi/2$ phase shift; there will thus be a relative phase of $\pi/2$ between the reflection and transmission coefficients. We choose to assign this phase factor to the transmission coefficients. In this case, the final transformation of a $50-50$ lossless beam splitter is

$$B = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}. \tag{5.6}$$

Now, if we insert a $-\pi/2$ phase shifter in the b-mode both before and after the beam splitter, as shown in Figure 13, the full transformation becomes

$$Q = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H, \tag{5.7}$$

and so implements the Hadamard gate.

The last gate in our universal set, the control-NOT gate, is more challenging to implement, and cannot be implemented fully deterministically using only linear optics [15]. The method we will use employs a quantum nondemolition (QND) measurement of photon number that will function as a conditional phase shift [18]. While a QND measurement employs a nonlinear optical device, namely a medium displaying the optical Kerr effect [19], optical QND measurements are well understood and do not add significantly to the circuit complexity for our purposes. For a more careful treatment of optical QND measurements, please see Appendix C.

The control-NOT gate is shown in Figure 14. Recall that, in the standard basis,

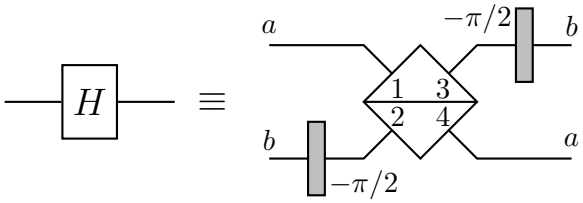

Figure 13: The Hadamard gate on a spatial qubit. The diamond is a beam splitter, and it is surrounded by $-\pi/2$ phase shifters in the b spatial mode.

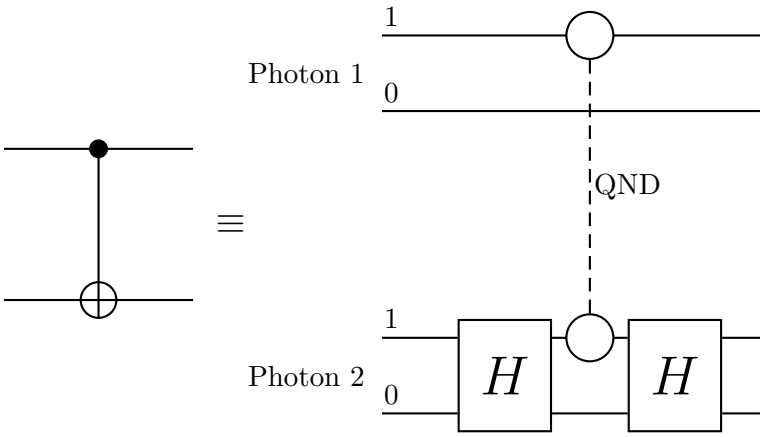

Figure 14: Control-NOT gate implemented with a QND as a conditional phase shift.

a control-NOT gate has the following matrix representation:

$$CN = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{5.8}$$

In the setup shown in Figure 14, the dual-photon transformations before and after the QND have the form

$$I \otimes H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}. \tag{5.9}$$

In the photon number basis[6], also known as the Fock basis, the QND measurement operator is given by [20]

$$Q = e^{i\delta n_{1\alpha} n_{2\beta}}, \tag{5.10}$$

where δ is the magnitude of the conditional phase shift, $n_{1\alpha}$ is the photon number operator for photon 1 in mode α, and $n_{2\beta}$ is the photon number operator for photon 2 in mode β. We specify that $\delta = \pi$, which, although difficult to obtain experimentally, will allow us to construct a control-NOT gate. Thus, in the standard basis, the QND measurement is given by

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{5.11}$$

Thus, the full transformation performed by the circuit in Figure 14 is given by

$$\begin{aligned} (I \otimes H)\, Q\, (I \otimes H) &= \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \end{aligned} \tag{5.12}$$

which is exactly the control-NOT gate.

[6] that is, the basis where the labels in the kets refer to the number of photons present in the system.

5.2.2 Polarization State Gates

Before showing implementations of these universal gates using photon polarizations as our qubits, we first discuss the optical device known as a waveplate. A waveplate is a birefringent crystal that introduces a phase shift between the ordinary photons polarized along the slow axis of the crystal and the extraordinary photons polarized along the fast axis [21]. Using $|o\rangle$ and $|e\rangle$ as the ordinary and extraordinary axes, respectively, this transformation is given by

$$U_{WP} = |o\rangle\langle o| + e^{i\phi}|e\rangle\langle e| . \tag{5.13}$$

Let us label the horizontal and vertical photon polarization bases as $|H\rangle$ and $|V\rangle$, respectively. For a waveplate oriented at some angle θ to the horizontal and vertical polarization axes, as shown in Figure 15, we can transform between these two bases

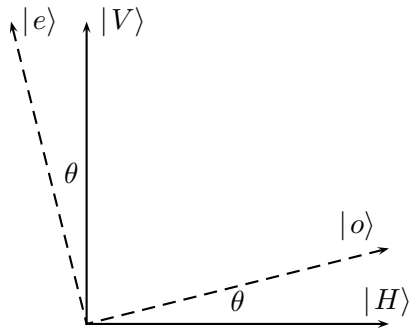

Figure 15: Geometry for a waveplate oriented at an angle θ.

using the following unitary rotation:

$$R(\theta) = \begin{pmatrix} \langle o|H\rangle & \langle o|V\rangle \\ \langle e|H\rangle & \langle e|V\rangle \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}. \tag{5.14}$$

Now suppose a photon in an arbitrary polarization state $|\psi_{in}\rangle = \psi_H|H\rangle + \psi_V|V\rangle$ enters the waveplate. Rotating into the $\{|o\rangle, |e\rangle\}$ basis, we have

$$|\psi_{in}\rangle = (\psi_H\cos\theta + \psi_V\sin\theta)|o\rangle + (-\psi_H\sin\theta + \psi_V\cos\theta)|e\rangle . \tag{5.15}$$

Now we can apply the transformation effected by the waveplate:

$$\begin{aligned} |\psi_{out}\rangle &= U_{WP}|\psi_{in}\rangle \\ &= (\psi_H\cos\theta + \psi_V\sin\theta)|o\rangle + e^{i\phi}(-\psi_H\sin\theta + \psi_V\cos\theta)|e\rangle . \end{aligned} \tag{5.16}$$

Finally, we must rotate back into the $\{|H\rangle, |V\rangle\}$ basis. The final state is

$$|\psi_{out}\rangle = \left[\psi_H \left(\cos^2\theta + e^{i\phi}\sin^2\theta\right) + \frac{1}{2}\psi_V \sin 2\theta \left(1 - e^{i\phi}\right)\right]|H\rangle + \left[\frac{1}{2}\psi_H \sin 2\theta \left(1 - e^{i\phi}\right) + \psi_V \left(\sin^2\theta + e^{i\phi}\cos^2\theta\right)\right]|V\rangle . \quad (5.17)$$

It is clear, then, that a waveplate performs the transformation

$$WP(\theta) = \begin{pmatrix} \cos^2\theta + e^{i\phi}\sin^2\theta & \frac{1}{2}\sin 2\theta \left(1 - e^{i\phi}\right) \\ \frac{1}{2}\sin 2\theta \left(1 - e^{i\phi}\right) & \sin^2\theta + e^{i\phi}\cos^2\theta \end{pmatrix}. \quad (5.18)$$

Now, using waveplates we can implement quantum gates. The $\pi/8$ gate can be constructed using an eighth waveplate. An eighth waveplate introduces a phase shift of an eighth of a wavelength, and so $\phi = \pi/4$. For an eighth waveplate oriented at $0°$, equation (5.18) reduces to

$$EWP(0) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \quad (5.19)$$

which is the $\pi/8$ gate.

Finally, half waveplate creates a phase difference of half a wavelength between the ordinary and extraordinary photons, and so has $\phi = \pi$. In general, then, for a half waveplate, equation (5.18) reduces to

$$HWP(\theta) = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix}. \quad (5.20)$$

A half waveplate oriented at an angle of $22.5°$, then, will perform the transformation

$$HWP(22.5°) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad (5.21)$$

which is a Hadamard gate.

The control-NOT gate (along with every multi-qubit gate) is difficult to implement using only polarization states. Most LOQC schemes, however, use polarization for only a single qubit, if at all. It is very simple to construct a control-NOT gate using one spatial mode qubit and one polarization qubit [22]. Either qubit may be used as the control qubit.

As shown in Figure 16(a), placing a half waveplate oriented at $90°$ in the spatial mode of a photon representing $|1\rangle$ implements a control-NOT gate with the polarization qubit as the target qubit. The polarization qubit can be used as the control qubit by utilizing a polarizing beam splitter, as shown in Figure 16(b).

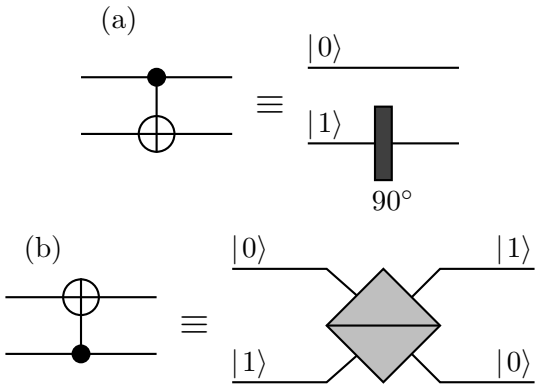

Figure 16: Hybrid spatial mode/polarization control-NOT gates. In each circuit diagram, the top line represents the spatial mode qubit while the bottom line represents the polarization qubit. (a) Using a half waveplate as a polarization rotator in the $|1\rangle$ path creates a control-NOT gate with the polarization qubit as the target qubit. (b) Here, the target and control qubits are reversed. The shaded diamond is a polarizing beam splitter.

5.3 The Fourier and Hadamard Transforms

The two most important transformations used in the known quantum algorithms are the Fourier and Hadamard transforms. While each of these can be built from the universal gate set described above (where we are differentiating the one-qubit Hadamard gate from the n-qubit Hadamard transform), LOQC provides more compact implementations of each. We consider the quantum Fourier transform first.

Consider a symmetric multiport beam splitter [6], which is a straightforward generalization of the two-port beam splitters we discussed earlier. A symmetric multiport beam splitter is simply a device with as many input ports as output ports that can take a beam of light from any of its input ports and divide it equally among all its output ports. Here we show that a symmetric multiport beam splitter can implement the quantum Fourier transform on the spatial modes of a photon qubit. For theoretical simplicity, we once again assume lossless optical components. An $N \times N$ symmetric multiport beam splitter, where $N = 2^n$ and n is the number of qubits, performs a unitary transformation with the defining characteristic that every matrix element has a common modulus of $1/\sqrt{N}$ [23]. We will now show that any such unitary transformation can perform a quantum Fourier transform. We begin by showing that any such matrix may be written in a so-called real bordered form by factoring out two diagonal matrices, which may be identified with phases of the input and output beams [24].

Lemma 5.1 *Let U be an $N \times N$ matrix such that every element has a common modulus; we will assume a unimodular matrix for simplicity. Then for diagonal matrices*

$$D_1 = \begin{pmatrix} \alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_N \end{pmatrix} \tag{5.22}$$

and

$$D_2 = \begin{pmatrix} \beta_1 & & & \\ & \beta_2 & & \\ & & \ddots & \\ & & & \beta_N \end{pmatrix} \tag{5.23}$$

where

$$\alpha_k = \left(\frac{U_{k1}^2}{U_{11}}\right)^{1/2} \tag{5.24}$$

and

$$\beta_k = \left(\frac{U_{1k}^2}{U_{11}}\right)^{1/2}, \tag{5.25}$$

$U = D_1 U' D_2$, where the first row and first column of U' are entirely ones. Such a matrix is called real bordered.

Proof Since both D_1 and D_2 are diagonal, their matrix elements must be their eigenvalues. All the α_k and β_k are of the form $e^{i\theta}$ because of the assumption of unimodularity. Each must therefore be unitary. Using this fact, we define

$$U' = D_1^\dagger U D_2^\dagger. \tag{5.26}$$

Furthermore, we have that

$$\alpha_k^* = \left(\frac{U_{11}}{U_{k1}^2}\right)^{1/2} \tag{5.27}$$

and

$$\beta_k^* = \left(\frac{U_{11}}{U_{1k}^2}\right)^{1/2}. \tag{5.28}$$

Now, let us find U'. Multiplying the matrices together, we have

$$\begin{aligned} U' &= \begin{pmatrix} \alpha_1^* & & \\ & \ddots & \\ & & \alpha_N^* \end{pmatrix} \begin{pmatrix} U_{11} & \cdots & U_{1N} \\ \vdots & \ddots & \vdots \\ U_{N1} & \cdots & U_{NN} \end{pmatrix} \begin{pmatrix} \beta_1^* & & \\ & \ddots & \\ & & \beta_N^* \end{pmatrix} \\ &= \begin{pmatrix} \alpha_1^* & & \\ & \ddots & \\ & & \alpha_N^* \end{pmatrix} \begin{pmatrix} U_{11}\beta_1^* & U_{12}\beta_2^* & \cdots & U_{1N}\beta_N^* \\ U_{21}\beta_1^* & U_{22}\beta_2^* & \cdots & U_{2N}\beta_N^* \\ \vdots & \vdots & \vdots & \vdots \\ U_{N1}\beta_1^* & U_{N2}\beta_2^* & \cdots & U_{NN}\beta_N^* \end{pmatrix} \\ &= \begin{pmatrix} \alpha_1^* U_{11}\beta_1^* & \alpha_1^* U_{12}\beta_2^* & \cdots & \alpha_1^* U_{1N}\beta_N^* \\ \alpha_2^* U_{21}\beta_1^* & \alpha_2^* U_{22}\beta_2^* & \cdots & \alpha_2^* U_{2N}\beta_N^* \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_N^* U_{N1}\beta_1^* & \alpha_n^* U_{N2}\beta_2^* & \cdots & \alpha_N^* U_{NN}\beta_N^* \end{pmatrix}. \end{aligned} \tag{5.29}$$

Thus, an arbitrary element of U' is

$$U'_{st} = \alpha_s^* U_{st} \beta_t^*. \tag{5.30}$$

Using our expressions for α_k^* and β_k^*, we have

$$U'_{st} = U_{st}\left(\frac{U_{11}^2}{U_{s1}^2 U_{1t}^2}\right)^{1/2} = \frac{U_{11}U_{st}}{U_{s1}U_{1t}}. \tag{5.31}$$

It is clear, then, that U' is real bordered; replacing either s or t with 1 makes the entire fraction 1.

□

Now let us apply the unitarity condition to the U' defined in the above lemma. Unitarity says that

$$U'^\dagger U' = I. \tag{5.32}$$

We must therefore have

$$\sum_{k}^{N} \left(U'^{\dagger}\right)_{sk} U'_{kt} = \sum_{k}^{N} U'^{*}_{ks} U'_{kt} = \delta_{st}. \tag{5.33}$$

The matrix elements of the quantum Fourier transform, indexing from zero, are given by

$$F_{st} = \frac{1}{\sqrt{N}} e^{2\pi i s t/N}. \tag{5.34}$$

Inserting this solution into the unitarity condition, we have

$$\sum_{k} F^{\dagger}_{sk} F_{kt} = \frac{1}{N} \sum_{k}^{N} e^{-2\pi i k s/N} e^{2\pi i k t/N} \tag{5.35}$$

which is the orthonormality condition for Fourier series, and thus is equal to δ_{st}. Thus, a solution consisting of the quantum Fourier transform matrix elements is consistent with unitarity.

We have thus shown that a symmetric multiport beam splitter, and indeed any unitary transform with elements of common modulus, can perform the quantum Fourier transform. The quantum Fourier transform can also be used to simulate the action of the Hadamard transform on the $|0\rangle^{\otimes n}$ qubit, an essential first step in many quantum algorithms. A proof of this is shown below.

Lemma 5.2 *The quantum Fourier transform and the Hadamard transform produce the same output state when applied to the input state* $|0\rangle^{\otimes n}$*; that is,*

$$F_n |0\rangle^{\otimes n} = H^{\otimes n} |0\rangle^{\otimes n}. \tag{5.36}$$

Proof In the standard basis, the matrix representation of $|0\rangle^{\otimes n}$ is

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{5.37}$$

and so therefore, $|0\rangle^{\otimes n}$ will only be affected by the first column of the matrix representation of any operator acting on it. Let us consider the matrix representation of the Fourier transform in this basis, as given in equation (5.34):

$$F_{jk} = \frac{1}{\sqrt{2^n}} e^{2\pi i j k/2^n}. \tag{5.38}$$

In the first column of this transformation, $k = 0$, and thus $F_{j0} = 1$ for all j. Thus the transformed qubit state will be

$$\frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{5.39}$$

which is an even superposition of the 2^n basis states of the Hilbert space of n qubits. But this is exactly the transformation effected by the Hadamard transform on n qubits, $H^{\otimes n}$, and so

$$F_n \left|0\right\rangle^{\otimes n} = H^{\otimes n} \left|0\right\rangle^{\otimes n}, \tag{5.40}$$

as desired.

□

5.4 Some Simplifications

In the above sections, we have shown that LOQC is, at least theoretically, a feasible physical realization of quantum computation. Unfortunately, LOQC's poor scalability will limit it to the domain of small quantum circuits. In this domain, however, there are some simplifications that can be made to make LOQC more robust.

First, we address the issue of the alignment of optical components. Because of the precision necessary to implement LOQC, the alignment of the necessary free space optical components can be very challenging. This problem may be solved by using fiber optics technology. Using optical fibers removes the difficulty of alignment, and makes the entire quantum circuit more compact. The greatest simplification comes in replacing any multiport beam splitters, perhaps the most difficult element of LOQC to implement, with symmetric $N \times N$ fiber couplers, which perform the same transformation on the spatial modes of a photon. Additionally, $N \times N$ couplers provide a simple way of performing the inverse quantum Fourier transform, just as they do for the transform itself. The inverse quantum Fourier transform matrix elements are given by

$$(F_{st})^{-1} = \frac{1}{\sqrt{N}} e^{-2\pi its/N}. \tag{5.41}$$

These can be obtained simply by changing the labelling of the output ports of the fiber coupler, as shown in Figure 17 [25].

Mathematically, this relabelling is represented by multiplication by the operator

$$U = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{pmatrix}. \tag{5.42}$$

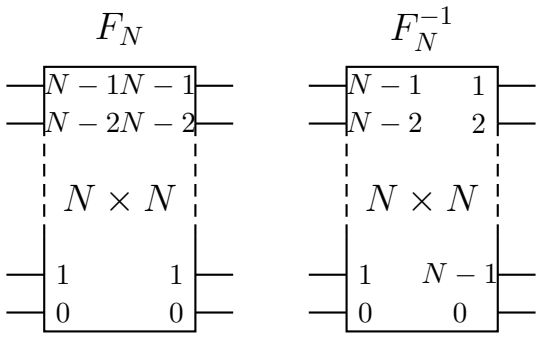

Figure 17: $N \times N$ fiber couplers for performing the quantum Fourier transform F_N and its inverse F_N^{-1}.

This operator leaves the 0^{th} row untouched while swapping the 1^{st} row with the $N-1^{st}$ row, the 2^{nd} row with the $N-2^{nd}$ row, and so on. In general, then, we are swapping the $(F_N)_{st}$ and $(F_N)_{(N-s)t}$ elements of the transform, where $0 < s < N$. We then have

$$\begin{aligned}(F_N)_{(N-s)t} &= \frac{1}{\sqrt{N}} e^{2\pi i(N-s)t/N} = \frac{1}{\sqrt{N}} \left(e^{2\pi i}\right)^t \left(e^{-2\pi ist/N}\right) \\ &= \frac{1}{\sqrt{N}} e^{-2\pi ist/N} = (F_N)^{-1}_{st}, \end{aligned} \tag{5.43}$$

as desired.

5.5 Sample Optical Quantum Circuits

Here we present two circuits that implement Grover's algorithm [10]. The first circuit searches a two-qubit database using two spatial modes and the polarization of single photon as its qubits, and is due to Kwiat *et al.* [16]. The second circuit is an extension to n qubits using linear integrated optics, and is a simplification of the circuit presented in Howell and Yeazell [18].

5.5.1 Two-Qubit Circuit

Recall that Grover's algorithm proceeds as follows: first, we initialize our register of qubits into an even superposition of all the basis states of that register. We then invoke an oracle that gives a π phase shift to the desired target state. We then perform a Hadamard transform on the register, followed by a π phase shift to every element except the $|0\rangle$ element. Finally, we perform another Hadamard transform

on the register. This process must in general be repeated, but for the two-qubit register presented in the following circuit, one execution of the main algorithm is sufficient to transfer all the probability amplitude to the desired target state. As mentioned above, the two qubits will be the polarization state of a single photon and two spatial modes. In Figure 18, we use the gates described above to implement the algorithm directly. We have simply used the gate constructions developed earlier in this section and laid them out according the Grover's specification.

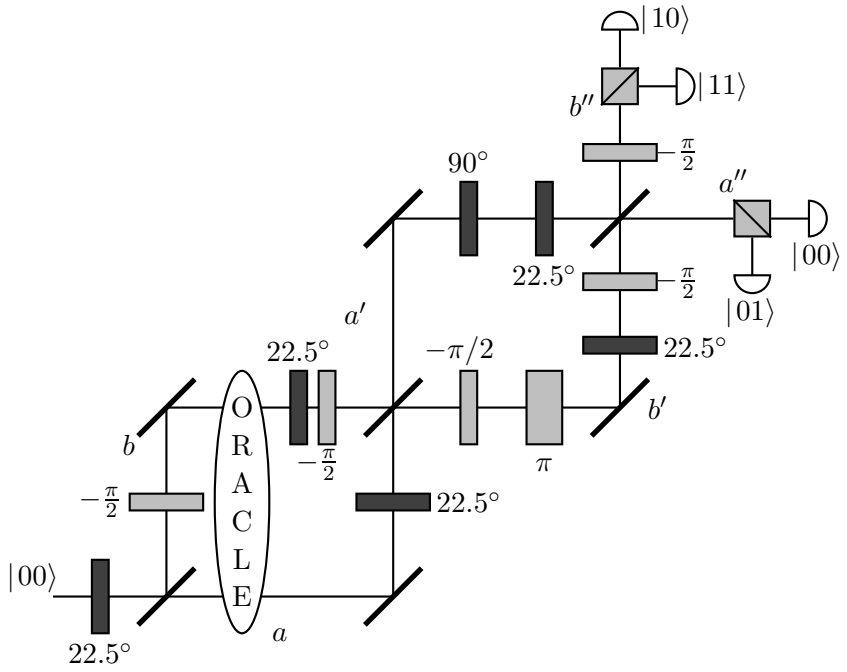

Figure 18: Optical circuitry to perform Grover's algorithm on a two-qubit database. The lightly shaded elements are phase shifters while the darkly shaded elements are half waveplates. At the end of each path lies a polarizing beam splitter followed by two photon detectors, which perform the final measurement on the circuit.

By inspection, though, we can eliminate redundant circuit elements and combine others, thereby creating a simpler circuit requiring far fewer optical components. First, we note that operations on the spatial mode qubit commute with operations on the polarization qubit, since the two qubits are noninteracting. Thus, we can move the half waveplate from path b into path b' past the beam splitter and phase shifter, since the waveplate acts on the polarization qubit and the beam splitter and phase shifter act on the spatial mode qubit. In path b', then, we have two half

waveplates oriented at the same angle directly next to each other. Since the half waveplate transformation as expressed in equation (5.20) is Hermitian as well as unitary, these two half waveplates together perform the identity transformation on the polarization qubit, and thus are not needed.

Likewise, we can move the half waveplate in path a to path a', where we now have three half waveplates in a row. These three waveplates together perform the transformation

$$\begin{aligned} HWP(22.5°)HWP(90°)HWP(22.5°) &= \tfrac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \cos\pi & \sin\pi \\ \sin\pi & -\cos\pi \end{pmatrix} = HWP(45°) \end{aligned} \tag{5.44}$$

and so these three half waveplates may be replaced by a single half waveplate oriented at 45°.

Now, looking at the components acting on the spatial mode qubit, in path b', we have three phase shifters that combine to a total phase shift of 0, and so may be eliminated. Similarly, the two $-\pi/2$ phase shifters in path b may be combined into a single $-\pi$ phase shifter. Finally, the last phase shifter, in path b'', changes only a global phase, since by that point the system will be in a pure state. Since the final step of the algorithm is measurement, this global phase does not matter, and so this final phase shifter can be dropped.

With these simplifications, the final circuit to implement Grover's algorithm in the two-qubit case is shown in Figure 19.

5.5.2 n-Qubit Extension

To construct an n-qubit circuit to implement Grover's algorithm, it is convenient to use the alternate version of the algorithm discussed in section 4.4.3. Recall that in this version of the algorithm, we use repeated applications of the operator $Q = -I_i U^\dagger I_t U$ to search the database, where U is any unitary transformation and $I_\psi = I - 2\left|\psi\right\rangle\left\langle\psi\right|$, that is, an identity matrix with the $\psi\psi$ element equal to -1.

In our circuit, we choose U to be the quantum Fourier transform, since both it and its inverse are easily constructed using fiber couplers. The I_ψ operator can easily be constructed by placing a π phase shifter in the appropriate path of the apparatus. A realization of the operator Q is shown in Figure 20. In this figure, we assume that $\left|0\right\rangle$ is the initial state of the computer and $\left|2\right\rangle$ is the target state. As with the standard version of Grover's algorithm, the so-called Grover iteration performed by the operator Q must be iterated $\mathcal{O}(\sqrt{N})$ times to reliably produce the desired target state upon measurement, and so a full circuit to implement the algorithm would contain $\mathcal{O}(\sqrt{N})$ copies of the circuit in Figure 20, followed by a measurement using photon detectors.

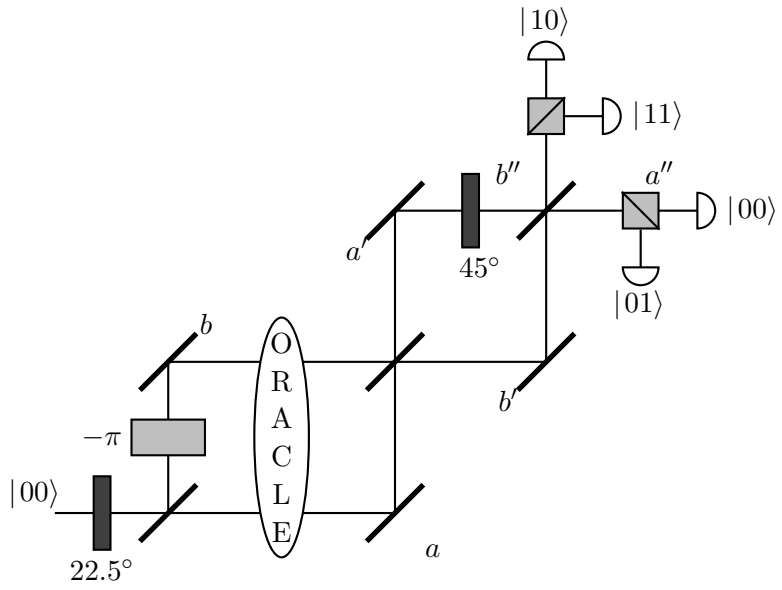

Figure 19: The "compiled" version of the Grover's algorithm circuit of Figure 18, with all redundant components removed.

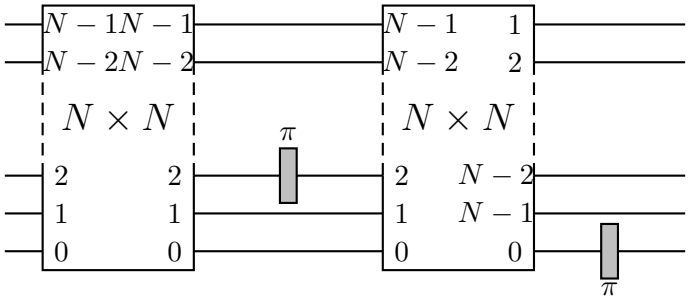

Figure 20: An optical realization of the operator Q described by Grover that may be used to search a quantum database. This example circuit treats the state $|2\rangle$ as the target state.

6 Conclusion

In this paper, we have discussed many of the central topics in quantum computation. Every subject we have mentioned is an area of active research. Though quantum computation appears to be an incredibly promising development in physics, there are very real issues with the subject. We have not discussed these problems in depth in this paper due to lack of space, but we take this time to introduce the major issues associated with quantum computation.

First and foremost, the reader must remember that there is nothing a quantum computer can do that a classical computer cannot, given enough time, because each device can compute the same class of functions. And while quantum computers would seem to be much more efficient than classical computers at some tasks, they have not been proved to be so. All evidence suggests that quantum computers are more efficient, but more work must be done to justify this belief. It is eminently possible that quantum computers are more efficient than classical computers only in some restricted class of problems. If this were the case, there would be no need to construct a true general purpose quantum computer; special purpose computers would suffice.

Secondly, the actual construction of quantum computers is very difficult, and is probably the largest area of research in quantum computation today. The most powerful quantum information processors to date can operate on only up to ten qubits. A quantum computer of this size is effectively useless: it can perform no real computations. The two main concerns in the construction of quantum computers have been decoherence and the difficulty of efficiently creating large scale entanglement.

Decoherence, also known as quantum noise, is the process by which superpositions decay. A quantum computer requires complex superpositions in order to operate; preventing the corruption of these states is vital to successful computation. Prevention and correction of qubit corruption is thus an important area of study in quantum computation, and several lines of research show great promise. Chief among these is the theory of quantum error correcting codes (QECCs), which builds on the similar classical theory. The main idea of a QECC is to encode the state of each logical qubit in several physical qubits in such a way that the corruption of the logical qubit is detectable and correctable. QECCs have been experimentally tested and work well; however, the best known code encodes every logical qubit in five physical qubits, further increasing the requirements for a useful quantum information processor.

Decoherence occurs because of a quantum system's interaction with its environment. Clearly, then, systems with minimal coupling to the rest of the world would make good quantum computers. A major problem arises with such systems, however: the smaller the coupling to the outside world, the more difficult it is to engineer the entangled states necessary for quantum computation. Conversely, the easier it is to manipulate a quantum state, the faster it will decohere. A delicate

balance must therefore be struck to construct a quantum computer.

Because of these difficulties, it is unlikely that there will be a quantum computer on every desk in the near future. To make an analogy with classical computers, quantum computation is not yet even to the vacuum tube stage. The immense promise of quantum computers, however, coupled with the dynamic state of the field today means that research in quantum computation will continue, and that the difficulties noted above will be solved. Indeed, researchers at IBM have just recently demonstrated Shor's algorithm experimentally [26], factoring 15 into 5 and 3. Computer development has always proceeded more quickly than estimated, and perhaps the development of quantum computers will be no different.

Appendices

A Classical Logic Gates

A classical computer, at its base level, is a complex combination of digital circuits. Digital circuits, in turn, are combinations of a discrete set of classical logic gates. In this appendix, we briefly discuss the standard set of classical logic gates.

A classical logic gate is a device that transforms some number of input binary values to some number of output values. The simplest logic gate of all is the NOT gate. A basic NOT gate takes one input value and inverts it. We define the symbol $\overline{X}$ to be the inverted value of an input X. The action of the NOT gate is then specified in the following truth table:

X	$\overline{X}$
0	1
1	0

(A.1)

In a truth table such as this, we place the input values on the left and the output values on the right. Each row specifies how one particular set of input values transforms.

Aside from the NOT gate, there are two other basic logic operations. These are the AND gate and the OR gate. In their basic forms, each of these gates takes two input values and returns one output value. We represent the AND operation by $\cdot$ and the OR operation by $+$. The action of these two gates is specified in the following two truth tables:

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

(A.2)

As the NOT, AND, and OR gates are the fundamental digital logic gates, any digital circuit may be built from them.

Consider now another gate, the NAND gate. A NAND gate is simply an AND gate with its output inverted, and so is represented by the truth table

X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

(A.3)

Interestingly, any digital circuit can be built out of NAND gates. This is simple to show by construction. We need simply to show that NAND gates alone can implement the NOT gate, AND gate, and OR gate.

The simplest of these is the NOT gate. Suppose we send one input bit into both input ports of a NAND gate. The two input bits to the NAND gate must therefore be the same, and so we are limited to the first and last line of truth table in equation (A.3). But these two lines (compressing the two input bits into a single bit) compose a truth table identical to that of a NOT gate, shown in equation (A.1). In this fashion, we can build a NOT gate from a NAND gate.

The AND gate is also simple to construct. By definition, we must simply invert the output of a NAND gate to obtain a AND gate. Connecting the output of a NAND gate to a NOT gate (or, rather, its implementation in NAND gates) suffices.

The OR gate is only slightly more complicated to construct from NAND gates. Consider a NAND gates with its two inputs inverted. It is simple to check that the truth table for this configuration is simply

0	0	0
0	1	1
1	0	1
1	1	1

(A.4)

But this is simply the truth table for an OR gate, as defined in equation (A.2), and so this configuration corresponds exactly to the OR gate.

Before leaving this appendix, we mention one more classical logic gate that is often used: the exclusive-OR, or XOR gate. The XOR gate behaves like an OR gate except for the fact that it returns 0 if both its inputs are 1. We denote the XOR operation by $\oplus$, and its truth table is

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

(A.5)

B Classical Algorithms Used in Shor's Algorithm

As we showed in Section 4.5, Shor's algorithm relies on both calculating greatest common divisors and continued fractions expansions. For Shor's algorithm as a whole to be efficient, we must be able to calculate both of these in polynomial time. Luckily, there exist efficient classical algorithms to find both of these quantities. We present those algorithms here.

B.1 The Euclidean Algorithm for Greatest Common Divisors

First, we define the greatest common divisor of two integers a and b as the largest integer d greater than zero that divides both a and b.

Euclid's algorithm for determining these greatest common divisors in turn depends on the basic result known as the division algorithm. The division algorithm is contained in the proof of the following theorem:

Theorem B.1 *Let a and b be integers such that $a \geq 0$ and $b > 0$. Then there exist integers q and r such that $a = qb + r$ with $0 \leq r < b$.*

Proof We prove this theorem using split induction. Let us consider b to be fixed, and let $P(a)$ be the statement that the theorem is true for a.

We now have b base cases, one for each $a < b$, so that a is in the range $[0, b-1]$. For each of these cases, $a = (0)b + a$, and so the theorem is true in each case, with $r = a$ in the language of the theorem.

Now, for our inductive step, let us assume that $P(a)$ is true. We want to show that $P(a+b)$ is also true. Since $P(a)$ is true, we know that $a = qb+r$ for $0 \leq r < b$. Thus, adding b to both sides of this equation, we have

$$a + b = qb + r + b = (q + 1)b + r. \tag{B.1}$$

Thus, the theorem still holds for the case $a + b$ with quotient $q + 1$ and remainder r. Thus, the truth of $P(a)$ implies the truth of $P(a + b)$, and since we have proved b base cases the theorem is proved by induction.

□

This proof suggests the general division algorithm, namely that, given some a and b we continually subtract b from a, increasing the quotient by one every time, until a is less than b, at which point we know both the quotient and the remainder. We express this algorithm more precisely below:

Algorithm: The Division Algorithm
$q = 0$
repeat until $a < b$
$a = a - b$

$q = q + 1$
end repeat
$r = a$
output q, r

Now, we can move on to the Euclidean algorithm. This algorithm makes use of two facts. The first of these is that if b divides a, then $\gcd(a, b) = b$, which follows directly from the definition of the greatest common divisor. The second fact is the following lemma:

Lemma B.1 *Suppose* $a = qb + r$. *Then* $\gcd(a, b) = \gcd(b, r)$.

Proof Suppose that some integer d divides both a and b. Then d must also divide all multiples of b, and so d divides qb. Thus, since d divides a and qb, it will also divide their difference, which is exactly r, so that if d divides a and b, it also divides r.

Now, suppose that d instead divides b and r. Then, as above, d divides qb. Since d divides qb and r, it will also divide their sum, a. Thus, if d divides b and r, it will also divide a.

Thus, we have shown that the pairs (a, b) and (b, r) have exactly the same common divisors, and so then the greatest common divisor of the two pairs will also be the same.

□

Using these two facts, we can write down the Euclidean algorithm, which again has as its input integers a and b such that $a \geq 0$ and $b > 0$:

Algorithm: The Euclidean Algorithm
$r = 1$
repeat until $r = 0$
$r =$ (the remainder when a is divided by b using the division algorithm)
$a = b$
$b = r$
end repeat
output a

B.2 Continued Fractions

Suppose x is a rational number. One simple way to represent x is by a continued fractions expansion, which will allow us to express x as a sequence of integers. A

continued fraction is an object of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ldots + \cfrac{1}{a_N}}}}. \tag{B.2}$$

We define an abbreviation for a continued fraction as $[a_0, a_1, \ldots, a_N]$, and further define $[a_0, a_1, \ldots, a_n]$ to be the n^{th} convergent to this expression. The algorithm to obtain a continued fraction from a rational number is simple; we will illustrate the procedure with an example. Consider $x = 77/65$. The first step is to split off the integer part of this number and invert the fractional part:

$$x = 1 + \cfrac{1}{\cfrac{65}{12}}. \tag{B.3}$$

We now continue this 'split-and-invert' process:

$$\begin{aligned} 1 + \cfrac{1}{\cfrac{65}{12}} &= 1 + \cfrac{1}{5 + \cfrac{1}{\cfrac{12}{5}}} \\ &= 1 + \cfrac{1}{5 + \cfrac{1}{2 + \cfrac{1}{\cfrac{5}{2}}}} \\ &= 1 + \cfrac{1}{5 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2}}}}. \end{aligned} \tag{B.4}$$

At this point, the algorithm terminates, since we can no longer split off a fractional part from 2. Thus, we have found the continued fractions representation of 77/65, which we can write as $[1, 5, 2, 2, 2]$.

The real power of the continued fractions method, however, comes from being able to solve the inverse problem of finding a rational number given a continued fractions expansion. Suppose we have a continued fraction given by $[a_0, a_1, \ldots, a_n]$. Let us make the following inductive definitions:

$$\begin{aligned} p_0 &\equiv a_0 \\ q_0 &\equiv 1 \\ p_1 &\equiv 1 + a_0 a_1 \\ q_1 &\equiv a_1 \end{aligned} \tag{B.5}$$

and for $n \geq 2$

$$\begin{aligned} p_n &= a_n p_{n-1} + p_{n-2} \\ q_n &= a_n q_{n-1} + q_{n-2}. \end{aligned} \tag{B.6}$$

With these definitions, we can prove the following useful theorem:

Theorem B.2 *Using the definitions of p_n and q_n above,*

$$[a_0, a_1, \ldots, a_n] = \frac{p_n}{q_n} \tag{B.7}$$

for all $n \geq 0$.

Proof We prove this theorem by induction on n. The basis steps are simple to prove. For $n = 0$,

$$[a_0] = a_0 = \frac{p_0}{q_0}. \tag{B.8}$$

For $n = 1$,

$$[a_0, a_1] = a_0 + \frac{1}{a_1} = \frac{1 + a_0 a_1}{a_1} = \frac{p_1}{q_1}. \tag{B.9}$$

Finally, for $n = 2$,

$$\begin{aligned} [a_0, a_1, a_2] &= a_0 + \frac{1}{a_1 + \dfrac{1}{a_2}} = a_0 + \frac{a_2}{1 + a_1 a_2} = \frac{a_2 + a_0(1 + a_1 a_2)}{1 + a_1 a_2} \\ &= \frac{a_2(1 + a_0 a_1) + a_0}{a_1 a_2 + 1} = \frac{p_2}{q_2}. \end{aligned} \tag{B.10}$$

Now, for the inductive step, suppose that $[a_0, \ldots, a_k] = p_k/q_k$ for all k such that $3 \leq k < n$. We know that

$$\begin{aligned} [a_0, \ldots, a_n] &= \left[a_0, \ldots, a_{n-2}, a_{n-1} + \frac{1}{a_n}\right] \\ &= \frac{\left(a_{n-1} + \frac{1}{a_n}\right) p_{n-2} + p_{n-3}}{\left(a_{n-1} + \frac{1}{a_n}\right) q_{n-2} + q_{n-3}} \end{aligned} \tag{B.11}$$

from the inductive hypothesis. But

$$a_{n-1} p_{n-2} + p_{n-3} = p_{n-1} \tag{B.12}$$

and

$$a_{n-1} q_{n-2} + q_{n-3} = q_{n-1} \tag{B.13}$$

by definition, so equation (B.11) becomes

$$\frac{\frac{1}{a_n} p_{n-2} + p_{n-1}}{\frac{1}{a_n} q_{n-2} + q_{n-1}} = \frac{p_n}{q_n}, \tag{B.14}$$

and so the theorem is proved by induction.

□

To see this process work, we consider again our earlier example of $x = 77/65$. Recall that we found the continued fractions expansion of x to be $[1, 5, 2, 2, 2]$. Using the definitions of p and q above, we have

$$\begin{aligned} p_0 &= 1, \quad q_0 = 1 \\ p_1 &= 6, \quad q_1 = 5 \\ p_2 &= 13, \quad q_2 = 11 \\ p_3 &= 32, \quad q_3 = 27 \\ p_4 &= 77, \quad q_4 = 65, \end{aligned} \tag{B.15}$$

and so indeed $p_4/q_4 = 77/65$.

Finally, let us show that $\gcd(p_n, q_n) = 1$, so that the fractions produced by the above procedure are always in lowest terms. First, consider the following lemma:

Lemma B.2 *For all $n \geq 1$, $q_n p_{n-1} - p_n q_{n-1} = (-1)^n$.*

Proof We will prove this lemma by induction on n. For the base case, consider $n = 1$. Then, using the definitions in equation (B.5), we have

$$q_1 p_0 - p_1 q_0 = a_1 a_0 - (1 + a_0 a_1) = -1, \tag{B.16}$$

so the base case is proved.

Now, for the inductive step, suppose that $q_n p_{n-1} - p_n q_{n-1} = (-1)^n$, and consider $q_{n+1} p_n - p_{n-1} q_n$. Using the inductive definitions of p_n and q_n in equation (B.6), we have

$$\begin{aligned} q_{n+1} p_n - p_{n+1} q_n &= (a_{n+1} q_n + q_{n-1}) p_n - (a_{n+1} p_n + p_{n-1}) q_n \\ &= a_{n+1} q_n p_n + q_{n-1} p_n - a_{n+1} p_n q_n - q_n p_{n-1} \\ &= -(q_n p_{n-1} - p_n q_{n-1}) = -(-1)^n = (-1)^{n+1}, \end{aligned} \tag{B.17}$$

where we have used the inductive hypothesis. Thus, the lemma is proved by induction.

□

Using this result, we can prove that the fractions produced by the continued fractions method are always in lowest terms:

Theorem B.3 *p_n/q_n is in lowest terms for all $n \geq 0$.*

Proof For the $n = 0$ case, p_0/q_0 is by definition an integer, and so is in lowest terms. For all $n \geq 1$, suppose that there was some d that divided both p_n and q_n, and thus would divide the expression $q_n p_{n-1} - p_n q_{n-1}$. But, as we have shown above, this expression reduces to $(-1)^n$, so this d must also divide $(-1)^n$. The only real integers that divide $(-1)^n$ are 1 and -1, and so therefore $\gcd(p_n, q_n) = 1$. Thus, p_n/q_n is in lowest terms for all $n \geq 0$.

□

C Optical Quantum Nondemolition Measurements

Recall that in Section 5.2.1, we used a so-called quantum nondemolition measurement to construct a control-NOT gate. In this appendix, we present the theory behind such measurements, especially as they relate to quantum optics.

It is well known that any two noncommuting Hermitian operators will satisfy a Heisenberg uncertainty relation [27]. Because of this fact, sequential measurements of the same quantum mechanical observable may not yield the same result. Consider the case of the position x and the momentum p, which satisfy the uncertainty relation

$$\Delta x \Delta p \geq \frac{\hbar}{2}. \tag{C.1}$$

Suppose we make a precise measurement of the position; we will then have introduced an uncertainty in the momentum such that

$$\Delta p \geq \frac{\hbar}{2\Delta x}. \tag{C.2}$$

Let us suppose we are working with a free particle. The Hamiltonian[7] is thus $H = p^2/2m$, and the subsequent evolution of the position is given by

$$\dot{x} = \frac{1}{i\hbar}[x, H] = \frac{p}{m}, \tag{C.3}$$

where we have used the Heisenberg picture equation of motion. Integrating this equation, we have

$$x(t) = x(0) + \frac{p(0)t}{m}, \tag{C.4}$$

which leads to [28]

$$[\Delta x(t)]^2 = [\Delta x(0)]^2 + \left[\frac{\Delta p(0)}{m}\right]^2 t^2 \geq [\Delta x(0)]^2 + \left[\frac{\hbar}{2m\Delta x(0)}\right]^2 t^2, \tag{C.5}$$

where we have used equation (C.2). This equation shows that due to the initial, precise position measurement we made, the accuracy of any subsequent position measurements is ruined. This can be a serious problem; we would like some way to measure a system without introducing this "back-action." In essence, we would like to be able to measure a quantum mechanical system in such a way that subsequent measurements of the same observable can be predicted deterministically based on some initial measurement. Such a scheme is known as a quantum nondemolition (QND) measurement.

Before giving an optical example of such a measurement, let us state the general conditions that must be met in a QND measurement scheme. In general, we want to

[7] Note that throughout this appendix, we use the symbol H to refer to the Hamiltonian operator rather than the Hadamard transform.

measure some observable A_s, which we term the signal observable. We will measure A_s by coupling it to another observable A_p, which we term the probe observable, in such a way that the evolution of A_s is unperturbed. Any observable A_s for which we can achieve this is termed a QND observable.

Let us write the Hamiltonian of the signal-probe system as

$$H = H_s + H_p + H_I, \tag{C.6}$$

where the three terms refer to the signal system, the probe system, and the interaction between the two, respectively. For a measurement of this coupled system to be a QND measurement, the following conditions must hold [28]:

• H_I must be a function of A_s. If it were not, a measurement of A_p could not give any information about A_s.

• A_p cannot be a constant of the motion, and so $[A_p, H_I] \neq 0$. If A_p were a constant of the motion, we could not use it to measure A_s, since its expectation value would not change.

• A_s must not be affected by its interaction with A_p; thus, $[A_s, H_I] = 0$. This is simply a statement of what we are trying to achieve by using a QND.

• The signal Hamiltonian H_s must not be a function of A_s^c, the canonical conjugate of A_s. This requirement ensures that the evolution of A_s will not be affected by its measurement, as in the case of the back-action discussed above for the position and momentum operators.

Let us now move on and consider QND measurements in quantum optics. Suppose we want to measure the number of photons in a particular beam, as we did in Section 5.2.1. Conventional measurement methods using photon counters are destructive, absorbing the photons in order to count them. As we saw in Section 5.2.1, however, there are times when we would like to allow the measured beam to continue to propagate through the system. Using nonlinear optics, we show below a QND method to measure photon number.

The nonlinear effect will we use is the Kerr effect. In electrodynamics, one often makes the approximation that dielectrics are linear media, so that the polarization is related to the electric field via $P = \epsilon_0 \chi E$, where ϵ_0 is the permittivity of free space and χ is the electric susceptibility. Linear dielectrics also have constant indices of refraction. In actuality, however, dielectrics are not linear, and the polarization must be written as a power series in E:

$$P = \epsilon_0 \left(\chi^{(1)} E + \chi^{(2)} E^2 + \chi^{(3)} E^3 + \ldots \right). \tag{C.7}$$

The index of refraction will be a similar power series in E. Generally, the first nonzero term in the expansion will be the dominant effect in the material; for example, the first order term in the index of refraction expansion gives rise to the Pockels effect. In crystals with a center of symmetry, however, the first order effect vanishes, and so the second order Kerr effect is the dominant nonlinear characteristic

of the material. The strength of the Kerr effect is proportional to $\chi^{(3)}$, the coefficient of the third power of E in the expansion of the polarization.

Now, consider Figure 21. Using the setup in this figure, we can measure the

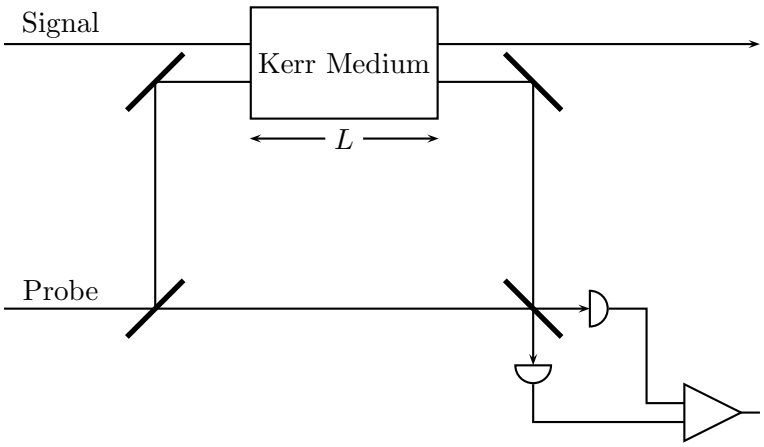

Figure 21: A diagram of a setup to measure the photon number of the signal beam. The signal beam is coupled to the probe beam via the nonlinear Kerr medium. The probe beam is sent through a Mach-Zehnder interferometer with the Kerr medium in one arm. A phase shift proportional to the number of photons in the signal beam will be induced in the part of the probe beam in the nonlinear leg of the interferometer, which can then be measured by the detectors at the end of the interferometer.

number of particles in the signal beam without destroying them. We achieve this by sending a probe beam through a Mach-Zehnder interferometer with a Kerr medium of length L in one leg. Both the signal beam and the probe beam will pass through the Kerr medium, which provides the coupling we need. We will use the Hamiltonian of equation (C.6), where [29]

$$\begin{aligned} H_s &= \hbar\omega_s \left(a_s^\dagger a_s + \frac{1}{2} \right) \\ H_p &= \hbar\omega_p \left(a_p^\dagger a_p + \frac{1}{2} \right) \\ H_I &= \hbar K a_s^\dagger a_s a_p^\dagger a_p. \end{aligned} \tag{C.8}$$

In these equations, a_i and $a_i^\dagger$ are the usual creation and annihilation operators for

the electromagnetic field, defined by [17]

$$\begin{aligned} a_i \,|\, n\rangle &= \sqrt{n}\,|\, n-1\rangle \\ a_i^\dagger \,|\, n\rangle &= \sqrt{n+1}\,|\, n+1\rangle \\ a_i \,|\, 0\rangle &= 0 \\ [a_i, a_j^\dagger] &= \delta_{ij}, \end{aligned} \tag{C.9}$$

the ω_i are the angular frequencies of the beams, and K is constant proportional to $\chi^{(3)}$. The QND observable we will be measuring is $a_s^\dagger a_s = n_s$, the photon number operator for the signal beam. The probe observable is the phase of the probe beam; due to the complications involved in defining a phase operator [17], we will not state the probe operator explicitly but will instead work only with the creation and annihilation operators of the probe field.

Now, in the Heisenberg picture, the equation of motion of the probe field annihilation operator is

$$\dot{a_p} = \frac{1}{i\hbar}[a_p, H_I] = -iKn_s[a_p, a_p^\dagger]a_p = -iKn_s a_p. \tag{C.10}$$

We can convert this equation into a spatial differential equation by recognizing that $t = -z/v_g$, where z is the distance traveled by the probe beam through the Kerr medium and v_g is the group velocity of the beam in the medium [29]. Integrating the resulting equation, we get

$$a_p{}' = e^{i\kappa n_s} a_p, \tag{C.11}$$

where $\kappa = KL/v_g$ and the primed annihilation operator refers to the probe field after the interaction with the signal field. Thus, we can see that a phase shift proportional to the number of photons in the signal beam has been introduced into the probe beam in one leg of the interferometer. The resulting interference pattern of the recombined probe beam will thus yield the number of photons in the signal beam.

Note, however, that we must have also induced an analogous phase shift in the signal beam as a result of this QND measurement. A more clever experimental setup can remove this phase, if it is unwanted [29]. As we have seen in Section 5.2.1, however, this extra phase can be useful, so for the purposes of this paper we do not correct it.

Acknowledgements

I would like to thank the faculty of the Swarthmore College Department of Physics and Astronomy, and particularly my advisor John Boccio, for four years of support and a wonderful education.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).

[2] R. Landauer, *IBM J. Res. Dev.* **5**, 183 (1961).

[3] D. Deutsch, Proc. R. Soc. Lond. A **400**, 97 (1985).

[4] J. R. Boccio, *Quantum Mechanics: Mathematical Structure, Physical Structure, and Applications in the Physical World, Volume 1* (unpublished).

[5] H. R. Lewis and C. Papadimitriou, *Elements of the Theory of Computation* (Prentice Hall, Upper Saddle River, NJ, 1997).

[6] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, Phys. Rev. Lett. **73**, 58 (1994).

[7] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, quant-ph/9906054, 1999.

[8] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Proc. R. Soc. Lond. A **454**, 339 (1998).

[9] D. Deutsch and R. Jozsa, Proc. R. Soc. Lond. A **439**, 553 (1992).

[10] L. K. Grover, Phys. Rev. Lett. **79**, 325 (1997).

[11] L. K. Grover, Phys. Rev. Lett. **80**, 4329 (1998).

[12] D. Bouwmeester, A. Ekert, and A. Zeilinger, *The Physics of Quantum Information* (Springer-Verlag, Berlin, 2000).

[13] P. W. Shor, SIAM J. Comp. **26**, 1484 (1997).

[14] A. Ekert and R. Jozsa, Rev. Mod. Phys. **68**, 733 (1996).

[15] E. Knill, R. Laflamme, and G. J. Milburn, quant-ph/0006088, 2000.

[16] P. G. Kwiat, J. R. Mitchell, P. D. D. Schwindt, and A. G. White, J. Mod. Opt. **47**, 257 (2000).

[17] R. Loudon, *The Quantum Theory of Light* (Oxford University Press, Oxford, 2000).

[18] J. C. Howell and J. A. Yeazell, Phys. Rev. Lett. **85**, 198 (2000).

[19] B. C. Sanders and G. J. Milburn, Phys. Rev. A **39** 694 (1989).

[20] Z. Y. Ou, Phys. Rev. Lett. **77**, 2352 (1996).

[21] E. Hecht, *Optics* (Addison Wesley Longman, Inc., Reading, MA, 2002).

[22] N. J. Cerf, C. Adami, and P. G. Kwiat, Phys. Rev. A **57**, R1477 (1998).

[23] M. Żukowski, A. Zeilinger, and M. A. Horne, Phys. Rev. A **55**, 2564 (1997).

[24] H. J. Bernstein, J. Math. Phys. **15**, 1677 (1974).

[25] J. C. Howell and J. A. Yeazell, Phys. Rev. A **61**, 012304 (1999).

[26] http://www.research.ibm.com/resources/news/20011219_quantum.shtml

[27] R. Shankar, *Principles of Quantum Mechanics* (Plenum Press, New York, NY, 1994).

[28] M. O. Scully and M. S. Zubairy, *Quantum Optics* (Cambridge University Press, Cambridge, 1997).

[29] F. X. Kärtner and H. A. Haus, Phys. Rev. A **47**, 4585 (1993).