Michele Mosca

Wolfson College

University of Oxford

Quantum Computer Algorithms

Thesis for the degree of Doctor of Philosophy at the University of Oxford

Trinity Term 1999

Abstract

Quantum computer algorithms are designed to exploit the properties of quantum physics. Quantum computations can be carried out in parallel on superpositions of exponentially many computational basis states and information about the outcomes of these computations can be measured via quantum interference. By casting quantum algorithms within the paradigm of quantum interferometry, most of the known quantum algorithms are clarified, unified, and generalised.

Further, the limitations of quantum computer algorithms is studied in the *black-box* model of computation, where the black-box reveals information about certain parameters (in this case Boolean values $X_1, X_2, \ldots, X_N$) and we wish to compute a function of these parameters. It is shown that the probability amplitudes of a quantum algorithm which has made T black-box calls is a polynomial in $X_1, X_2, \ldots, X_N$ of degree at most T. From this fact we can derive several lower bounds on the number of queries required to compute various functions of $X_1, X_2, \ldots, X_N$, such as their parity, or majority value. Further, any function that can be evaluated probabilistically using T queries on a quantum computer can be evaluate deterministically on a classical computer using at most $(4T)^6$ queries. Several other relationships are also established.

Lastly, techniques for better exploiting a fixed amount of quantum resources are illustrated and versions of quantum algorithms that can be implemented with very few qubits are suggested. I describe some of the first quantum algorithms and detail the first implementation, namely, the Deutsch algorithm.

Preface

Quantum computer algorithms are designed to exploit the properties of quantum physics. Few powerful quantum algorithms are known. Although it is interesting to find applications of the existing quantum algorithms, we really seek a *fundamentally* new one or an understanding of why we cannot find one.

As part of this endeavour I have sought to understand the fundamental aspects of the existing quantum algorithms. Quantum computations can be carried out in parallel on superpositions of exponentially many input states and information about the outcomes of these computations can be measured via quantum interference. By casting the known quantum algorithms within the paradigm of quantum interferometry, I was able (together with co-authors) to clarify, unite and generalise most of the known quantum algorithms in a series of papers [**CEMM98, CEH$^+$99, vDDE$^+$99, Mos98, Mos99, ME99, BHMT99**]. These results appear in Chapter 2. One class of algorithms that I have not yet studied closely are those for simulating physical systems. However, one of the algorithms developed for this purpose [**AL98**] turns out to be a rediscovery of the eigenvalue estimation algorithm described in [**CEMM98**] (which was based on the one in [**Kit95**]).

As a second part of this endeavour, I have sought to understand the limitations of quantum computers, identifying approaches that cannot work, and shedding light on why we find it so hard to find fundamentally new quantum algorithms. One approach is to study the limitations of quantum computer algorithms in the *black-box* model of computation, where we have a black-box that reveals information about certain parameters (such as a string X of Boolean values $X_1, X_2, \ldots, X_N$), and we wish to compute a function $F(X)$ of those parameters. We [**BBC+98**] showed that the probability amplitudes of a quantum algorithm which has made T black-box calls is a polynomial in $X_1, X_2, \ldots, X_N$ of degree at most T. From this fact we derive several lower bounds on the number of queries required to compute various functions of $X_1, X_2, \ldots, X_N$, such as their parity, or majority value. We show that any function that can be evaluated probabilistically using T queries on a quantum computer can be evaluate deterministically on a classical computer using at most $(4T)^6$ queries. Several other relationships are also established. This approach and the major results are presented in Chapter 3.

Some algorithms are interesting even if we never implement them. Perhaps they prove a new mathematical theorem or possess some other intrinsic beauty. However algorithms are often interesting because they solve a problem of practical interest in which case their realisability is of fundamental importance. A good example are algorithms for breaking cryptographic protocols. There is a large gap between theory and practice in quantum computer algorithms. For the first few years since the field of quantum computing was born [**Fey82, Deu85**] not even two qubit algorithms had been implemented. Recently, nuclear magnetic resonance technology was shown to be suitable for implementing quantum logic [**CFH96, GC97**]. However no practical way of scaling NMR quantum computers to large numbers of qubits is known and we are currently limited to a handful of qubits. We illustrate techniques for better exploiting a fixed amount of quantum resources and suggest versions of quantum algorithms that can be implemented with very few qubits. We have realised several of the first quantum computations [**JM98, JMH98, JM99**] and these are described in Chapter 4.

Acknowledgements

I am truly grateful to my supervisors Artur Ekert and Dominic Welsh for all their teaching, advice, help and support.

Many thanks to my co-authors and collaborators. I thank those who have generously hosted me, and the rest of the people at Oxford and around the world who have helped me learn about this field over the past three years. I also thank my examiners for the many helpful suggestions.

Special thanks to Adriano Barenco, Simon Benjamin, Harry Buhrman, Richard Cleve, Holly Cummins, Wim van Dam, Mark Ettinger, Matt Gevaert, Rasmus Hvass Hansen, Bernard Howes, Peter Høyer, Hitoshi Inamori, Jonathan Jones, Chiara Macchiavello, Frédéric Magniez, Juan Poyatos, Miklos Santha, Alain Tapp, Vlatko Vedral, Sue Witney, and Ronald de Wolf.

I am grateful to Richard Booth, Pete Seeviour, Kevin Thacker, and the C.E.S.G. who funded this D.Phil. Many thanks to Wolfson College for supporting me in so many ways over the past four years, this last year as the Robin Gandy Junior Research Fellow. I thank the Wolfson College Boat Club and the members of the many other clubs and societies in which I have participated during the preparation of this thesis.

Many special thanks to my parents, brother, family, and friends for their love and support.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

An algorithm is a procedure for performing a task. For example, a cake recipe is an algorithm that takes as input standard kitchen ingredients and uses standard kitchen equipment to output a cake (sometimes with only a bounded probability of success). A computer is a physical device that helps us process information, and an information processing task can always be translated into physical one. Theorists tend to work with an abstract model of computation, but when pondering the capabilities and limitations of a computing device for some practical reason, it is important not to forget the relationship between computing and physics. Dramatic examples are the physical attacks used to break certain cryptosystems that made use of RSA encryption. These protocols were designed with the hope that the security of the system relies on the difficulty of solving a problem which is closely related to factoring large numbers. However, without factoring any integers, people have been able correctly to infer the encryption keys. They did so by cleverly probing physical devices performing the encryption for clues leaked during the physical act of computing the ciphertext (see for example, [**Koc96**]).

A prototype computer is the *Turing machine* (see e.g. [**GJ79, AHU74, Pap94, MR95, Wel88**]). The key ingredients are a piece of hardware which runs according to some software and uses additional memory to carry out the software instructions and produce some output. A sufficiently complex Turing machine with an arbitrarily large supply of memory can simulate any other Turing machine M provided the input contains a description of how M works. Such a

Turing machine is called a *universal Turing machine.* If we equip a Turing machine with the ability to flip a fair coin, we get a *probabilistic Turing machine.* A probabilistic Turing machine computers a function by outputting an answer that is correct with probability at least $\frac{2}{3}$ (the average is over all possible outcomes of the coin flips, *not* the different inputs to the function f). A universal Turing machine can simulate any other Turing machine and in fact, they can simulate any other reasonable computer known to date. Furthermore, a universal probabilistic Turing machine can simulate any Turing machine (and any other reasonable device known to date) with at most a polynomial overhead (this is the strong form of the Church-Turing thesis). By *polynomial overhead* we mean that if our machine uses T units of some resource (usually time or space), then the universal probabilistic Turing machine will simulate it using at most $p(T)$ units of that resource where p is a fixed polynomial (often referred to as $poly(T)$). To do so, we specify the way our reasonable computer works, and then the universal Turing machine simulates our machine. Another model of computation is that of a *uniform families of acyclic circuits* (see [**Pap94**] for example). An acyclic circuit C_n is described by a circuit diagram which has n input wires, and at each time step t each wire can enter at most one gate G. The term circuit seems to correspond to a particular physical implementation. The acyclic circuits look more like an array or network of gates, which is the terminology we will use in the quantum setting. The gates come from a finite family of gates which take information from input wires and output information along some output wires. A family of acyclic circuits is a family of circuits $\{C_n | n \in \mathbb{Z}^+\}$. The family is *uniform* if we can easily construct C_n (say by an appropriately resource-bounded Turing machine - see [**GJ79, Pap94, MR95, Cle99**], for a discussion). A deterministic Turing machine can compute whatever a probabilistic one can (in finite time) by trying all possible outcomes of the coin flips.

There is a well-defined family of functions that can be computed in a finite number of steps and those that cannot (see e.g. [**Rog87**] or [**Dav82**]). However not all problems which have "effective procedures" for solving them will have "efficient procedures", that is procedures that use a 'reasonable' amount of resources. The *computational complexity* of a problem or task attempts to quantify the amount of resources, such as time, space, or energy, necessary to perform the task for an input of size n. We will restrict attention to worst-case complexities, that is the complexity of a problem on the worst-case input of a specific size. In particular, when using the acyclic circuit model, a natural measure of complexity is the number of gates used in the circuit C_n.

In studying the minimum energy requirements of any computing device, Bennett [**Ben73**] observed that the amount of energy necessary can be made arbitrarily small if the computer has reversible components, that is if each operation was logically reversible. For example, the NOT operation is reversible, but the *AND* operation is not reversible (see figure 1.1).

x	$\overline{x}$
0	1
1	0

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

FIGURE 1.1. The *NOT* and *AND* gate. Note that the *NOT* gate is logically reversible while the *AND* is not.

He also showed how any irreversible classical algorithm can be transformed into a reversible one. This is easy to see in the circuit model of computation. Each gate in a finite family of gates can be made reversible by adding some additional input and output wires if necessary. For example, the AND gate can be made reversible by adding an additional input wire and two additional output wires (see figure 1.2). Note that additional information necessary to reverse the operation is now kept

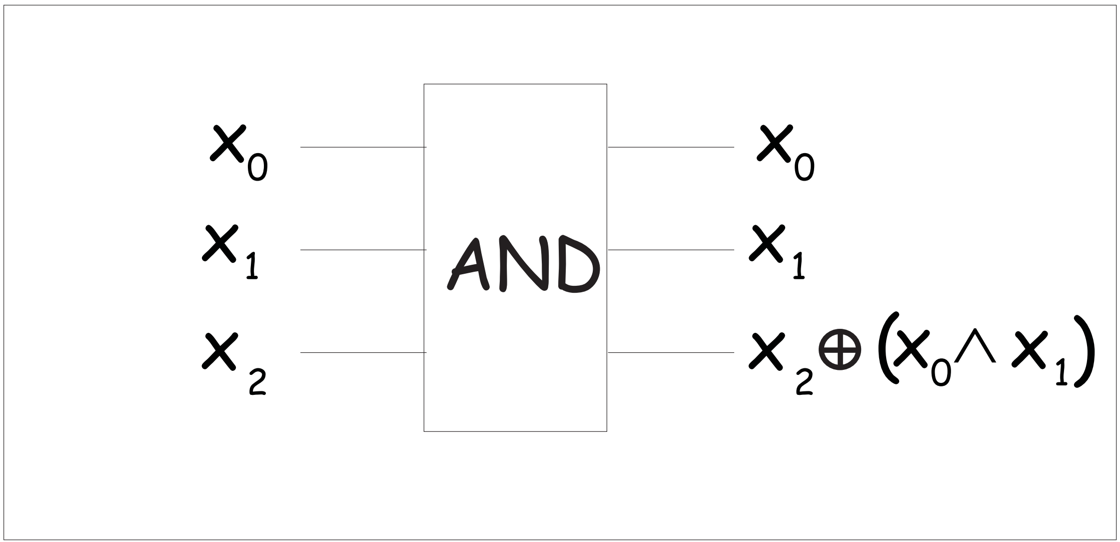

FIGURE 1.2. The reversible AND gate keeps a copy of the inputs and adds the AND of x_0 and x_1 (denoted $x_1 \wedge x_2$) to the value in the additional input bit. Note that by fixing the additional input bit to 0 and discarding the copies of the x_0 and x_1 we can simulate the non-reversible AND gate.

instead of being somehow absorbed into the environment, as is done in any logically irreversible computation. By simply replacing all the non-reversible components with their reversible counterparts [1], we get a reversible version of the algorithm.

[1]Our universal set of reversible gates might not contain this particular gate, but a fixed size circuit made from our universal set of gates could be used instead.

If we start with the output, and run the circuit backwards, we obtain the input again. The reversible version might introduce some constant number of additional wires for each gate. Thus if we have a non-reversible algorithm which used time T (the *depth* of the circuit) and space S, we can easily construct a reversible version that used a total of $O(T+S)$ space and time T. Furthermore, the additional 'junk' information generated by making each gate reversible can also be erased at the end of the computation by first copying the output, and then running the reversible algorithm in reverse to obtain the starting state again. This is illustrated in figure 1.3. Bennett subsequently showed how to turn a computation using time T and space S in a reversible one using time $O(T^{1+\epsilon})$ and space $O(S \log T)$ or time $O(T)$ and space $O(ST^{\epsilon})$, for any $\epsilon > 0$ (see [**Ben89**], [**LTV98**]).

This beautiful theory of reversible computation was sparked by a simple question about the physics of computation. It is important to note that Turing machines and modern-day computers implicitly refer only to notions of physics that are over a century old and known as 'classical physics'. This past century a new theory has been developed which includes the theory of quantum mechanics.

In the next section we take a closer look at quantum physics and its relationship to computation.

1.1. Quantum Physics

One experimental set-up that exemplifies in a very simple way some of the main principles of quantum mechanics is the *Mach-Zehnder* interferometer. The interferometer is an apparatus that allows us to measure the interference of photons following two different paths. It consists of two half-silvered mirrors (or *beam-splitters*), some full-mirrors to help direct the photons along one of two desired paths (we will not illustrate these full-mirrors, but whenever you see a light path take a sharp turn, there implicitly is a mirror there!), and two photon detectors

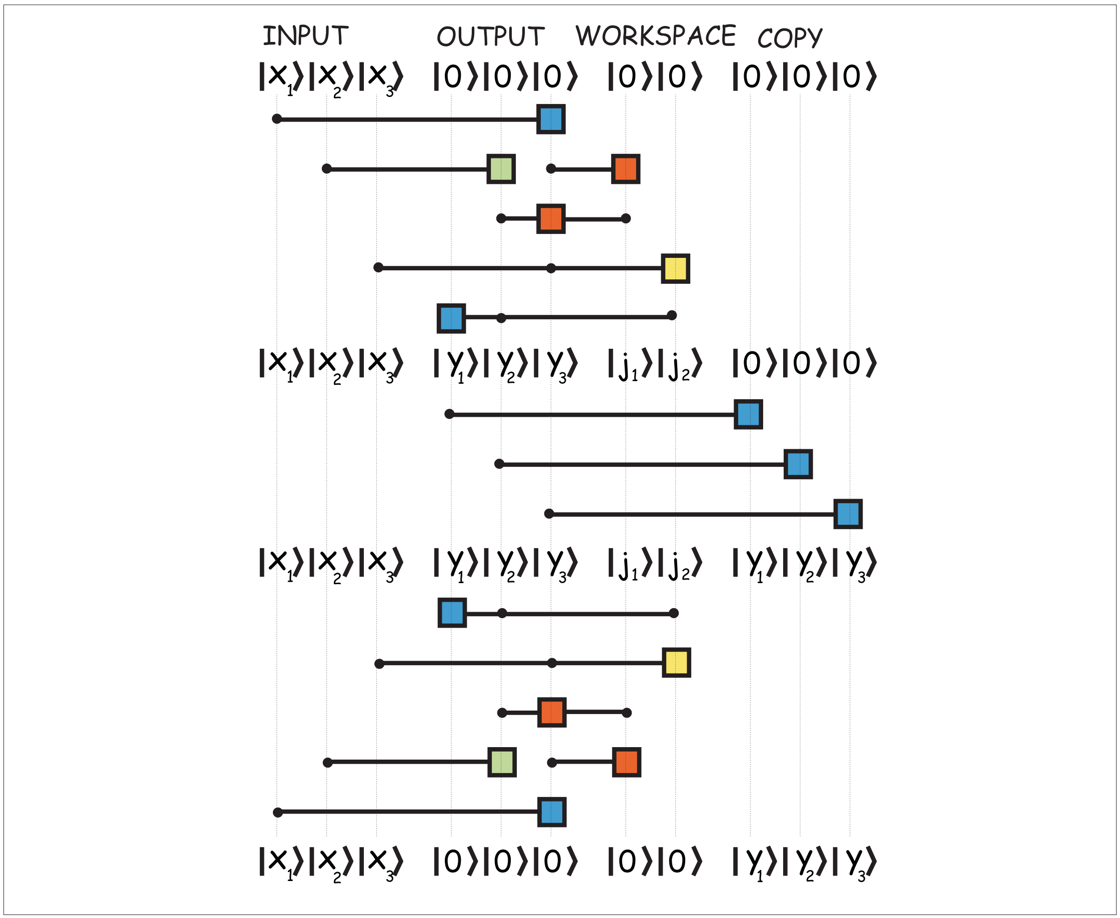

FIGURE 1.3. A network for computing $f(x)$ reversibly. Start with the input. Compute $f(x)$ using reversible logic, possibly generating some extra 'junk' bits of information. Copy the output $f(x)$ to another register (this is a 'classical' state, so copying simply corresponds to performing a controlled-NOT between every qubit of the output register and a qubit of the copy register). Run the circuit for f backwards (replacing each gate by its inverse gate - here we illustrate self-inverse gates) to erase the contents of the output and workspace registers.

at the end of each path. Let us denote a photon in the upper path as being in state $|\,0\rangle$, and in the lower path as state $|\,1\rangle$. We first describe the behaviour of the half-silvered mirrors (see figure 1.4). We start with a single photon in the $|\,0\rangle$

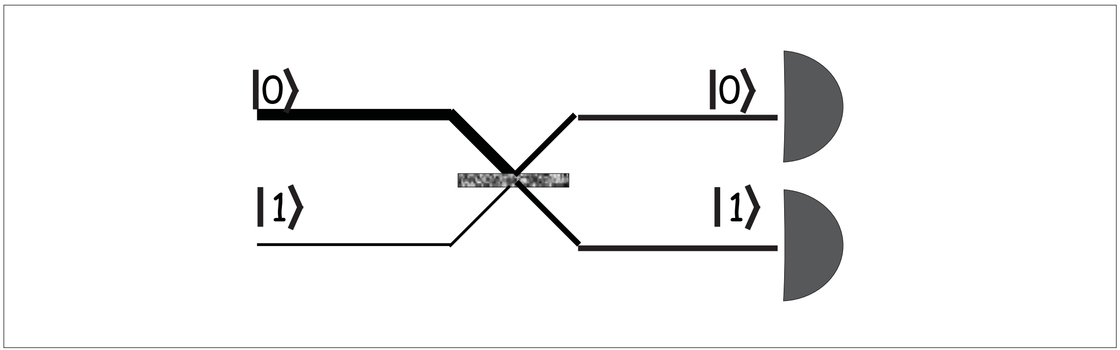

FIGURE 1.4. When a photon enters the beam-splitter along the $|\,0\rangle$ path, we get a single click at either the $|\,0\rangle$ or $|\,1\rangle$ detector, each with probability 50%.

path and set up the photon detectors along both possible directions the photon could take. When the photon passes through the half-silvered mirror, we notice that exactly one of the two photon detectors clicks (this is an ideal situation). We never get two "half-clicks", or other fractions of a click. This discreteness is one of the main features of quantum mechanics. We repeat this experiment many times, and each time exactly one detector clicks, each one about half of the time. The simplest explanation is that the photon impinging upon the half-silvered mirror takes one of the two paths with 50% probability - a simple coin flip.

Now let us repeat this experiment, except this time with two beam-splitters aligned so that no matter which of the two paths the photon takes it will arrive at the second beam-splitter at the same time (see figure 1.5).

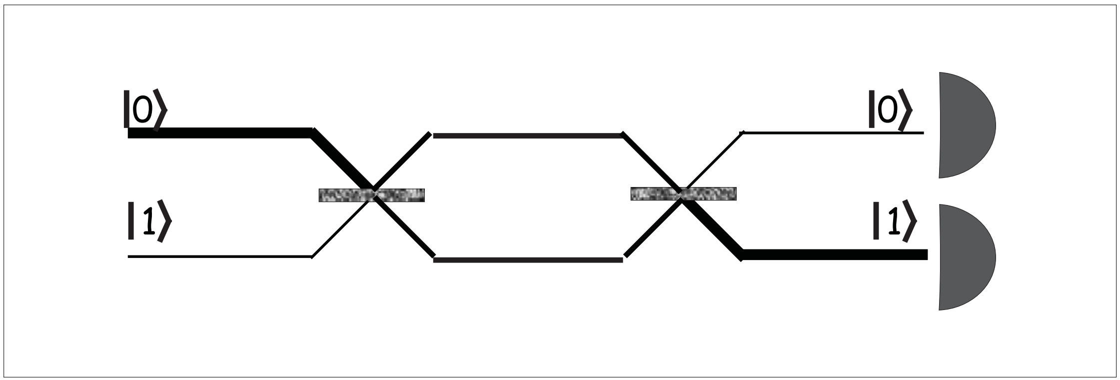

FIGURE 1.5. Contrary to our classical intuition, all of the photons are detected at detector $|1\rangle$.

Assuming that a photon going through a beam-splitter simply takes one of the two paths at random, then we expect roughly half of the photons to be detected at the $|1\rangle$ detector and the other half at the $|0\rangle$ detector. Quite logical, but false. Set up this apparatus, and you will notice that you only detect photons at the $|1\rangle$ detector! By inserting an appropriate *phase-shifter* (usually a piece of glass), along the $|0\rangle$ path, the probability distribution shifts from only $|1\rangle$ detections to only $|0\rangle$ detections. We call this a π-phase-shifter. Thus we can reliably distinguish between the presence and absence of this π-phase-shifter. Further, by placing a ϕ_0-phase-shifter along the $|0\rangle$ path and ϕ_1-phase-shifter (we can adjust the phase shift ϕ of a phase shifter by changing internal properties such as thickness and refractive index) along the $|1\rangle$ path the proportions are $\sin^2(\frac{\phi_0-\phi_1}{2})$ $|0\rangle$ detections and $\cos^2(\frac{\phi_0-\phi_1}{2})$ $|1\rangle$ detections.

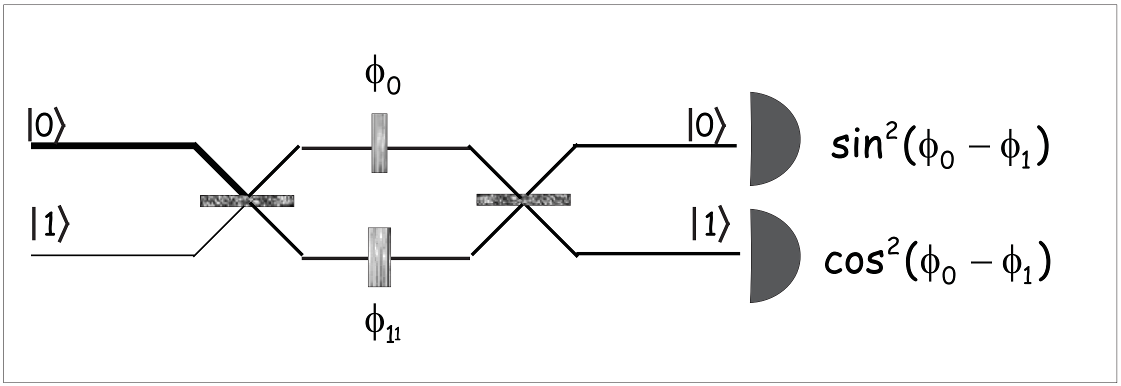

FIGURE 1.6. The phase shifters induce an interference pattern depending on the difference between the two phase shifts.

So our simple explanation needs to be revamped. Our current understanding is the following: when exiting the first beam-splitter, the photon is in the state $\frac{i}{\sqrt{2}}\,|\,0\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle$, that is a linear combination or *superposition* of the two states. The amplitude of $|\,0\rangle$ tells us the probability of observing the photon in path $|\,0\rangle$ if we set up our detectors. You obtain the probability by squaring the modulus of the amplitude. Thus we measure the photon in path $|\,0\rangle$ with probability $\left|\frac{i}{\sqrt{2}}\right|^2 = \frac{1}{2}$. Similarly, if we send a photon through the beam-splitter starting in the $|\,1\rangle$ path it would come out in the state $\frac{1}{\sqrt{2}}\,|\,0\rangle + \frac{i}{\sqrt{2}}\,|\,1\rangle$ (that is the reflected beam picks up a phase of i). The phase shifts of ϕ_0 and ϕ_1 change the state from $\frac{i}{\sqrt{2}}\,|\,0\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle$ to $\frac{ie^{i\phi_0}}{\sqrt{2}}\,|\,0\rangle + \frac{e^{i\phi_1}}{\sqrt{2}}\,|\,1\rangle$. The second beam-splitter acts in the same way as the first one, that is it maps $|\,0\rangle \to \frac{i}{\sqrt{2}}\,|\,0\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle$ and $|\,1\rangle \to \frac{1}{\sqrt{2}}\,|\,0\rangle + \frac{i}{\sqrt{2}}\,|\,1\rangle$, and therefore it maps $\frac{ie^{i\phi_0}}{\sqrt{2}}\,|\,0\rangle + \frac{e^{i\phi_1}}{\sqrt{2}}\,|\,1\rangle$ to

$$\text{(1)}\qquad \frac{ie^{i\phi_0}}{\sqrt{2}}\left(\frac{i}{\sqrt{2}}\,|\,0\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle\right) + \frac{e^{i\phi_1}}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\,|\,0\rangle + \frac{i}{\sqrt{2}}\,|\,1\rangle\right).$$

Thus a photon can arrive at the $|\,0\rangle$ detector by two different paths, one with amplitude $\left(\frac{ie^{i\phi_0}}{\sqrt{2}}\right)\left(\frac{i}{\sqrt{2}}\right) = \frac{-e^{i\phi_0}}{2}$, and the other path with amplitude $\frac{e^{i\phi_1}}{2}$. The total amplitude of the state $|\,0\rangle$ is $\frac{-e^{i\phi_0}}{2}+\frac{e^{i\phi_1}}{2}=ie^{i\frac{\phi_0+\phi_1}{2}}\sin(\frac{\phi_0-\phi_1}{2})$. Similarly we see that the total amplitude of the state $|\,1\rangle$ is $ie^{i\frac{\phi_0+\phi_1}{2}}\cos(\frac{\phi_0-\phi_1}{2})$ and the state in equation (1) is equal to $ie^{i\frac{\phi_0+\phi_1}{2}}\left(\sin(\frac{\phi_0-\phi_1}{2})\,|\,0\rangle+\cos(\frac{\phi_0-\phi_1}{2})\,|\,1\rangle\right)$. Note that all that determines the magnitude of the amplitudes is the difference between ϕ_0 and ϕ_1. In general the photon could be in any linear combination or *superposition* state of the form

$$\alpha_0\,|\,0\rangle+\alpha_1\,|\,1\rangle$$

where $|\alpha_0|^2+|\alpha_1|^2=1$. This state can also be described by the vector $(\alpha_0,\alpha_1)\in\mathbb{C}^2$, where the basis vectors $(1,0)$ and $(0,1)$ correspond to $|\,0\rangle$ and $|\,1\rangle$ respectively. The action of the beam-splitter can thus be described by the 2×2 matrix

$$\begin{pmatrix} \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix}$$

and the phase-shifters by the matrix

$$\begin{pmatrix} e^{i\phi_0} & 0 \\ 0 & e^{i\phi_1} \end{pmatrix}.$$

Suppose we had two such Mach-Zehnder apparatuses set up with the state of each of photon described by $\frac{1}{\sqrt{2}}\,|\,0\rangle+\frac{1}{\sqrt{2}}\,|\,1\rangle$. The 2-photon system would be described by the state

$$(2)\qquad \left(\frac{1}{\sqrt{2}}\,|\,0\rangle+\frac{1}{\sqrt{2}}\,|\,1\rangle\right)\left(\frac{1}{\sqrt{2}}\,|\,0\rangle+\frac{1}{\sqrt{2}}\,|\,1\rangle\right)=\sum_{x_1,x_2\in\{0,1\}}\frac{1}{2}\,|\,x_1\rangle\,|\,x_2\rangle\,.$$

We will often denote a state $|\,x_1\rangle\,|\,x_2\rangle$ as $|\,x_1\rangle\otimes|\,x_2\rangle$ or simply as $|\,x_1x_2\rangle$. The two photon system could in fact exist in any linear combination of the form

$$(3)\qquad \sum_{x\in\{00,01,10,11\}}\alpha_x\,|\,x\rangle\,,$$

where $\sum |\alpha_x|^2 = 1$, and $|\alpha_x|^2$ is the probability of observing the state $|\,x\rangle = |\,x_1\rangle\,|\,x_2\rangle$ if we set up our four detectors along the possible paths. This means the 2-photon system can be in a state that is not simply two independent 1-photon systems as we describe in equation (2). Consider for example the state, also known as an Einstein-Podolsky-Rosen (EPR) [**EPR35**] state,

$$\frac{1}{\sqrt{2}}\,|\,0\rangle\,|\,0\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle\,|\,1\rangle\,. \tag{4}$$

This state cannot be factored into the product of two independent states. Note that superpositions of the form in equation (3) can also be denoted as vectors in $\mathbb{C}^4$, $(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11})$. In general, if we have two systems in states $(a_0\,|\,0\rangle + a_1\,|\,1\rangle)$ and $(b_0\,|\,0\rangle + b_1\,|\,1\rangle)$, the joint system

$$a_0 b_0\,|\,00\rangle + a_0 b_1\,|\,01\rangle + a_1 b_0\,|\,10\rangle + a_1 b_1\,|\,11\rangle$$

is denoted by the vector

$$(a_0 b_0, a_0 b_1, a_1 b_0, a_1 b_1) = (a_0, a_1) \otimes (b_0, b_1)$$

the tensor product of the vectors describing the subsystems. We can prove that state (4) cannot be factorised into two independent one-qubit states by showing that the vector $(\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}})$ cannot be decomposed as the tensor product of two complex vectors $(a_0, a_1) \otimes (b_0, b_1)$. The two particles are strongly correlated in a way we refer to as being *entangled*. This notation where $(1, 0, 0, 0)$ is denoted $|\,00\rangle$ is Dirac's *bra-ket* notation [**Dir58**].

However, the path of a photon in a Mach-Zehnder interferometer is just one example of a system with two discrete states which we label $|\,0\rangle$ and $|\,1\rangle$. Any quantum system with at least two reliably distinguishable states (for example, two discrete energy levels which can represent the logical states 0 and 1) can be used as a qubit and prepared in a superposition of its logical states

$$(5) \qquad |\,\Psi\rangle = \alpha_0\,|\,0\rangle + \alpha_1\,|\,1\rangle\,,$$

where $|\alpha_0|^2 + |\alpha_1|^2 = 1$, and as before, $|\alpha_0|^2$ is the probability of observing $|\,0\rangle$ and $|\alpha_1|^2$ is the probability of observing $|\,1\rangle$.

Other candidates for physical realisation of qubits are polarised photons, trapped ions and nuclear spins. The physical details of preparing, manipulating and measuring a qubit depend on the type of physical realisation. For example, a proton has a nuclear spin of $\frac{1}{2}$ and therefore when in a static magnetic field oriented along the z-axis it has two possible energy levels, one corresponding to "spin-up" which we call state $|\,1\rangle$ and one corresponding to "spin-down" which we call state $|\,0\rangle$. If the spin is in state $|\,0\rangle$ and we wish to apply the NOT operation, we can send in a radio-frequency pulse of appropriate intensity and duration, with frequency corresponding to the energy difference between the $|\,1\rangle$ and $|\,0\rangle$ states. This causes a spin initially in the state $|\,0\rangle$ to evolve into the state $|\,1\rangle$ and vice versa. We can also apply the same pulse for half of the duration to give us an equally weighted superposition of $|\,0\rangle$ and $|\,1\rangle$, similar to the state of the photon after it goes through the beam-splitter [2] Just as going through a second beam-splitter sends the photon to the state $|\,1\rangle$, applying another pulse of half the duration of the NOT pulse sends the spin to the state $|\,1\rangle$ thereby completing the NOT operator. This operation is often called the square-root of NOT, $\sqrt{NOT}$.

[2]When we view the beam-splitter (from the previous example) and radio-frequency pulse as quantum objects, we see that the interaction between the qubit and the apparatus does leave in the apparatus a very small trace of information about the state of the qubit. The degree of entanglement between the qubit and apparatus in the case of a beam-splitter or a large radio-frequency pulse is negligible, so in practice we ignore it.

In general a single spin can be prepared, by a suitable sequence of various types of pulses, in any superposition as in equation (5).

We can also consider a register composed of n qubits. Any classical register of n bits can store just one of 2^n strings at a time. Suppose we prepare a quantum register of 3 qubits, say nuclear spins, in the state $|0\rangle|0\rangle|0\rangle$ and we apply a suitably tuned radio-frequency pulse to evolve the first qubit into the state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Then the three-qubit system will be in the state

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

A similar pulse applied to the second qubit will act in parallel on both configurations $|0\rangle|0\rangle|0\rangle$ and $|1\rangle|0\rangle|0\rangle$ to produce

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

and another pulse applied to the third qubit creates a superposition of all eight configurations 000, 001, . . . , 111. With n qubits we can similarly prepare superpositions of all 2^n configurations. All 2^n possible strings are now physically present, and any operation we apply to the register of n qubits will be applied to all configurations in parallel. For example, if we apply an operation U only once to this system, we will produce the state

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

An n-qubit system can be in any state of the form

$$\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$$

where $\sum |\alpha_x|^2 = 1$ and $|\alpha_x|^2$ is the probability of observing the state $|\, x\rangle = |\, x_1\rangle\, |\, x_2\rangle \ldots |\, x_n\rangle$. The amplitude α_x of $|\, x\rangle$ in the state $|\, \Psi\rangle$ is the inner product $\langle\, x|\, \Psi\rangle$ between the state vectors $|\, \Psi\rangle$ and $|\, x\rangle$. We can thus also write

$$|\, \Psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x \,|\, x\rangle = \sum_{x\in\{0,1\}^n} |\, x\rangle\, \alpha_x = \sum_{x\in\{0,1\}^n} |\, x\rangle\, \langle\, x|\, \Psi\rangle.$$

Any linear transformation U that maps states $\sum_x \alpha_x \,|\, x\rangle$ satisfying

$$\sum |\alpha_x|^2 = 1$$

to other states $\sum_x \beta_x \,|\, x\rangle$ satisfying

$$\sum |\beta_x|^2 = 1$$

must be *unitary*. Unitary transformations are precisely those that satisfy $U^* = U^{-1}$, where U^* denotes the conjugate transpose of U (often denoted $U^\dagger$ in the literature).

A quantum operation such as applying a radio-frequency pulse or putting a beam-splitter in the path of a photon, can be described without reference to any particular quantum technology. We will describe operations on qubits in terms of some prescribed elementary operations which we refer to as quantum logic gates. We have already described the NOT and $\sqrt{NOT}$. A similar operation which maps

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

and

$$|\, 1\rangle \rightarrow \frac{1}{\sqrt{2}}\,|\, 0\rangle - \frac{1}{\sqrt{2}}\,|\, 1\rangle$$

is called a *Hadamard gate* denoted by H. A gate such as the phase shifter of angle θ along the $|1\rangle$ path, that maps

$$\begin{aligned} |0\rangle &\rightarrow |0\rangle \\ |1\rangle &\rightarrow e^{i\theta}|1\rangle \end{aligned} \tag{6}$$

is called a *phase gate* R_θ.

These one-qubit gates alone will not allow us to perform every unitary operation possible on n-qubits. They will not even allow us to create entangled states from unentangled ones. We need a non-trivial two-qubit gate, such as the controlled-NOT which maps

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |01\rangle &\rightarrow |01\rangle \\ |10\rangle &\rightarrow |11\rangle \\ |11\rangle &\rightarrow |10\rangle. \end{aligned}$$

We call this a controlled-NOT because the NOT is effected to the second qubit (or *target* qubit) when the first qubit (or *control* qubit) is in the state $|1\rangle$. Another non-trivial two-qubit gate is the controlled-R_θ that maps

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |01\rangle &\rightarrow |01\rangle \\ |10\rangle &\rightarrow |10\rangle \\ |11\rangle &\rightarrow e^{i\theta}|11\rangle \end{aligned}$$

(note the symmetry between the control and target qubits).

We need such two-qubit gates to map the state $|0\rangle|0\rangle$ to the state $\frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle$. We can do this by first applying a Hadamard gate to the first qubit to create $\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)|0\rangle = \frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle$. Applying a controlled-$NOT$ then yields the EPR state $\frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle$. As mentioned earlier, this state *cannot* be decomposed as the product of two independent one-qubit states, and is thus *entangled*. Superpositions that *can* be so decomposed are called *unentangled* or *separable*. Qubits typically become very entangled in the intermediate stages of most interesting quantum computations. Entangled pairs of qubits are also a valuable physical resource in other applications of quantum information theory, such as quantum communication and quantum key exchange.

The fact that one qubit in an entangled pair of qubits cannot be described as a superposition independently of the other qubit forces us to reconsider the mathematical description of quantum states in terms of state vectors. For example, how should we describe the state of one qubit of an EPR pair? Such a state can be described in terms of *density operators* [**CTDL77**] which provide a more general mathematical description of a quantum state. For a single qubit in the superposition state (or *pure state*) $|\,\Psi\rangle = \alpha\,|\,0\rangle + \beta\,|\,1\rangle$, the corresponding density operator is the projector $\rho = |\,\Psi\rangle\,\langle\,\Psi| = \alpha\alpha^*\,|\,0\rangle\,\langle\,0| + \alpha\beta^*\,|\,0\rangle\,\langle\,1| + \beta\alpha^*\,|\,1\rangle\,\langle\,0| + \beta\beta^*\,|\,1\rangle\,\langle\,1|$. This operator can be written in the $\{|\,0\rangle\,,|\,1\rangle\}$ basis as

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \cdot \begin{pmatrix} \alpha^* & \beta^* \end{pmatrix} = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}.$$

The probability of measuring $|\,0\rangle$ is equal to

$$|\alpha|^2 = \alpha\alpha^* = \langle\,0|\,\Psi\rangle\langle\,\Psi|\,0\rangle = Tr(\langle\,0|\,\Psi\rangle\langle\,\Psi|\,0\rangle) = Tr(|\,\Psi\rangle\langle\,\Psi||\,0\rangle\langle\,0|) = Tr(\rho|\,0\rangle\langle\,0|).$$

In general, given a state described by a density matrix ρ, the probability of measuring a state $|\,\Phi\rangle$ equals

$$Tr(\rho\,|\,\Phi\rangle\,\langle\,\Phi|).$$

Note that for any unitary operator U acting on a qubit, the density operator $|\,\Psi'\rangle\,\langle\,\Psi'|$ of the state $|\,\Psi'\rangle = U\,|\,\Psi\rangle$ is equal to $U\rho U^*$ where $\rho = |\,\Psi\rangle\,\langle\,\Psi|$.

With any quantum state we can associate a density operator, ρ, such that ρ is positive and the trace of ρ, $Tr(\rho) = 1$ (see [**CTDL77**]). If we apply the unitary operator U to that state, the outcome state has density operator $U\rho U^*$.

If we wish to describe a qubit that is entangled to another we start by writing the quantum state vector of the two qubit system

$$|\,\Psi\rangle = \sum_{w,x\in\{0,1\}} \alpha_{wx}\,|\,wx\rangle$$

then we construct the density operator of the two qubits

$$\rho = |\,\Psi\rangle\,\langle\,\Psi| = \sum_{w,x,y,z\in\{0,1\}} \alpha_{wx}\alpha^*_{yz}\,|\,wx\rangle\,\langle\,yz|\,.$$

To compute the density operator ρ_1 of the first qubit, we perform the partial trace (see [**CTDL77**]) over the second qubit to get

$$\rho_1 = Tr_2(\rho) = \sum_{w,y,x\in\{0,1\}} \alpha_{wx}\alpha^*_{yx}\,|\,w\rangle\,\langle\,y|\,.$$

Similarly we can compute the density operator ρ_2 of the second qubit by performing the partial trace over the first qubit to get

$$\rho_2 = Tr_1(\rho) = \sum_{w,x,z\in\{0,1\}} \alpha_{wx}\alpha^*_{wz}\,|\,x\rangle\,\langle\,z|\,.$$

For example, the state

$$|\,\Psi\rangle = \frac{1}{\sqrt{2}}\,|\,00\rangle + \frac{1}{2}\,|\,01\rangle + \frac{i}{2}\,|\,11\rangle$$

is described by the density operator

$$\rho = \begin{pmatrix} \frac{1}{2} & \frac{1}{2\sqrt{2}} & 0 & \frac{-i}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & \frac{1}{4} & 0 & \frac{-i}{4} \\ 0 & 0 & 0 & 0 \\ \frac{i}{2\sqrt{2}} & \frac{i}{4} & 0 & \frac{1}{4} \end{pmatrix}$$

and, after taking the partial traces we get

$$\rho_1 = Tr_2(\rho) = \begin{pmatrix} \frac{3}{4} & \frac{-i}{4} \\ \frac{i}{4} & \frac{1}{4} \end{pmatrix}$$

$$\rho_2 = Tr_1(\rho) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & \frac{1}{2} \end{pmatrix}.$$

If the bit value is measured on the two qubits then with the probabilities $\frac{1}{2}$, $\frac{1}{4}$,and $\frac{1}{4}$ one obtains respectively 00 , 01, and 11. The outcome 10 has the probability zero. These probabilities are given by the diagonal elements of ρ. Suppose the second qubit was taken away so that one can measure only the bit value on the first qubit. In this case one obtains the bit values 0,1 with probabilities $\frac{3}{4}$, $\frac{1}{4}$ respectively, which are given by the diagonal elements of ρ_1. The diagonal elements of ρ_2 give probabilities of obtaining 0 or 1 when the measurement is performed on the second qubit alone. The partial trace of larger systems is defined similarly [**CTDL77**].

So how do we actually effect an operation, such as the controlled-NOT, that will create entanglement? Let us return to our example of a spin-$\frac{1}{2}$ particle, say a proton, in a magnetic field oriented along the z-axis. The energy of a proton in the spin-up state is $\frac{\omega}{2}\hbar$ and in the spin-down state is $\frac{-\omega}{2}\hbar$ where $\hbar = 6.626 \times 10^{-34} J \cdot s$ and $\frac{\omega}{2\pi}$ is a frequency. The frequency $\frac{\omega}{2\pi}$ depends on the strength B of the magnetic field and on the type of nucleus according to $\omega = \gamma B$, where γ is the *gyromagnetic ratio* (which for a proton is about $2.675 \times 10^{-6} Hz/Tesla$). Thus a proton in a field of strength $9.4 Tesla$ has a frequency $\omega = 400 MHz$. (The strength of magnetic fields is often given in terms of the frequency $\frac{\omega}{2\pi}$ the magnetic field would induce on a proton.) We thus denote the energy operator H, or the *Hamiltonian*, by the

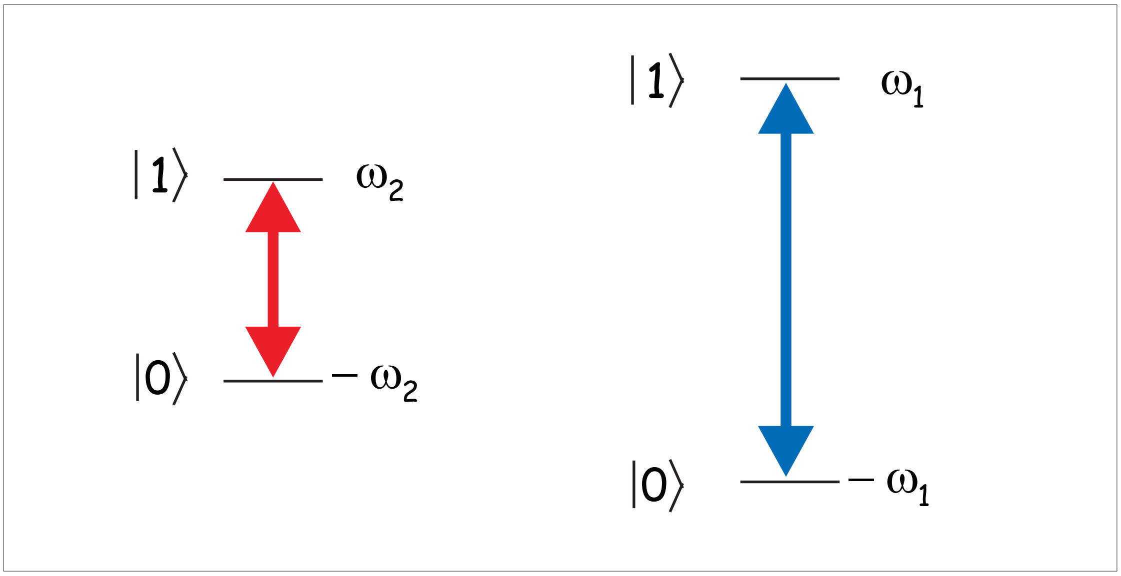

FIGURE 1.7. The energy eigenstates of the energy operator are $|\,0\rangle$ with energy eigenvalue $\frac{\omega\hbar}{2}$ and $|\,1\rangle$ with energy eigenvalue $\frac{-\omega\hbar}{2}$.

matrix

$$\begin{pmatrix} \frac{\omega\hbar}{2} & 0 \\ 0 & \frac{-\omega\hbar}{2} \end{pmatrix} = \frac{1}{2}\omega\hbar\sigma_z$$

where

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

As mentioned earlier, we can effect the NOT gate by sending a radio-frequency pulse of frequency $\frac{\omega}{2\pi}$ of appropriate intensity and duration. Suppose we had two independent nuclei in the same magnetic field, one with energy eigenvalues $\pm\frac{\omega_1\hbar}{2}$ and the other with energy eigenvalues $\pm\frac{\omega_2\hbar}{2}$. The joint system has a Hamiltonian equal to

$$\frac{\hbar}{2}\begin{pmatrix} \omega_1+\omega_2 & 0 & 0 & 0 \\ 0 & \omega_1-\omega_2 & 0 & 0 \\ 0 & 0 & -\omega_1+\omega_2 & 0 \\ 0 & 0 & 0 & -\omega_1-\omega_2 \end{pmatrix} = \frac{1}{2}\omega_1\hbar\sigma_z\otimes\mathbb{I}+\frac{1}{2}\omega_2\hbar\mathbb{I}\otimes\sigma_z$$

as illustrated in figure 1.8.

However two such nuclei do not behave independently of each other, since their own magnetic fields affect the energy levels of surrounding nuclei. The energy levels are increased by $\frac{\pi}{2}J_{12}\hbar$ if the nuclei are oriented in the same direction, and decreased by the same amount if they are oriented in the opposite direction. The Hamiltonian thus gets shifted to $\frac{\omega_1\hbar}{2}\sigma_z\otimes\mathbb{I}+\frac{\omega_2\hbar}{2}\mathbb{I}\otimes\sigma_z+\frac{\pi\hbar}{2}J_{12}\sigma_z\otimes\sigma_z$

$$=\frac{\hbar}{2}\begin{pmatrix} \omega_1+\omega_2+\pi J_{12} & 0 & 0 & 0 \\ 0 & \omega_1-\omega_2-\pi J_{12} & 0 & 0 \\ 0 & 0 & -\omega_1+\omega_2-\pi J_{12} & 0 \\ 0 & 0 & 0 & -\omega_1-\omega_2+\pi J_{12} \end{pmatrix}$$

(see figure 1.9). A radio-frequency pulse of frequency $\frac{1}{2\pi}(\omega_2+\pi J_{12})$ of appropriate intensity and duration would effect the NOT on the second nucleus if and only if the first nucleus is in the state $|1\rangle$. The coupling of the two nuclei thus gives us the opportunity to effect conditional dynamics on the two qubits.

We have just illustrated one way of realising the controlled-NOT. We can use similar pulse sequences to implement quantum algorithms using NMR technology.

I have said how quantum operations are unitary (some operations may seem non-unitary, but only when ignoring part of the entire system), and have even illustrated some examples of unitary operations, such as putting a beam-splitter in the path of a photon, or applying a radio-frequency pulse to a nucleus. The Schrödinger equation tells us how these black-boxes work.

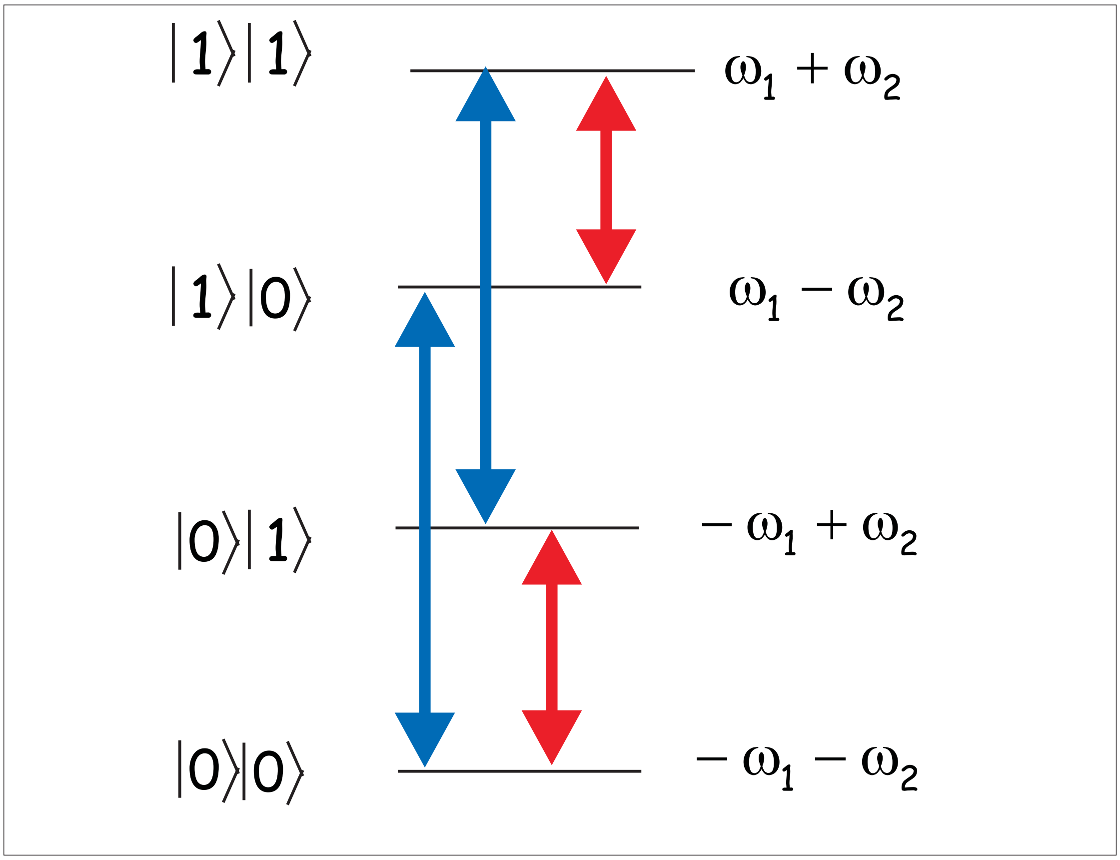

FIGURE 1.8. With two independent nuclei, we have four energy eigenstates. A radio-frequency pulse of frequency ω_1 and appropriate duration and intensity will effect a *NOT* on the first qubit. A radio-frequency pulse of frequency ω_2 and appropriate duration and intensity will effect a *NOT* on the second qubit.

Consider a closed system in the state $|\Psi(0)\rangle \in \mathbb{H}_N$. This state can change over time, and we denote by $|\Psi(t)\rangle$ the state of this system at time t. Let us suppose that we start with a system in the state $|\Psi(0)\rangle$ and the system is in an environment where the energy of a state $|\Psi\rangle$ is defined by the operator $H(t)$. The

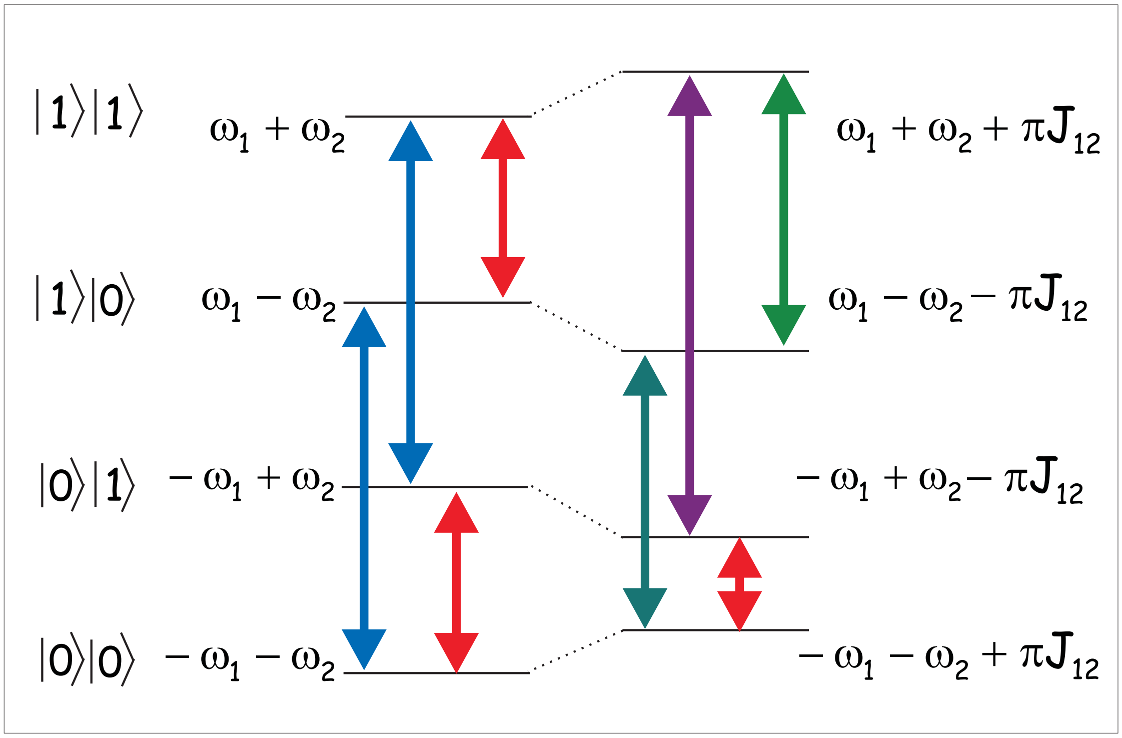

FIGURE 1.9. The energy eigenvalues of two nuclei with respective energies $\pm\frac{\omega_1\hbar}{2}$ and $\pm\frac{\omega_2\hbar}{2}$ and coupling term J_{12} are illustrated. A radio-frequency pulse of frequency $\frac{\omega_2}{2\pi}+\frac{J_{12}}{2}$ of appropriate intensity and duration will effect the controlled-NOT.

Schrödinger equation states

$$i\hbar\frac{d}{dt}\,|\,\Psi(t)\rangle = H(t)\,|\,\Psi(t)\rangle\,.$$

This equation implies that the evolution of the state $|\,\Psi\rangle$ is unitary. If we assume the Hamiltonian remains constant then the solution to the Schrödinger equation is

$$|\,\Psi(t)\rangle = e^{\frac{-i}{\hbar}Ht}\,|\,\Psi(0)\rangle\,.$$

This equation implies that the evolution of the state $|\,\Psi\rangle$ is unitary. For example, if we change the environment of a qubit so that we effect a Hamiltonian $\omega\hbar\mathbb{I}_x$, where

$$\mathbb{I}_x = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} = \frac{1}{2}\sigma_x,$$

($\mathbb{I}_x$ is typically used in chemistry) then by letting the qubit evolve in this environment for a period of time t we effect the unitary operation

$$\begin{pmatrix} \cos(\frac{\omega}{2}t) & -i\sin(\frac{\omega}{2}t) \\ -i\sin(\frac{\omega}{2}t) & \cos(\frac{\omega}{2}t) \end{pmatrix}.$$

So if we set $t = \frac{\pi}{\omega}$ we get a unitary operation equal to

$$e^{-i\pi\mathbb{I}_x} = -i\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

which corresponds to a NOT gate (apart from the global phase factor [3] of $-i$). This is often called a π-pulse, or a $180°\mathbb{I}_x$ pulse. If $t = \frac{\pi}{2\omega}$, we call this a $\frac{\pi}{2}$-pulse or a $90°\mathbb{I}_x$ pulse, which realises the square root of NOT operation.

A $180°\mathbb{I}_y$ pulse, where

$$\mathbb{I}_y = \begin{pmatrix} 0 & -\frac{i}{2} \\ \frac{i}{2} & 0 \end{pmatrix} = \frac{1}{2}\sigma_y,$$

realises the operation

$$e^{-i\pi\mathbb{I}_y} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

[3] *Global* phase shifts, say by a factor $e^{i\phi}$, of a state $|\,\Psi\rangle$ do not affect the probability of observing any particular state $|\,x\rangle$ even if the state $|\,\Psi\rangle$ is transformed by unitary operator U.

and applying a $90°\mathbb{I}_y$ pulse realises what we call a *pseudo-Hadamard* gate [**JM98**]:

$$e^{-i\frac{\pi}{2}\mathbb{I}_y} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}. \tag{7}$$

One- and two-qubit gates like these will form the basis of a quantum computer, which we describe in the next section.

This brief summary of quantum physics should suffice for this course. For more details, see for example [**Fey65, CTDL77**].

1.2. Quantum Computers

Our model of computation will be a quantum version of the uniform families of acyclic circuits, namely uniform families of *quantum networks* or *acyclic quantum gate arrays*. We will restrict attentions to finite *universal* sets $\mathbb{G}$ of gates that can efficiently simulate any other finite set of gates. Let us explain this in more detail.

A set $\mathbb{G}$ of gates is universal if for every positive integer n, any real $\epsilon > 0$, and any unitary operation U on n qubits, we can approximate U with an error of at most ϵ using gates from $\mathbb{G}$. This means that there is a network $\mathcal{N}$ of gates from $\mathbb{G}$ such that for any input $|\,x\rangle$, the network outputs a state $\mathcal{N}\,|\,x\rangle$ that is within distance ϵ (in the Euclidean norm) of the state $U\,|\,x\rangle$.

An example of a universal (but not finite) set of gates that can *exactly* simulate any unitary operation is the set of single-qubit operations and the controlled-NOT gate [**BBC$^+$95**]. It is not reasonable of course to have an infinite number of gates at our disposal, but the controlled-NOT with almost any one-qubit gate allows us to approximate any unitary operator arbitrarily well. Other finite families of universal gates are also known [**Deu89, BDEJ95, DiV95, Llo95**].

We also require our universal set $\mathbb{G}$ of gates to be able to efficiently simulate any network $\mathcal{N}$ of T gates from another specified set of gates. This means that for any $\epsilon > 0$, we can efficiently construct a network $\mathcal{N}'$ of at most $poly(T, \frac{1}{\epsilon})$ gates from $\mathbb{G}$ that approximates $\mathcal{N}$ with accuracy ϵ. We therefore require our universal set $\mathbb{G}$ of gates to have the property that given a gate G (acting on a finite number of qubits) and an integer $N > 0$, we can efficiently find a sequence of $poly(N)$ gates in $\mathbb{G}$ that approximates G with accuracy $\frac{1}{N}$. Many such families are known [**BV97, Cle99, Kit97, ADH97**]. Kitaev [**Kit97**] and Solovay [**Sol99**] define sets of gates that have only a $polylog(N)$ overhead.

As in the classical case, we also require uniformity. In this course it will be clear from the description of the quantum networks that a classical Turing machine (either a reversible or an irreversible one) can easily construct them. For example, one could easily write a short C program that will efficiently output circuit diagrams for the algorithms described. Some of the algorithms involve constructing a network, executing it, and then depending on the output, we construct and execute another quantum network if necessary. Another natural model of computation is the universal quantum Turing machine [**Deu85**]. See [**BV97, Yao93**] for a discussion of the relationship between these models of computation. We will stick with the uniform families of networks, because they most accurately represent what researchers are attempting to construct in practice and is the model in which most quantum algorithms have been designed.

How are quantum computers different from classical reversible ones? The states of classical reversible computers are confined to one of the computational basis states. Quantum computers can branch out into superpositions of exponentially many computational basis states such as $\sum_{x=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |\, x\rangle$. Once in such a superposition, they are able to compute functions in parallel on all of the basis states. That is, by computing a reversible function σ *only once*, via a quantum gate array that maps $|\, x\rangle \rightarrow |\, \sigma(x)\rangle$, we will map the state $\sum_{x=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |\, x\rangle$ to the state $\sum_{x=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |\, \sigma(x)\rangle$.

The function σ could for example map $|\,x_1\rangle\,|\,x_2\rangle \rightarrow |\,x_1\rangle\,|\,x_2 \oplus f(x_1)\rangle$ for any function f (even a non-reversible f will work since we keep a copy of the input). With exponentially many outputs of σ in superposition, we can probe some of the global properties of σ (that is properties depending on many of its inputs) by techniques we describe later.

Are quantum computers essentially different from classical probabilistic computers? Probabilistic algorithms can also effectively branch over exponentially many computational paths, but there is a subtle difference in how the various outcomes of different paths combine to produce an output. In probabilistic computing, if outcome 0 can be reached by two different paths with probabilities $p_{00} = |a_{00}|^2$ and $p_{10} = |a_{10}|^2$, then the probability of obtaining outcome 0 is $p_{00} + p_{10} = |a_{00}|^2 + |a_{10}|^2$. In a quantum algorithm, if outcome 0 can be obtained by two different paths, like the $|\,0\rangle$ detection in the Mach-Zehnder interferometer, with probability amplitudes a_{00} and a_{10}, then the probability of observing 0 is $|a_{00} + a_{10}|^2$. The relative phase between the two amplitudes is important and can result in either *constructive* or *destructive* interference. Destructive interference occurs when the amplitudes a_{00} and a_{10} cancel each other out, the most extreme case being when $a_{00} = -a_{10}$ and we thus measure 0 with probability 0. Constructive interference occurs when the amplitudes are aligned in some common direction, an extreme case being when $a_{00} = a_{10} = \frac{1}{\sqrt{2}}$ in which case we measure 0 with probability 1. The challenge is to design quantum algorithms which induce constructive interference on *good* states and destructive interference on *bad* states.

CHAPTER 2

Algorithms

This chapter will discuss the algorithms which make explicit use of the quantum nature of a quantum computer. I summarise the known quantum algorithms and present them in a simple and unified way. The simplest quantum algorithm is presented first. I then generalise the basic elements of this algorithm and show how they lead to the known powerful quantum algorithms.

2.1. The Deutsch algorithm

In [**Deu85**], Deutsch considers the task of determining $f(0) \oplus f(1)$ for some Boolean function $f : \{0,1\} \rightarrow \{0,1\}$, where $\oplus$ corresponds to addition modulo 2. However, he supposes that we only have the resources to implement f once. Note that $f(0) \oplus f(1) = 0$ if and only if f is constant. If we restrict ourselves to a 'classical' setting, it is impossible to determine $f(0) \oplus f(1)$ using only one evaluation of f. In a quantum setting however, given a reversible means of computing f, we *can* perform this task! Deutsch presented an algorithm which outputs the correct answer with probability $\frac{1}{2}$ and outputs an inconclusive result otherwise. By a subtle modification [**CEMM98**], as we show here, we *can* compute the correct answer with probability 1 using only one application of f.

We are given a quantum means of computing f, namely, we have a unitary operator U_f which implements:

$$|\,x\rangle\,|\,b\rangle \rightarrow |\,x\rangle\,|\,b \oplus f(x)\rangle \tag{8}$$

where $x, b \in \{0,1\}$. The 'trick' used in [**CEMM98**] is to the encode the value of $f(x)$ into the phase. To do this we start with a second, *auxiliary* register in the state $|\,0\rangle - |\,1\rangle$ (remember that we will often drop the normalisation factors), then the operation $|\,b\rangle \to |\,b \oplus f(x)\rangle$ maps $|\,0\rangle - |\,1\rangle$ to

$$|\,f(x)\rangle - |\,1 \oplus f(x)\rangle = (-1)^{f(x)}(|\,0\rangle - |\,1\rangle).$$

Consequently we can describe U_f in a different basis as

$$\begin{aligned} |\,x\rangle\,(|\,0\rangle - |\,1\rangle) &\to (-1)^{f(x)}\,|\,x\rangle\,(|\,0\rangle - |\,1\rangle) \\ |\,x\rangle\,(|\,0\rangle + |\,1\rangle) &\to |\,x\rangle\,(|\,0\rangle + |\,1\rangle) \end{aligned}$$

for $x \in \{0,1\}$. If we fix the auxiliary register to be in state $|\,0\rangle - |\,1\rangle$, then its state remains unchanged and we effectively have a way of mapping $|\,x\rangle \to (-1)^{f(x)}\,|\,x\rangle$.

Let us go back to the example of the Mach-Zehnder interferometer. There we saw that if the relative phase-shift between the two computational paths was either 0 or π, then we could correctly distinguish the two cases 100% of the time. The same 'algorithm' can be applied to solve Deutsch's problem, as illustrated in figure 2.1. We will now ignore the state of the auxiliary qubit, since it stays

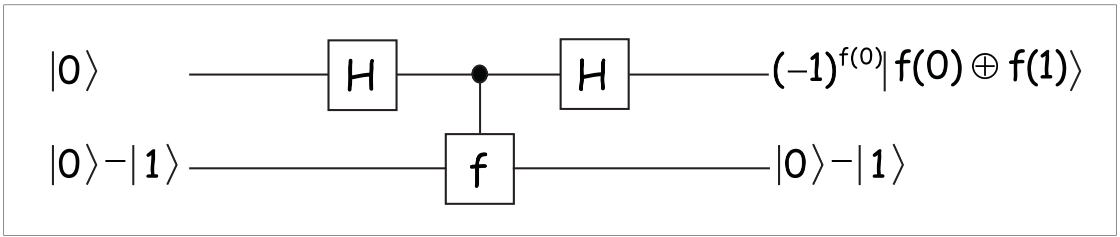

FIGURE 2.1. Network representation of the Deutsch algorithm

in the state $|\,0\rangle - |\,1\rangle$. The first Hadamard gate H creates the state $|\,0\rangle + |\,1\rangle$. The U_f induces phase shifts to produce $(-1)^{f(0)}\,|\,0\rangle + (-1)^{f(1)}\,|\,1\rangle$, and the final H

completes the interference that produces the state $(-1)^{f(0)}\,|\,f(0)\oplus f(1)\rangle$, in other words, *the answer*.

As we will see in the subsequent sections, this simple algorithm illustrates all the main ingredients of the known quantum algorithms.

2.2. Eigenvalue Kick-Back

One of the key steps in the quantum solution to Deutsch's problem was encoding the value of $f(x)$ into the phases given a quantum version of a reversible algorithm for computing f, namely, an operation U_f that maps $|\,x\rangle\,|\,b\rangle \to |\,x\rangle\,|\,b\oplus f(x)\rangle$. Here $\oplus$ refers to an appropriate group addition, such as addition modulo 2 if $b \in \{0,1\}$, or addition in Z_2^n if $b \in \{0,1\}^n$, or addition modulo N if $b \in \{0,1,\ldots,N-1\}$. *Computing f into the phases*, where $b \in \{0,1\}$ meant sending $|\,x\rangle \to (-1)^{f(x)}\,|\,x\rangle$. More generally, when $f(x) \in \{0,1\}^n$ this means sending $|\,x\rangle \to (-1)^{f(x)\cdot b}\,|\,x\rangle$ for some $b \in \{0,1\}^n$ and when $f(x) \in \{0,1,\ldots,N-1\}$ this means sending $|\,x\rangle \to e^{\frac{2\pi i f(x)}{N}}\,|\,x\rangle$.

In this section we generalise the method of inducing relative phases in the states of the register (which we call the *control* register) that contains the input values $|\,x\rangle$. We make use of an auxiliary or *target* register and a function evaluation.

Suppose we have a family of unitary operators U_x which share an eigenvector $|\,\Psi\rangle$ with eigenvalues $e^{2\pi i\omega_x}$, respectively. The value ω_x corresponds to $\frac{f(x)}{2}$, $\frac{f(x)\cdot b}{2}$, or $\frac{f(x)}{N}$ in the above examples. We define the controlled-U_x to be a unitary operation acting on two systems, a control system of dimension M and a target system of dimension N. The controlled-U_x maps $|\,x\rangle\,|\,y\rangle$ to $|\,x\rangle\,U_x\,|\,y\rangle$. We give an example of such a controlled-U_x, and how to implement it, later in this section. Figure 2.2 shows how to use this operator to compute w_x into the phases. Starting with the state $|\,\Psi\rangle$ in a second target register, and the state $|\,x\rangle$ in a control register,

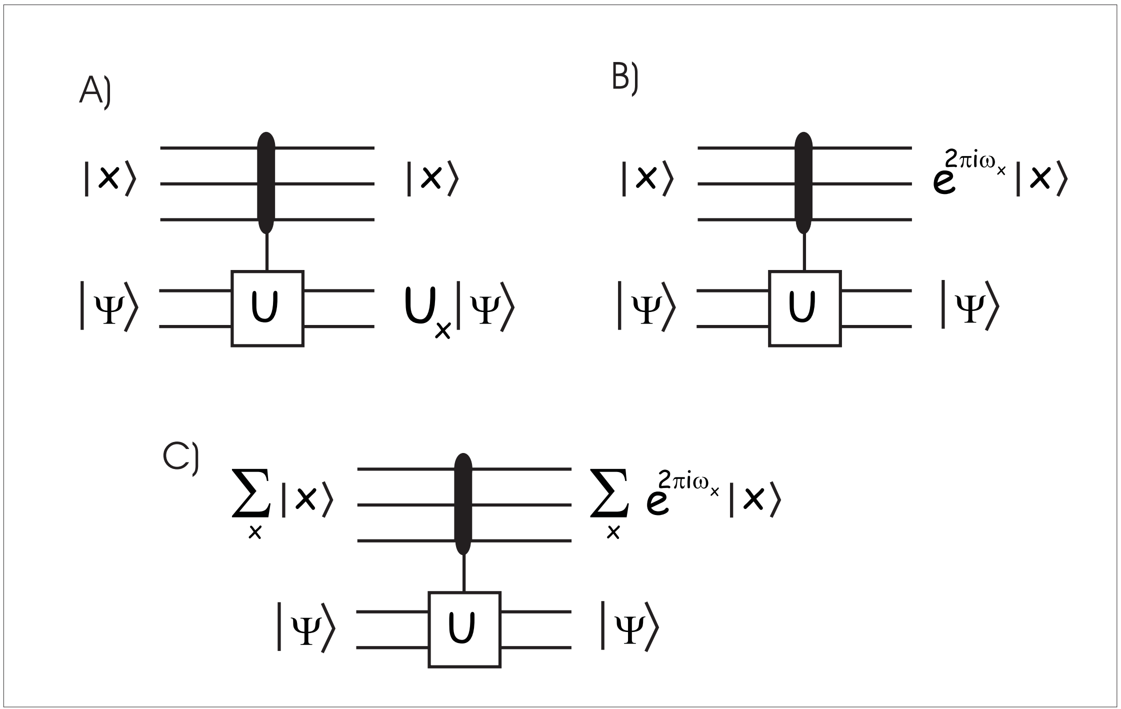

FIGURE 2.2. A) The controlled-U_x applies U_x to the second register when the first register is in state $|x\rangle$. B) The net effect of a controlled-U_x when the target register is in an eigenstate of U_x with eigenvalue $e^{2\pi i\omega_x}$ is to induce a phase factor of $e^{2\pi i\omega_x}$ on the state $|x\rangle|\Psi\rangle$. C) When the target register is in eigenstate $|\Psi\rangle$ and the control register is in any superposition of computational basis states, we can write the outcome as a product of the control register with all the appropriate phase factors kicked back and the target register in the original eigenstate.

apply the controlled-U_x operator. The net effect is to map the state $|x\rangle|\Psi\rangle$ to the state $e^{2\pi i\omega_x}|x\rangle|\Psi\rangle$; the phase $e^{2\pi i\omega_x}$ is associated to the product of the two states. If the second register is in the eigenstate $|\Psi\rangle$, and the first register is in a superposition, say $\sum_{x=0}^{M-1}|x\rangle$, applying a controlled-$U_x$ will produce the product state

$\sum_{x=0}^{M-1} e^{2\pi i \omega_x} |x\rangle |\Psi\rangle$, which can be nicely written as $\left(\sum_{x=0}^{M-1} e^{2\pi i \omega_x} \,|\, x \rangle \right) |\, \Psi \rangle$. In other words, the eigenvalues were "kicked back" into the first register. This might seem wrong since the operators U_x were applied to the second register and we 'only' used the first register as a control. However the first register behaves as an unaffected "control" register only when the controlled-U_x is considered in the computational basis. The controlled-U_x acts linearly on all superpositions of these basis states and has eigenstates $|\, x \rangle \,|\, \Psi \rangle$ with eigenvalues $e^{2\pi i \omega_x}$.

An important example of a controlled-U_x is the operator $\Lambda_M(U)$ which we now define.

DEFINITION 1. *Let U be any unitary operator on a vector space $\mathbb{H}_N$ of dimension N. The operator $\Lambda_M(U)$ acts on the space $\mathbb{H}_M \otimes \mathbb{H}_N$ and sends $|\, x \rangle \,|\, y \rangle \rightarrow |\, x \rangle \, U^x \,|\, y \rangle$.*

Given a quantum network for computing U that uses T elementary quantum gates, we can create a quantum network for $\Lambda_M(U)$ that has $O(MT)$ elementary quantum gates (see appendix A.2). If $U^2, \ldots, U^{2^l}$ are also easily expressible using only T elementary gates, then the network for $\Lambda_M(U)$ has only $O(T \log M)$ elementary gates.

Note that the operator $\Lambda_M(U)$ applied to two registers in the state

$$\sum_{x=0}^{M-1} |\, x \rangle \,|\, \Psi \rangle \,,$$

where $U \,|\, \Psi \rangle = e^{2\pi i \omega} \,|\, \Psi \rangle$, encodes the value ω in the state

$$\sum_{x=0}^{M-1} e^{2\pi i x \omega} \,|\, x \rangle \,|\, \Psi \rangle \,.$$

The next section describes how estimating this phase shift is related to the quantum Fourier transform.

2.3. Phase Estimation and the Quantum Fourier Transform

One of the important features that distinguishes quantum computation from classical randomised computation is the relative phase between states. Estimating such a phase turns out to lie in the kernel of most quantum algorithms. Therefore in this section we will describe some techniques for estimating parameters encoded in the phase of quantum states.

We start with a simple phase estimation task whose natural solution will directly lead to the quantum Fourier transform methods. Suppose, for some real ω, $0 \leq \omega < 1$, that we have the product state

$$(|0\rangle + e^{2\pi i(4\omega)}\,|1\rangle)(|0\rangle + e^{2\pi i(2\omega)}\,|1\rangle)(|0\rangle + e^{2\pi i\omega}\,|1\rangle).$$

Let us (for now) conveniently assume that ω is of the form $\frac{a}{8}$, for some integer a.

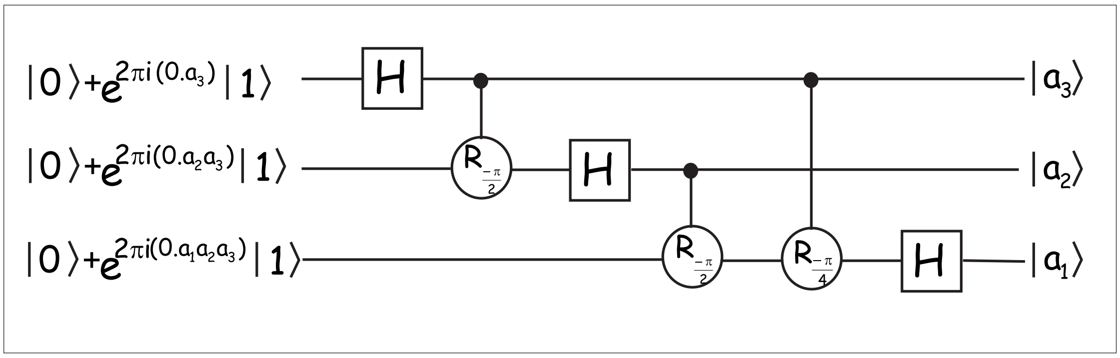

FIGURE 2.3. This phase estimation network implements the inverse quantum Fourier transform (reversing the order of the output qubits).

Let $a_1a_2a_3$ be the binary expansion of a, that is $a = 4a_1 + 2a_2 + a_3$, and each a_j is either 0 or 1. Using the fact that $e^{2\pi i} = 1$, we can also express this state as

$$(|0\rangle + e^{2\pi i(0.a_3)}\,|1\rangle)(|0\rangle + e^{2\pi i(0.a_2a_3)}\,|1\rangle)(|0\rangle + e^{2\pi i(0.a_1a_2a_3)}\,|1\rangle).$$

We know immediately how to obtain the value of a_3 since the Hadamard gate H defined in the introduction maps, for $b \in \{0,1\}$,

$$|\, b\rangle \to |0\rangle + (-1)^b\,|1\rangle = |0\rangle + e^{2\pi i(0.b)}\,|1\rangle\,,$$

and vice versa since $H^{-1} = H$. Thus applying the Hadamard gate to the leftmost qubit gives us the state

$$|\, a_3\rangle\,(|0\rangle + e^{2\pi i(0.a_2a_3)}\,|1\rangle)(|0\rangle + e^{2\pi i(0.a_1a_2a_3)}\,|1\rangle).$$

It would be very convenient if the second qubit were in the state $|0\rangle + e^{2\pi i(0.a_2)}\,|1\rangle$ since we could then easily determine a_2. Fortunately, since the first qubit is in the state $|a_3\rangle$, we can simply apply a controlled-$R_{-\frac{\pi}{2}}$ gate (recall equation (6)) to the leftmost and middle qubits to give us this state. Applying this gate and then a Hadamard gate on the second qubit gives us

$$|\, a_3\rangle\,|\, a_2\rangle\,(|0\rangle + e^{2\pi i(0.a_1a_2a_3)}\,|1\rangle).$$

A controlled-$R_{-\frac{\pi}{2}}$ gate and a controlled-$R_{-\frac{\pi}{4}}$ gate, as shown in figure 2.3, gives us the state

$$|\, a_3\rangle\,|\, a_2\rangle\,(|0\rangle + e^{2\pi i(0.a_1)}\,|1\rangle)$$

which after applying a Hadamard gate becomes

$$|\, a_3\rangle\,|\, a_2\rangle\,|\, a_1\rangle\,.$$

A straightforward generalisation of this network maps the n qubits

$$(|0\rangle + e^{2\pi i(2^{n-1}\omega)}\,|1\rangle)(|0\rangle + e^{2\pi i(2^{n-2}\omega)}\,|1\rangle)...(|0\rangle + e^{2\pi i(2\omega)}\,|1\rangle)(|0\rangle + e^{2\pi i\omega}\,|1\rangle), \tag{9}$$

with $\omega = 0.a_1a_2\ldots a_n$, to the state

$$|a_n\rangle\,|a_{n-1}\rangle\,...\,|a_2\rangle\,|a_1\rangle\,.$$

This network might seem familiar since state (9), is in fact equal to

$$\sum_{x=0}^{2^n-1} e^{2\pi i x\omega} |x\rangle, \tag{10}$$

which for $\omega = \frac{0}{2^n}, \frac{1}{2^n}, ..., \frac{2^n-1}{2^n}$ is a basis state in what we call the *Fourier basis.* Thus the network we described above, apart from a reversing of the final qubits, realises the inverse quantum Fourier transform.

DEFINITION 2. *(QFT(M)) For any integer $M > 1$, the quantum Fourier transform, $QFT(M)$, acts on the vector space generated by the states $|0\rangle, |1\rangle, |2\rangle, ..., |M-1\rangle$, and maps $|j\rangle$ to $\sum\limits_{x=0}^{M-1} e^{2\pi i \frac{j}{M} x} |x\rangle$.*

We just illustrated how to efficiently implement the quantum Fourier transform for $M = 2^n$ (first done in [**Cop94**]), and it is easy to generalise these methods for any M whose prime factors are bounded above by a polynomial in n (done for distinct primes in [**Sho94**] and generally in [**Cle94**]; see appendix A.4). For general M, say a large prime, we can also efficiently approximate $QFT(M)$ by the technique of Kitaev [**Kit95**] but for all applications mentioned in this course, an appropriate power of 2 suffices.

Given the state $\sum\limits_{k=0}^{M-1} e^{2\pi i k\omega} |k\rangle$, where M is known and ω is of the form $\frac{j}{M}$ for some integer j, applying $QFT(M)^{-1}$ will yield the state $|j\rangle$ and thus correctly determine $\omega = \frac{j}{M}$. This is an immediate consequence of the definition of the quantum Fourier transform. What happens if we apply the same estimation technique for arbitrary ω?

DEFINITION 3. *For any real ω, define $\widetilde{|\,\omega\rangle}_M = QFT(M)^{-1}\left(\sum\limits_{k=0}^{M-1} e^{2\pi i k\omega}\,|\,k\rangle\right)$.*

When the value of M is implicit we will just use $\widetilde{|\,\omega\rangle}$.

LEMMA 4. *For all positive integers M,*

- *if* $\omega = \frac{j}{M}, j = 0, 1, \ldots, M-1$, *we have* $\widetilde{|\,\omega\rangle}_M = |\,j\rangle$ *and*
- *for all other real* ω, $0 < \omega < 1$, $\widetilde{|\,\omega\rangle}_M = \sum_{j=0}^{M-1} \alpha_{j,\omega}\,|\,j\rangle$, *where* $|\alpha_{j,\omega}| = \left|\frac{\sin(\pi(M\omega - j))}{M\sin(\pi(\omega - \frac{j}{M}))}\right|$.

PROOF. We have that

$$|\alpha_{j,\omega}| = \frac{1}{\sqrt{M}}\left|\langle\, j|\, QFT^{-1}(M) \sum_{k=0}^{M-1} e^{2\pi i\omega k}\,|\,k\rangle\right| \tag{11}$$

$$= \frac{1}{M}\left|\sum_{k=0}^{M-1} e^{2\pi i k(\omega - \frac{j}{M})}\right| \tag{12}$$

$$= \frac{1}{M}\left|\frac{1 - e^{2\pi i M(\omega - \frac{j}{M})}}{1 - e^{2\pi i(\omega - \frac{j}{M})}}\right| \tag{13}$$

$$= \left|\frac{\sin(\pi(M\omega - j))}{M\sin(\pi(\omega - \frac{j}{M}))}\right|. \tag{14}$$

□

We have the following corollaries describing the quality of this estimator. As we can see in Lemma 4, what matters is $\left|\sin(\pi(\frac{j}{M} - \omega))\right|$ and thus the relevant distance between our estimate $\frac{j}{M}$ and ω is not simply $\left|\frac{j}{M} - \omega\right|$. For example $\frac{1}{M}$ is just as good an estimate of $\omega = 0$ as $\frac{M-1}{M}$.

DEFINITION 5. *The distance* $d(\omega_0, \omega_1)$ *between two real numbers* ω_0 *and* ω_1 *is the real number* d *such that the shortest arclength of the unit circle between* $e^{2\pi i\omega_0}$ *and* $e^{2\pi i\omega_1}$ *is* $2\pi d$.

In other words $d(\omega_0, \omega_1) = \min_{z\in\mathbf{Z}}\{|\omega_0 - \omega_1 + z|\}$. Roughly speaking, the probability of getting an error of size Δ is proportional to $\frac{1}{\Delta^2}$.

COROLLARY 6. *Given the integer* $M > 0$ *and the state* $\sum_{j=0}^{M-1} e^{2\pi i j\omega}\,|\,j\rangle$, *applying* $QFT(M)^{-1}$ *and then measuring will yield the state* $|\,j\rangle$ *satisfying the following. Let* $\Delta = d(\omega, \frac{j}{M})$.

- – *If $M\omega$ is an integer, then with probability* 1, $j = M\omega$, *that is* $\Delta = 0$.
  – *Otherwise, the probability of observing* $|\,j\rangle$ *is* $\frac{\sin^2(M\Delta\pi)}{M^2\sin^2(\Delta\pi)} \leq \frac{1}{(2M\Delta)^2}$.
- *With probability at least* $\frac{8}{\pi^2}$ *we have* $\Delta \leq \frac{1}{M}$.
- *For* $k > 1$, *with probability at least* $1 - \frac{1}{2(k-1)}$ *we have* $\Delta \leq \frac{k}{M}$.

PROOF. We make use of the fact that for x between 0 and $\frac{1}{2}$, $2x \leq \sin(\pi x) \leq \pi x$. From Lemma 4 the probability that we measure j is $\left|\frac{\sin(\pi(M\omega - j))}{M\sin(\pi(\omega - \frac{j}{M}))}\right|^2 \leq \left(\frac{1}{M\sin(\pi\Delta)}\right)^2 \leq \left(\frac{1}{2M\Delta}\right)^2$. We use this fact to prove the rest of the theorem.

$$
\begin{aligned}
\operatorname{Prob}\left(|j - M\omega| \leq k\right) &= 1 - \operatorname{Prob}(|j - M\omega| > k) \\
&\geq 1 - 2\sum_{j=k}^{\infty} \frac{1}{4M^2(\frac{j}{M})^2} \\
&\geq 1 - \frac{1}{2(k-1)}.
\end{aligned}
$$

The last inequality is easily proved by bounding the summation by an appropriate integral.

We now use the fact that for $M > 2$, the function

$$
\frac{\sin^2(x)}{M^2}\left(\frac{1}{\sin^2(x/M)} + \frac{1}{\sin^2((\pi - x)/M)}\right),
$$

$0 \leq x \leq \pi$, attains its minimum at $x = \frac{\pi}{2}$

$$
\begin{aligned}
\operatorname{Prob}\left(|j - M\omega| \leq 1\right) &= \operatorname{Prob}(j = \lfloor M\omega \rfloor) + \operatorname{Prob}(j = \lceil M\omega \rceil) \\
&= \frac{\sin^2(M\Delta\pi)}{M^2\sin^2(\Delta\pi)} + \frac{\sin^2(M(\frac{1}{M} - \Delta)\pi)}{M^2\sin^2((\frac{1}{M} - \Delta)\pi)} \\
&\geq \frac{8}{\pi^2}.
\end{aligned}
$$

□

We have described and analysed techniques for estimating phases encoded in the state $\sum_{x=0}^{M-1} e^{2\pi i x\omega}\,|\,x\rangle$. This is not the only way to encode phases, but it is simple and elegant. In [**vDDE+99**] it is shown how encoding ω in a state of the form $\sum_{x=0}^{M-1} \alpha_x e^{2\pi i x\omega} |\, x\rangle$ for appropriate α_x and then applying $QFT^{-1}(M)$ can provide a 'better' estimate of ω, where the α_x depend on the definition of 'better'. We will not discuss these methods here since the techniques we have described above suffice for the algorithms presented in this course. It would be slightly misleading not to note that the QFT as described earlier is not practical since it uses exponentially precise phase rotations. For the purpose of phase estimation and for the algorithms described herein, the transformations described above can be approximated sufficiently well with phase shifts of bounded size as described in [**Cop94**] and [**BEST96**]. By using different encodings of ω, Kitaev [**Kit95**] shows how to estimate ω without any quantum controlled rotations. I also noted that it is possible to combine the phase estimation techniques used by Kitaev [**Kit95**] with the approximate quantum Fourier transform methods. Some work in this direction has been carried out [**Bhi98**] and these techniques show promise of being practical and more efficient than those in [**Kit95, Cop94, BEST96**]. We will not discuss these methods in this course, but simply point out that they exist, so that exponential precision in the full QFT is not necessary. We use the QFT methods for their simplicity.

With these last two sections under our belt, the content of the next section might now seem obvious, yet it is the essence of the powerful quantum algorithms such as those for factoring and finding discrete logarithms.

2.4. Quantum Eigenvalue Estimation

The previous two sections have described the two main ingredients for estimating an eigenvalue of a unitary operator U. The algorithm, whose network is illustrated in figure 2.4, uses a control register and a target register. We initialise the control register to the state $|\, 00\ldots0\rangle$. The target register can be in any state, but we analyse the algorithm in the basis of eigenvectors of U. Let $|\Psi_k\rangle, k = 0, 1, \ldots, N-1$, be the eigenvectors of U with respective eigenvalues $e^{2\pi i \omega_k}$. We will make use of the operator $\Lambda_M(U)$ from definition 1.

It is easy to see that $\Lambda_M(U)\,|x\rangle\,|\Psi_k\rangle = e^{2\pi i x \omega_k}\,|x\rangle\,|\Psi_k\rangle$. Combining this phase kick-back with the phase estimation method described earlier, we have the following algorithm.

ALGORITHM 7. *(Eig_Est(U, ρ, M))*

Input:

- *An integer N.*
- *A quantum network for a unitary operator U which acts on a vector space $\mathbb{H}_N$ of dimension N.*
- *A positive integer M.*
- *A target register in an arbitrary state described by the density matrix ρ. [Note that the input is the quantum state itself not a classical description of it.]*

Output:

- *A rational $\tilde{\omega}$ (i.e. an integer x such that $\tilde{\omega} = \frac{x}{M}$).*

Complexity:

- *1 application of $\Lambda_M(U)$.*
- *$O(\log^2 M)$ other elementary operations.*

Procedure:

1. *Initialise a control register to the state $|0\rangle \in \mathbb{H}_M$.*
2. *Apply $QFT(M)$ to the control register.*
3. *Apply $\Lambda_M(U)$.*

4. *Apply* $QFT(M)^{-1}$ *to the control register.*
5. *Measure the control register, denote the outcome* x*. Output* $\tilde{\omega} = \frac{x}{M}$.

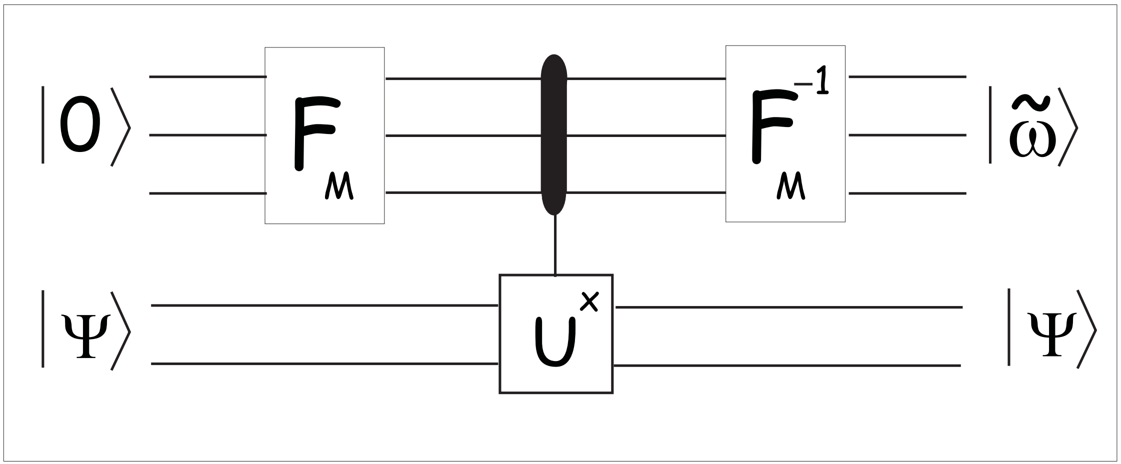

FIGURE 2.4. We illustrate the effect of the algorithm *Eig_Est* with the target register in an eigenvector of U and the control register in the specified starting state.

THEOREM 8. *After running algorithm Eig_Est*(U, ρ, M)*, the control register is in the state*

$$\sum_{k=0}^{N-1} \rho_{k,k} \,|\, \widetilde{\omega_k} \rangle \langle \widetilde{\omega_k} \,|$$

where the target register started in the state $\rho = \sum_{j,k} \rho_{j,k} \,|\, \Psi_j \rangle \langle \Psi_k |$*. Thus if we measure the control register we will get an estimate* $\widetilde{\omega_k}$ *of* ω_k *with probability* $Tr(\rho \,|\, \Psi_k \rangle \langle \Psi_k |) = \rho_{k,k}$*. The probability distribution of* $\widetilde{\omega_k}$ *is equivalent to that of measuring* $|\, \widetilde{\omega_k} \rangle$*, as described in corollary 6, and outputting the result* x *divided by* M*.*

More simply, if the target register contains the state $|\, \Psi_j \rangle$ with probability $\rho_{j,j}$, then we will measure $|\, \widetilde{\omega_j} \rangle$ with probability $\rho_{j,j}$.

PROOF. At step 1 the density matrix of the two-system state is

$$|\,0\rangle\,\langle\,0|\otimes\rho=\sum_{j,k}\rho_{j,k}\,|\,0\rangle\,\langle\,0|\otimes|\,\Psi_j\rangle\,\langle\,\Psi_k|\,.$$

The sequence $(QFT(M)^{-1}\times I)\Lambda_M(U)(QFT(M)\times I)$ maps $|\,0\rangle\,|\,\Psi_j\rangle$ to $|\,\widetilde{\omega_j}\rangle\,|\,\Psi_j\rangle$. Thus the state of the two systems becomes

$$\sum_{j,k}\rho_{j,k}\,|\,\widetilde{\omega_j}\rangle\,\langle\,\widetilde{\omega_k}|\otimes|\,\Psi_j\rangle\,\langle\,\Psi_k|\,.$$

Tracing out the second register gives us the state

$$\sum_{j}\rho_{j,j}\,|\,\widetilde{\omega_j}\rangle\,\langle\,\widetilde{\omega_j}|\,.$$

The result follows.

$\square$

We described the above algorithm in the most general setting where we were given any input state ρ. In this chapter we only use pure states.

COROLLARY 9. *Let the density operator ρ describe the state*

$$\sum_{k=0}^{r-1}\frac{1}{\sqrt{r}}\,|\,\Psi_k\rangle\,.$$

Algorithm ($Eig\_Est(U,\rho,M)$) outputs an estimate $\widetilde{\omega_k}$ of ω_k with probability $\frac{1}{r}$. The probability distribution of the estimate $\widetilde{\omega_k}$ is equivalent to that of measuring $|\,\widetilde{\omega_k}\rangle$ and outputting the result x divided by M.

2.5. Finding orders

We are now ready to describe the most well-known application of the above techniques. The task is a 'classical' one:

Let G be any finite group for which we know how to compute the group operation and for which we have a unique binary representation for every element and an algorithm for computing the group operation in that representation. We will represent the operation multiplicatively.

PROBLEM 10. *Given an element a from a finite group G, find the order of a, that is the smallest positive integer r such that $a^r = 1$.*

This problem is not only of theoretical interest, but it is of great practical interest. The reason is that integer factorisation and cracking the RSA cryptosystem reduce to this problem for $G = \mathbb{Z}_N^*$, the multiplicative group integers modulo N (see appendix A.5.3).

The quantum version we will address is the following. Let G be any group for which we have unique binary representatives for each element and an algorithm for computing the group operation and for computing inverses.

PROBLEM 11. *Given an element a of a finite group G, denote by U_a the unitary operator that maps, for all $x \in G$, $|\,x\rangle \to |\,ax\rangle$. Find the order of U_a.*

Before we describe the solution to this problem, we will make a few remarks regarding the statement of these problems. First note that to turn Problem 10 into Problem 11, it suffices to know a^{-1}, since this means we have an efficient reversible way of implementing U_a (see appendix A.3). For groups where a^{-1} is not easily obtained, we can apply the period-finding technique discussed later to find the order r of a (note that we can then simply compute $a^{-1} = a^{r-1}$). We need unique representatives for each element of G since this is necessary for quantum interference to occur. Most groups of interest have some easy-to-compute canonical representatives. Note that the group G does not need to be Abelian: it suffices that the subgroup generated by a is Abelian, which is always true.

Solving this quantum version is simple once we measure the following. Since $U_a^x = U_{a^x}$, then r is also the order of U_a. Since $U_a^r = I$, the identity operator, then any eigenvector $|\Psi\rangle$ must satisfy $U_a^r |\Psi\rangle = |\Psi\rangle$, which implies that the eigenvalues of U_a are rth roots of unity. For each coset $\{c, ac, a^2c, ..., a^{r-1}c\}$ of $\langle a\rangle$, the subgroup generated by a, there correspond r eigenvectors [**Kit95**]:

$$|\Psi_k^c\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-\frac{2\pi ijk}{r}} \left|ca^j\right\rangle, \text{ for } k = 0, 1, ..., r-1. \tag{15}$$

Then note that

$$U_a |\Psi_k^c\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-\frac{2\pi ijk}{r}} \left|ca^{j+1}\right\rangle = \frac{e^{\frac{2\pi ik}{r}}}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-\frac{2\pi ijk}{r}} \left|ca^j\right\rangle = e^{\frac{2\pi ik}{r}} |\Psi_k^c\rangle .$$

Further,

$$|c\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\Psi_k^c\rangle$$

(that is we get constructive interference on the basis state $|\, ca^0\rangle$ and totally destructive interference on the other $|\, ca^j\rangle$ states).

Estimating such random eigenvalues precisely enough will allow us to find r. This fact is based on the theory of *continued fractions*. Every real number y has a sequence of rationals, called *convergents*, that approximate it. The convergents can be efficiently computed as outlined in [**Kob94**] or [**Knu98**]. The following lemma follows from Theorem 184 of [**HW79**].

LEMMA 12. *Given the integers x and M, if*

$$\left|\frac{k}{r} - \frac{x}{M}\right| \le \frac{1}{M} < \frac{1}{2r^2}$$

then the fraction $\frac{k}{r}$ is a convergent of $\frac{x}{M}$.

It is easy to see that there can be at most one fraction $\frac{a}{b}$ that satisfies $b \le r$ and $\left|\frac{a}{b} - \frac{x}{M}\right| < \frac{1}{2r^2}$. The continued fractions algorithm will find integers a and b such that $\frac{a}{b} = \frac{k}{r}$ after computing at most $O(\log M)$ convergents.

We will use the following algorithm as a subroutine later. This algorithm assumes we have an upper bound on r, whereas this subsequent algorithm does not.

ALGORITHM 13.

Input:

- *An integer M*
- *An element $a \in G$.*

Output:

- *A positive integer t or FAIL.*

Complexity:

- *$O(T \log M)$ elementary operations where a group operation uses $O(T)$ elementary operations.*
- *$O(\log^2 M)$ other elementary operations.*

Procedure:

1. *Repeat Eig_Est$(U_a, |1\rangle\langle 1|, M)$ twice to obtain two estimates $\frac{x_1}{M}$ and $\frac{x_2}{M}$ of random eigenvalues of U_a.*
2. *Use the continued fractions algorithm to seek two fractions $\frac{a_1}{r_1}$ and $\frac{a_2}{r_2}$ with $r_1, r_2 \leq \sqrt{\frac{M}{2}}$ that satisfy*

$$\left|\frac{x_1}{M} - \frac{a_1}{r_1}\right| \leq \frac{1}{M} \tag{16}$$

$$\left|\frac{x_2}{M} - \frac{a_2}{r_2}\right| \leq \frac{1}{M}. \tag{17}$$

   If both do not exist, then return FAIL.

3. *Let t be the lowest common multiple of r_1 and r_2. If this is not less than $\sqrt{\frac{M}{2}}$ then return FAIL.*

4. *If* $a^t \neq 1$ *return FAIL,*

5. *Return* t.

PROPOSITION 14. *If* $M > 2r^2$*, then Algorithm 13 finds the correct order* r *with probability at least* $\frac{32}{\pi^4}$*. If it does not output FAIL, then it outputs a multiple of* r*.*

PROOF. The proposition follows by first showing that with probability at least $\left(\frac{8}{\pi^2}\right)^2$, x_1 and x_2 satisfy the equations (16) and (17) for k_1 and k_2 chosen independently and uniformly at random from the set $\{1, \dots, r\}$. The second part of the proof shows that assuming x_1 and x_2 satisfy these equations for such k_1 and k_2 then we obtain r with probability at least $\frac{1}{2}$.

By Theorem 8 (Eig_Est) we will obtain integers x_1 and x_2 such that for each k_1 and $k_2 \in \{0, 1, \dots, r-1\}$ with probability $\frac{1}{r}$ the value $\frac{x_1}{M}$ is an estimate of $\frac{k_1}{r}$, and similarly for $\frac{x_2}{M}$. Further, for $j = 0, 1$ the distribution of x_j is equivalent to that of measuring $\left|\widetilde{\frac{k_j}{r}}\right\rangle$, as described in corollary 6, and outputting the outcome x divided by M. Since $Eig\_Est(U_a, M, |1\rangle\langle 1|)$ was run twice independently, the integers k_1 and k_2 are independent.

Since the error in the estimate is thus at most $\frac{1}{2r^2}$, then the continued fractions algorithm will efficiently find the fractions $\frac{a_1}{r_1} = \frac{k_1}{r}$ and $\frac{a_2}{r_2} = \frac{k_2}{r}$. We still might not know r since the integers k_1, k_2 and r might not be coprime. Taking the lowest common multiple of the denominators yields $t = \frac{r}{\gcd(k_1, k_2, r)}$. Since k_1 and k_2 are selected uniformly at random between 1 and r, they are coprime with probability at least $1 - \sum_{\text{prime } p} \frac{1}{p^2} > \frac{1}{2}$. Thus this procedure will successfully find r with probability greater than $\frac{32}{\pi^4}$.

The final test that $a^t = 1$ guarantees that the algorithm only outputs FAIL or multiples of r. $\square$

Note that $|1\rangle$ could have been replaced by any mixture of $|a^j\rangle$ states.

We are now ready to solve Problem 11. We will assume the worst case, that is that we have no good bounds on r.

ALGORITHM 15. *($Find\_Order(a)$)*

Input:

- *An element $a \in G$.*

Output:

- *A positive integer t.*

Complexity:

- *Expected $O(T \log r)$ elementary operations where a group operation requires $O(T)$ elementary operations.*
- *Expected $O(\log^2 r)$ other elementary operations.*

Procedure:

1. *Set $M = 2$.*
2. *Apply Algorithm 13 three times.*
3. *If all three outputs are FAIL, double M and go to step 2. Otherwise let t be the minimum value of the non-FAIL outputs.*
4. *Test if $a^t = 1$. If so, output t. Otherwise double M and to go step 2.*

THEOREM 16. *Algorithm 15 ($Find\_Order(a)$) finds a multiple of r (the order of a). With probability at least $\frac{2}{3}$ the multiple is indeed r. The expected complexity is $O(\log r)$ group multiplications and $O(\log^2 r)$ other elementary operations.*

PROOF. While $M \leq 2r^2$, step 3 of Algorithm 13 guarantees that we will obtain FAIL (the outputted $t > 0$ satisfies $t < \sqrt{\frac{M}{2}} \leq r$ and thus cannot satisfy $a^t = 1$). Once $M \geq 2r^2$, then Proposition 14 tells us that we will find the correct r with probability at least $1 - (1 - \frac{32}{\pi^4})^3 > \frac{2}{3}$ at each iteration. To bound the expected running time, note that once $M \geq 2r^2$, the algorithm will output an answer with probability at least $\frac{2}{3}$. This means the expected running time is $O(\log r)$ group multiplications and $O(\log^2 r)$ elementary operations. The final test guarantees any output is a multiple of r. $\square$

We can modify this algorithm so that it always outputs the correct r, by factoring the output t (with the factoring algorithm described in [**Buh96**]) and removing unnecessary factors.

2.6. Discrete Logarithms

Not all public key cryptosystems in use today rely on the difficulty of factoring. Breaking many cryptosystems in use today can be reduced to finding discrete logarithms in groups such as the multiplicative group of finite fields or the additive group of points on certain elliptic curves (see appendix A.6 and [**MvOV97**]). Shor [**Sho94**] also shows how to find discrete logarithms in $GF(p)^*$, and the algorithm easily extends to other groups.

The discrete logarithm problem in a group G is the following. Let G be any finite group for which we have a unique binary representation for every element and an algorithm for computing the group operation. We represent the group multiplicatively.

PROBLEM 17. *Given elements a and $b = a^s$, $0 \leq s < r$ (r is the order of a) from the group G, find s, also referred to as the* discrete logarithm of b with respect to the base a.

We can translate this discrete logarithm problem into a discrete logarithm problem in the group of unitary operators on the vector space spanned by the elements of G. Namely, letting $U_a : |\,x\rangle \to |\,ax\rangle$ and $U_b : |\,x\rangle \to |\,bx\rangle$, we wish to find the logarithm r of U_b to the base U_a. We can also assume that we know a^{-1}, b^{-1} and r, the order of a, due to the order-finding or period-finding algorithms discussed in the previous section. As mentioned in the previous section, given a^{-1} and b^{-1} we can easily construct a reversible network for implementing U_a and U_b. Since we can factor r, we can further assume that r is prime (see section 4.2.1, for details and other advantages).

The operators U_a and U_b share the same eigenvectors $|\Psi_k^c\rangle$ defined in the previous section, with respective eigenvalues $e^{\frac{2\pi ik}{r}}$ and $e^{\frac{2\pi iks}{r}}$. The idea is to apply the eigenvalue estimation algorithm to estimate these two eigenvalues accurately enough to determine both $\frac{k}{r}$ and $\frac{ks \bmod r}{r}$. Since we know r, we only need to estimate these eigenvalues with an error of at most $\frac{1}{2r}$ in order to find the correct numerator. If $k \neq 0$ we can simply compute $s = k^{-1}ks \bmod r$. The eigenvalue estimation algorithm thus gives us the following algorithm for finding discrete logarithms.

Algorithm 18. *(Discrete_Log(b,a))*

Input:

- *Elements a and $b \in G$, where b is a power of a.*
- *The order r of a.*

Output:

- *An integer t or $FAIL$.*

Complexity:

- *$O(\log r)$ group operations.*
- *$O(\log^2 r)$ other elementary operations.*

Procedure:

1. *Prepare three registers in the state*
$$|\,00\ldots0\rangle\,|\,00\ldots0\rangle\,|\,1\rangle\,.$$
2. *Let* $n = \lceil \log_2 2r \rceil + 1$. *Apply* $QFT(2^n) \otimes QFT(2^n)$ *to the two control registers.*
3. *Apply* $\Lambda_{2^n}(U_a)$ *using the first control register and the target register.*
4. *Apply* $\Lambda_{2^n}(U_b)$ *using the second control register and the target register.*
5. *Apply* $QFT^{-1}(2^n) \otimes QFT^{-1}(2^n)$ *to the two control registers.*
6. *Measure the first two registers and denote the outcome* $|\,x_1\rangle\,|\,x_2\rangle$. *Compute* $s_1 = \left[\frac{x_1 r}{2^n}\right]$ *and* $s_2 = \left[\frac{x_2 r}{2^n}\right]$.
7. *If* $s_1 = 0$, *output* $FAIL$. *Otherwise let* $s = s_1^{-1} s_2$ *mod* r. *Output* s.

Note that this algorithm also works in non-Abelian groups G, since it suffices that the subgroup generated by a and b is Abelian, and this of course is true.

Corollary 19. *Let* a *and* $b = a^s$ *be elements from a group* G. *Algorithm 18 (Discrete_Log*(b, a)*) outputs* s *with probability at least* $\left(\frac{r-1}{r}\right)\left(\frac{8}{\pi^2}\right)^2$. *This algorithm has expected running time* $O(\log r)$ *group operations and* $O(\log^2 r)$ *other elementary operations.*

Proof. After step 5 we have the state
$$\sum_{k=0}^{r-1} \left|\,\widetilde{\frac{k}{r}}\right\rangle \left|\,\widetilde{\frac{ks}{r}}\right\rangle |\,\Psi_k\rangle\,.$$
With probability at least $(\frac{8}{\pi^2})^2$ the integers s_1 and s_2 will correspond to integers j and js, where j is chosen uniformly at random from the integers $0, 1, \ldots r-1$. In this case $s_1 = j \neq 0$ with probability $\frac{r-1}{r}$ and we can easily compute $j^{-1}js = s \mod r$. The result follows. The running time follows by noting that the $O(\log r)$ bit arithmetic described can be done with $O(\log^2 r)$ elementary (classical) operations. $\square$

2.7. Amplitude Estimation

In the previous sections we detailed and used a simple algorithm for estimating a phase shift induced by a unitary operator U. Now suppose we wish to estimate a different parameter, namely the probability with which the operator U produces a *good* output $|\, x\rangle$. More specifically, let $U = A$ be a quantum algorithm that starts with the state $|\, 00\ldots 0\rangle$ and the aim is to output states $|\, x\rangle$ that satisfy $f(x) = 1$. The function $f : \{0, 1, \ldots, N-1\} \to \{0, 1\}$ characterises the *good* states. We denote by X_1 the set of *good* states $|\, x\rangle$, namely those satisfying $f(x) = 1$. The set X_0 denotes the *bad* states, which satisfy $f(x) = 0$. Denote by p_1 the probability of obtaining a good state when we measure $|\, \Upsilon\rangle$. As discussed earlier, we can assume that we can implement the operator $U_f : |\, x\rangle \to (-1)^{f(x)}\, |\, x\rangle$. In other words we wish to determine

$$p_1 = \sum_{x \in X_1} |\langle\, x|\, \Upsilon\rangle|^2$$

where

$$|\, \Upsilon\rangle = A\, |\, 00\ldots 0\rangle\,. \tag{18}$$

When $0 < p_1 < 1$, we can assume that

$$A\, |\, 00\ldots 0\rangle = \sqrt{p_0}\, |\, X_0\rangle + \sqrt{p_1}\, |\, X_1\rangle$$

where $p_0 = 1 - p_1$,

$$|\, X_0\rangle = \frac{1}{\sqrt{p_0}} \sum_{x \in X_0} |\, x\rangle\, \langle\, x|\, \Upsilon\rangle$$

and

$$|\, X_1\rangle = \frac{1}{\sqrt{p_1}} \sum_{x \in X_1} |\, x\rangle\, \langle\, x|\, \Upsilon\rangle.$$

We will start by reviewing the algorithm for amplitude amplification [**BBHT98, BH97, BHT98, Gro98, BHMT99**], based on the quantum searching algorithm by Grover [**Gro96**], and then show how the main iterate of this algorithm can be used to solve the amplitude estimation problem mentioned above.

2.7.1. Amplitude Amplification. It is helpful to work in a basis containing the states $|X_0\rangle$ and $|X_1\rangle$ (a full basis contains another $2^n - 2$ states). We have at our disposal the operator U_f which sends $|X_j\rangle \rightarrow (-1)^j |X_j\rangle$, the unitary operators A and A^{-1}, and the operator U_0 which maps $|0\rangle$ to $-|0\rangle$ and leaves the other basis states alone. By conjugating U_0 with the operator A, we have the operator $U_\Upsilon = AU_0A^{-1}$ which sends $|\Upsilon\rangle = A|00\ldots0\rangle$ to $-|\Upsilon\rangle$ and leaves all the other orthogonal basis states alone (it is convenient to describe this operator in a basis containing $|\Upsilon\rangle$). Let us consider the action of the operator $-U_\Upsilon U_f$ in the space spanned by $|X_0\rangle$ and $|X_1\rangle$. Given any state

$$|\Psi\rangle = \cos(\phi)\,|X_0\rangle + \sin(\phi)\,|X_1\rangle\,,$$

applying U_f gives us

$$\cos(\phi)\,|X_0\rangle - \sin(\phi)\,|X_1\rangle\,. \tag{19}$$

Let ω be the real number between 0 and $\frac{1}{2}$ that satisfies $p_1 = \sin^2(\pi\omega)$ and $p_0 = \cos^2(\pi\omega)$. Representing the state (19) in a basis containing $|\Upsilon\rangle = \cos(\pi\omega)\,|X_0\rangle + \sin(\pi\omega)\,|X_1\rangle$ and $\overline{|\Upsilon\rangle} = \sin(\pi\omega)\,|X_0\rangle - \cos(\pi\omega)\,|X_1\rangle$ gives us

$$\cos(\phi)\,|X_0\rangle - \sin(\phi)\,|X_1\rangle = \cos(\phi+\pi\omega)\,|\Upsilon\rangle + \sin(\phi+\pi\omega)\overline{|\Upsilon\rangle}.$$

Since $\overline{|\Upsilon\rangle}$ is orthogonal to $|\Upsilon\rangle$, applying $-U_\Upsilon$ gives us

$$\begin{aligned} &\cos(\phi+\pi\omega)\,|\Upsilon\rangle - \sin(\phi+\pi\omega)\overline{|\Upsilon\rangle} \\ = \;&\cos(\phi+2\pi\omega)\,|X_0\rangle + \sin(\phi+2\pi\omega)\,|X_1\rangle\,. \end{aligned}$$

The two reflections U_f and $-U_\Upsilon$ thus give us a rotation of $2\pi\omega$ in the plane spanned by $|\,X_0\rangle$ and $|\,X_1\rangle$ (see figure 2.5 for an illustration with $\phi = \pi\omega$).

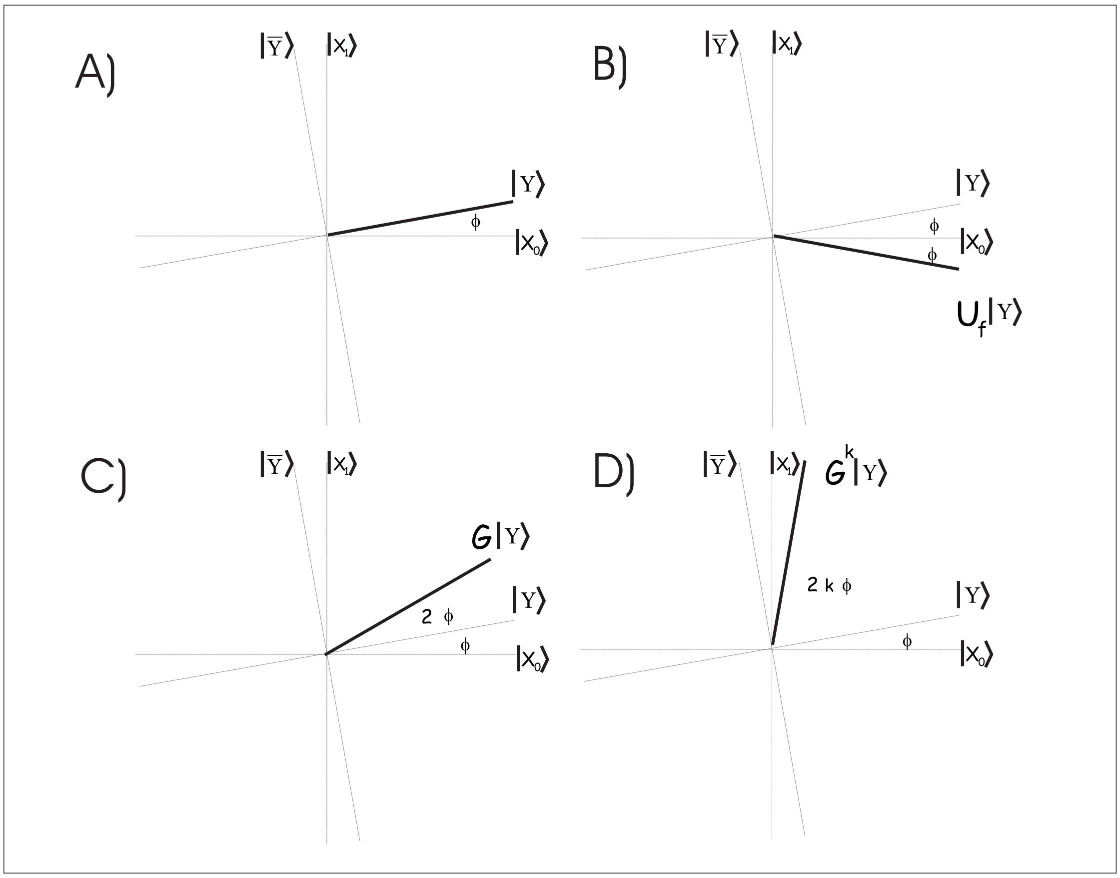

FIGURE 2.5. A) We start off in the state $|\,\Upsilon\rangle = A\,|\,00\ldots0\rangle = \sin(\pi\omega)\,|\,X_1\rangle + \cos(\pi\omega)\,|\,X_0\rangle$. B) The U_f operation changes the sign in front of the $|\,X_1\rangle$ component. C) The $-U_\Upsilon$ (which is equal to a $U_{\overline{\Upsilon}}$) changes the sign in front of the $\left|\,\overline{\Upsilon}\right\rangle$ component. The net effect of applying $G = -U_\Upsilon U_f$ can be described as a rotation of $2\phi = 2\pi\omega$ in the plane spanned by $|\,X_0\rangle$ and $|\,X_1\rangle$. D) Repeating G a total of k times effects a rotation of $2k\phi = 2\pi k\omega$.

Such a rotation $G = -U_\Upsilon U_f$ will have eigenvalues $e^{\pm 2\pi i\omega}$ in the subspace generated by $|\,X_0\rangle$ and $|\,X_1\rangle$. In fact, we can explicitly describe the eigenvectors (as defined in [**Mos98**]):

$$|\,\Psi_+\rangle = \frac{1}{\sqrt{2}}\,|\,X_0\rangle + \frac{i}{\sqrt{2}}\,|\,X_1\rangle \tag{20}$$

$$|\,\Psi_-\rangle = \frac{1}{\sqrt{2}}\,|\,X_0\rangle - \frac{i}{\sqrt{2}}\,|\,X_1\rangle \tag{21}$$

with respective eigenvalues $e^{2\pi i\omega}$ and $e^{-2\pi i\omega}$.

The state

$$|\,\Upsilon\rangle = \cos(\pi\omega)\,|\,X_0\rangle + \sin(\pi\omega)\,|\,X_1\rangle$$

is expressed in this eigenbasis as

$$\frac{e^{\pi i\omega}}{\sqrt{2}}\,|\,\Psi_+\rangle + \frac{e^{-\pi i\omega}}{\sqrt{2}}\,|\,\Psi_-\rangle\,.$$

Since $G\,|\,\Psi_+\rangle = e^{2\pi i\omega}\,|\,\Psi_+\rangle$ and $G\,|\,\Psi_-\rangle = e^{-2\pi i\omega}\,|\,\Psi_-\rangle$, then applying the operator G^k to the state $|\,\Upsilon\rangle = A\,|\,00\ldots0\rangle$ gives us

$$\frac{e^{\pi i(2k+1)\omega}}{\sqrt{2}}\,|\,\Psi_+\rangle + \frac{e^{-\pi i(2k+1)\omega}}{\sqrt{2}}\,|\,\Psi_-\rangle \tag{22}$$

$$= \cos(\pi(2k+1)\omega)\,|\,X_0\rangle + \sin(\pi(2k+1)\omega)\,|\,X_1\rangle\,. \tag{23}$$

The probability of measuring an element of X_1 is $\sin^2(\pi(2k+1)\omega)$ and assuming $p_1 > 0$ then to measure an element of X_1 with probability close to 1 we should apply G roughly $k = \left[\frac{1}{4\omega} - \frac{1}{2}\right] \in \Theta(\frac{\pi}{4\sqrt{p_1}})$ times.

Having summarised the use of the Grover iterate G for amplifying the amplitude $\sqrt{p_1}$ of $|\,X_1\rangle$ in $|\,\Upsilon\rangle = A\,|\,00\ldots0\rangle$, we move on to describe its use for estimating p_1.

2.7.2. Amplitude Estimation. The strategy for estimating $p_1 = \sin^2(\pi\omega)$ is simple. We will just estimate one of the eigenvalues $e^{\pm 2\pi i\omega}$ of G using the eigenvalue estimation algorithm. Note that when $p_1 = 0$ or 1, $|\Upsilon\rangle = A\,|00\ldots0\rangle$ is an eigenvector of G with eigenvalue $e^{2\pi i\omega}$ equal to $1 = e^{\pm 0}$ or $-1 = e^{\pm i\pi}$ respectively, and so we still have $p_1 = \sin^2(\pi\omega)$.

ALGORITHM 20. *(Amp_Est(A, f, M))*

Input:

- *An integer M (a precision parameter).*
- *An integer N.*
- *A quantum network for implementing the operator (algorithm) A acting on a vector space $\mathbb{H}_N$ of dimension N.*
- *A unitary operator which marks elements of a* good *set X_1 by mapping $|x\rangle \to (-1)^{f(x)}\,|x\rangle$, where $f(x) = 1$ iff $x \in X_1$.*

Output:

- *A description of a real number $\tilde{p}$ (i.e. an integer x such that $\tilde{p} = \sin^2(\frac{\pi x}{M})$).*

Complexity:

- *$O(M)$ applications of U_f*
- *$O(M)$ applications of A and A^{-1}*
- *$O(M\log_2 N)$ other elementary operations*

Procedure:

1. *Prepare the input state $|00\ldots0\rangle\, A\,|00\ldots0\rangle \in \mathbb{H}_M \otimes \mathbb{H}_N$.*
2. *Apply $QFT(M)$ to the first register.*
3. *Apply $\Lambda_M(G)$.*
4. *Apply $QFT(M)^{-1}$ to the first register.*

5. *Measure the first register to obtain a state* $|\, x\rangle$, $x \in \{0, 1, \ldots, M-1\}$ *and output* $\tilde{p} = \sin^2(\pi \frac{x}{M})$.

THEOREM 21. *For any positive integer* M*, the algorithm* $Amp\_Est(A, f, M)$ *outputs* $\tilde{p}$ *such that*

- *with probability at least* $\frac{8}{\pi^2}$ *we have*

$$|p_1 - \tilde{p}| \leq 2\pi \frac{\sqrt{p_1(1-p_1)}}{M} + \frac{\pi^2}{M^2}$$

- *for any integer* $k > 1$*, with probability at least* $1 - \frac{1}{2(k-1)}$ *we have*

$$|p_1 - \tilde{p}| \leq 2\pi k \frac{\sqrt{p_1(1-p_1)}}{M} + \frac{k^2\pi^2}{M^2}$$

and uses M *applications of* U_f.

PROOF. After step 3 we have the state

$$\begin{aligned} & \frac{1}{\sqrt{2M}} \sum_{j=0}^{M-1} |\, j\rangle \left(e^{\pi i(2j+1)\omega} \,|\, \Psi_+\rangle + e^{-\pi i(2j+1)\omega} \,|\, \Psi_-\rangle\right) \\ = \; & \frac{e^{\pi i\omega}}{\sqrt{2M}} \sum_{j=0}^{M-1} e^{2\pi i j\omega} \,|\, j\rangle \,|\, \Psi_+\rangle + \frac{e^{-\pi i\omega}}{\sqrt{2M}} \sum_{j=0}^{M-1} e^{-2\pi i j\omega} \,|\, j\rangle \,|\, \Psi_-\rangle \end{aligned}$$

and after step 4 we have the state

$$\frac{e^{\pi i\omega}}{\sqrt{2M}} \widetilde{|\, \omega\rangle}_M \,|\, \Psi_+\rangle + \frac{e^{-\pi i\omega}}{\sqrt{2M}} \widetilde{|\, -\omega\rangle}_M \,|\, \Psi_-\rangle \,.$$

In step 5, with probability $\frac{1}{2}$ we get an estimate $\widetilde{\omega}$ of ω with the properties given in corollary 6. In this case, letting $\Delta = d(\widetilde{\omega}, \omega)$, and applying some standard trigonometric identities, we have

$$\begin{aligned}
|\widetilde{p_1} - p_1| &= \left|\sin^2(\pi\widetilde{\omega}) - \sin^2(\pi\omega)\right| = \left|\sin^2(\pi\omega \pm \pi\Delta) - \sin^2(\pi\omega)\right| \\
&= \left|(\sin(\pi\omega)\cos(\pi\Delta) \pm \sin(\pi\Delta)\cos(\pi\omega))^2 - \sin^2(\pi\omega)\right| \\
&= \left|\sin^2(\pi\Delta)\cos(2\pi\omega) \pm \frac{\sin(2\pi\omega)}{2}\sin(2\pi\Delta)\right| \\
&\leq 2\pi\Delta\sqrt{p_1(1-p_1)} + \pi^2\Delta^2
\end{aligned}$$

The result in the case that we effectively measure $|\,\widetilde{\omega}\rangle$ now follows by corollary 6. Otherwise, we effectively measure $\left|\,\widetilde{-\omega}\right\rangle$, but since $\sin^2(\pi\omega) = \sin^2(-\pi\omega)$, the same result holds, and the theorem follows. □

2.7.3. Application to counting. We give three applications of the amplitude estimation algorithm to counting. Suppose we wish to estimate or determine the cardinality of X_1. Let A be any transformation, such as the Fourier transform F_N, that maps the state $|\,00\ldots0\rangle$ to a uniformly weighted superposition of elements in $\{0, 1, \ldots, N-1\}$. Then $p_1 = \frac{|X_1|}{N}$ and estimates of p_1 translate directly to estimates of $t = |X_1| = Np_1$.

ALGORITHM 22. *(Count(f, M))*

Input:

- *An integer N.*
- *A quantum network for $U_f : |\,x\rangle \to (-1)^{f(x)}\,|\,x\rangle$.*
- *An integer M.*

Output:

- *A description of a real number $\tilde{t}$ (i.e. an integer x such that $\tilde{t} = N\sin^2(\frac{\pi x}{M})$.*

Complexity:

- *$O(M)$ applications of U_f.*
- *$O(M \log^2 N)$ other operations.*

Procedure:

1. *Output $N \times Amp\_Est(F_N, f, M)$*

In [**BHMT99**] are given the following algorithms and theorems (based on results in [**BBHT98, BHT98, Mos98, Mos99**]) related to counting t.

The following theorem and corollary follow easily from Theorem 21.

THEOREM 23. *For any integer $M > 0$, Count(f, M) outputs a real number $\tilde{t}$ satisfying*

- *with probability at least $\frac{8}{\pi^2}$ we have*

$$\left|t - \tilde{t}\right| \leq 2\pi \frac{\sqrt{t(N-t)}}{M} + \frac{N\pi^2}{M^2}$$

- *for any integer $k > 1$, with probability at least $1 - \frac{1}{2(k-1)}$ we have*

$$\left|t - \tilde{t}\right| \leq 2\pi k \frac{\sqrt{t(N-t)}}{M} + \frac{Nk^2\pi^2}{M^2}$$

and uses M applications of U_f.

COROLLARY 24. *Let $\tilde{t} =$ Count$(f, \lceil c\sqrt{N}\rceil)$. Then*

$$\left|\tilde{t} - t\right| \leq \frac{2\pi}{c}\sqrt{\frac{t\,(N-t)}{N}} + \left(\frac{\pi}{c}\right)^2$$

with probability at least $\frac{8}{\pi^2}$.

Note that $Count(f, M)$ outputs a real number. In the following applications we will be rounding off $Count(f, M)$ to an integer. We do not simply round off $Count(f, M)$ to a nearest integer, since in general this requires arbitrary precision. We will instead round off $Count(f, M)$ to an integer that is within $\frac{2}{3}$ of it. The number $\frac{2}{3}$ is arbitrary and can be replaced by any constant fraction strictly between $\frac{1}{2}$ and 1.

We next describe an algorithm for approximately counting t with accuracy ϵ (see [**MR95**]), that is outputting an estimate $\tilde{t}$ such that with probability at least $\frac{2}{3}$ we have $\left|\tilde{t} - t\right| \leq \epsilon t$.

ALGORITHM 25. *(Approx_Count(f, ϵ))*

Input:

- *An integer N.*
- *A quantum network for the operator $U_f : |\, x\rangle \to (-1)^{f(x)} \,|\, x\rangle$.*
- *An accuracy parameter ϵ, $0 < \epsilon \leq 1$ (given, say, as two integers $a, b > 0$, $\epsilon = \frac{a}{b}$).*

Output:

- *An integer $\tilde{t}$.*

Complexity:

- $O\left(\frac{1}{\epsilon}\sqrt{\frac{N}{t+1}}\right)$ *applications of U_f.*
- $O\left(\left(\frac{\log^2 N}{\epsilon}\right)\sqrt{\frac{N}{t+1}}\right)$ *other elementary operations.*

Procedure:

1. *Set $l = 2$.*
2. *Apply Count$(f, 2^l)$.*
3. *If $\tilde{t} = 0$ and $2^l < 2\sqrt{N}$ then increment l by 1 and go to step 2.*
4. *Set $M = \lceil \frac{20\pi^2}{\epsilon} 2^l \rceil$.*
5. *Apply Count(f, M) to obtain t'. Output an integer $\tilde{t}$ satisfying $\left|\tilde{t} - t'\right| \leq \frac{2}{3}$.*

THEOREM 26. *Approx_Count(f, ϵ) outputs an integer $\tilde{t}$ satisfying*

$$\left|t - \tilde{t}\right| \leq \epsilon t$$

with probability at least $\frac{2}{3}$ (t is the number of solutions to $f(x) = 1$). If $t > 0$ it uses an expected number of $O\left(\frac{1}{\epsilon}\sqrt{\frac{N}{t}}\right)$ applications of U_f. If $t = 0$, the algorithm outputs $\tilde{t} = t = 0$ with certainty and U_f is evaluated $\Theta(\sqrt{N})$ times.

PROOF. The case $t = 0$ is easy. For $t > 0$, let $e^{2\pi i\omega}$ be the eigenvalue of $|\Psi_+\rangle$, so $\sin^2(\pi\omega) = \frac{t}{N}$, $0 \leq \omega \leq \frac{1}{2}$. Let $m = \lfloor \log_2\left(\frac{1}{5\pi\omega}\right)\rfloor$. From Lemma 4 we have that the probability that in step 2 $Count(f, 2^l) = 0$ for $l = 1, 2, \ldots, m$ is

$$\prod_{l=1}^{m} \frac{\sin^2(2^l\pi\omega)}{2^{2l}\sin^2(\pi\omega)} \geq \prod_{l=1}^{m}\cos^2(2^l\pi\omega) = \frac{\sin^2(2^{m+1}\pi\omega)}{2^{2m}\sin^2(2\pi\omega)} \geq \cos^2\left(\frac{2}{5}\right).$$

The previous inequalities are obtained by using the fact that $\sin(x\pi\omega) \geq x\sin(\pi\omega)\cos(x\pi\omega)$ for any $x \geq 0$ and $0 \leq \omega x < \frac{1}{2}$, which can be seen by looking at the Taylor expansion of $\tan(x)$ at $x = M\pi\omega$.

Now assuming step 2 was repeated at least m times (note that $2^{m+1} \leq \frac{2}{5}\sqrt{N}$), after step 4 we have $M \geq \frac{4\pi}{\epsilon\omega}$ and since $\pi\omega \leq \frac{\pi}{2}\sin(\pi\omega) = \frac{\pi}{2}\sqrt{\frac{t}{N}}$, then by Theorem 23, the probability that $Count(f, M)$ outputs a number t' satisfying $|t' - t| \leq \frac{\epsilon}{4}t + \frac{\epsilon^2}{64}t$ is at least $\frac{8}{\pi^2}$. If $\epsilon t < 1$, then when we round off t' to get $\tilde{t}$ we will get the correct t and thus the final error is $0 \leq \epsilon t$. If $t\epsilon \geq 1$, then rounding off t' to $\tilde{t}$ can introduce an error of at most $\frac{2}{3} \leq \frac{2}{3}\epsilon t$, which increases the error to something less than $\frac{\epsilon t}{3} + \frac{2}{3} \leq \epsilon t$. The overall probability of obtaining this estimate is at least $\frac{8}{\pi^2}\cos^2(\frac{2}{5}) > \frac{2}{3}$. □

Lastly, we describe an algorithm for exactly counting t.

ALGORITHM 27. *(Exact_Count(f))*

Input:

- *The integer N.*
- *A quantum network for U_f.*

Output:

- *An integer $\tilde{t}$.*

Complexity:

- $O(\sqrt{(t+1)(N-t+1)})$ *applications of U_f.*
- $O(\log^2 N\sqrt{(t+1)(N-t+1)})$ *other elementary operations.*

Procedure:

1. *Set $\tilde{t_1} = Count(f, \lceil 9\sqrt{N}\rceil)$ and $\tilde{t_2} = Count(f, \lceil 9\sqrt{N}\rceil)$.*
2. *Set $M_1 = \lceil 30\sqrt{\tilde{t_1}(N-\tilde{t})}\rceil$, $M_2 = \lceil 30\sqrt{\tilde{t_2}(N-\tilde{t_2})}\rceil$, $\tilde{M} = \min(M_1, M_2)$.*
3. *Apply $Count(f, M)$ to obtain output t'. Output an integer $\tilde{t}$ satisfying $\left|\tilde{t}-t'\right| \le \frac{2}{3}$.*

THEOREM 28. *Algorithm Exact_Count(f) outputs an integer $\tilde{t}$ which equals t, the number of solutions to $f(x)=1$, with probability at least $\frac{2}{3}$. It uses an expected $\Theta(\sqrt{(t+1)(N-t+1)})$ applications of U_f.*

PROOF. By Theorem 21, setting $k=7$, we see that with probability greater than $\frac{11}{12}$, $\tilde{t_1}$ satisfies $\left|\tilde{t_1}-t\right| \le \frac{\sqrt{t(N-t)}}{N} + \frac{1}{4}$, and similarly for $\tilde{t_2}$. This inequality implies that $\sqrt{t(N-t)} \le \sqrt{2}M_j$, for $j=1,2$. Thus with probability at least $\left(\frac{11}{12}\right)^2$ we have

$$\frac{\sqrt{t(N-t)}}{M} \le \frac{\sqrt{2}}{30}.$$

When this is the case, then Theorem 23 tells us that with probability $\frac{8}{\pi^2}$ we have

$$\left|\tilde{t}-t\right| \le \frac{\sqrt{2}\pi}{15} + \frac{\pi^2}{30^2} < \frac{1}{3}$$

(so when we round off in step 3 we will get $\tilde{t}=t$). Therefore with probability at least $\frac{8}{\pi^2}\left(\frac{11}{12}\right)^2 > \frac{2}{3}$ we have $\tilde{t}=t$. □

The following table summarises the complexities of these algorithms, along with their classical counterparts for comparison. The complexity measure we use

Problem	Quantum Complexity	Classical Complexity
Decision	$\Theta(\sqrt{\frac{N}{t+1}})$	$\Theta(\frac{N}{t+1})$
Searching	$\Theta(\sqrt{\frac{N}{t+1}})$	$\Theta(\frac{N}{t+1})$
Count with error $\sqrt{t}$	$\Theta(\sqrt{N})$	$O(N)$
Count with accuracy ϵ	$O((\frac{1}{\epsilon})\sqrt{\frac{N}{t+1}})$	$O((\frac{1}{\epsilon^2})\frac{N}{t+1})$
Exact counting	$\Theta(\sqrt{(t+1)(N-t+1)})$	$\Theta(N)$

TABLE 2.5. This table compares quantum and classical counting complexities. The lower bounds are in the black-box model of computation.

here is the number of applications of U_f (that is evaluations of f) used in order to solve the problem with probability $\frac{2}{3}$ of being correct. The lower bounds are lower bounds for solving these problems in the *black-box* model of computation which we describe in the next section.

For fixed ϵ, or for all $t \leq N(1-\epsilon)$, the upper bound for quantumly counting with accuracy epsilon is tight up to a constant factor since Nayak and Wu [**NW99**] show a lower bound of

$$\Omega\left(\sqrt{\frac{N}{\lceil\epsilon(t+1)\rceil}} + \frac{\sqrt{t(N-t)}}{\lceil\epsilon(t+1)\rceil}\right).$$

We believe it is possible to match the upper and lower bounds into a tight bound for all ϵ and t (Nayak and Wu conjecture their lower bound is tight). I am not aware of a tight lower bound for the classical complexity (some related results appear in [**CEG95**] and [**Gol99**]), but I conjecture that the classical lower bound is simply the square of the quantum lower bound.

2.8. Finding Hidden Subgroups

Let us now return to the problem of finding orders and discrete logarithms and explore the natural generalisations. Both can be described as special cases of the following problem as illustrated in figure 2.6.

PROBLEM 29. *Let f be a function from a finitely generated group G to a finite set X such that f is constant on the cosets of a subgroup K (of finite index, since X is finite), and distinct on each coset. Given a quantum network for evaluating f, namely $U_f : |x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$, find a generating set for K.*

Generalisations of this form have been known since shortly after Shor presented his factoring and discrete logarithm algorithms (for example, [**Vaz97**] presents the hidden subgroup problem for a large class of finite Abelian groups, or more generally in [**Høy97**] for finite Abelian groups presented as a product of finite cyclic groups. In [**ME99**] the natural Abelian hidden subgroup algorithm is related to eigenvalue estimation.)

Other algorithms which can be formulated in this way include:

Deutsch's Problem: Consider a function f mapping $\mathbb{Z}_2 = \{0,1\}$ to $\{0,1\}$. Then $f(x) = f(y)$ if and only if $x - y \in K$, where K is either $\{0\}$ or $\mathbb{Z}_2 = \{0,1\}$. If $K = \{0\}$ then f is $1-1$ or 'balanced' and if $K = \mathbb{Z}_2$ then f is constant. [**Deu85, CEMM98**].

Simon's Problem: Consider a function f from $\mathbb{Z}_2^l$ to some set X with the property that $f(x) = f(y)$ if and only if $x - y \in \{0, s\}$ for some $s \in \mathbb{Z}_2^l$. Here $K = \{0, s\}$ is the hidden subgroup of $\mathbb{Z}_2^l$. Simon [**Sim94**] presents an efficient algorithm for solving this problem, and the solution to the hidden subgroup problem in the Abelian case is a generalisation.

Finding Orders (Factoring): Let a be an element of a group H. Consider the function f from $\mathbb{Z}$ to the group H where $f(x) = a^x$ for some element a of H. Then $f(x) = f(y)$ if and only if $x - y \in r\mathbb{Z}$. The hidden subgroup is $K = r\mathbb{Z}$ and a generator for K gives us the order r of a.

Discrete Logarithms: Let a be an element of a group H, with $a^r = 1$, and suppose $b = a^k$ from some unknown k. The integer k is called the *discrete logarithm of b to the base a.* Consider the function f from $\mathbb{Z}_r \times \mathbb{Z}_r$ to H satisfying $f(x_1, x_2) = a^{x_1}b^{x_2}$. Then $f(x_1, x_2) = f(y_1, y_2)$ if and only if $(x_1, x_2) - (y_1, y_2) \in \{(t, -tk), t = 0, 1, \ldots, r-1\}$ which is the subgroup $\langle(1, -k)\rangle$ of $\mathbb{Z}_r \times \mathbb{Z}_r$. Thus finding a generator for the hidden subgroup K will solve the discrete logarithm k.

Hidden Linear Functions: Let g be some permutation of $\mathbb{Z}_N$ for some integer N. Let h be a function from $\mathbb{Z} \times \mathbb{Z}$ to $\mathbb{Z}_N$, $h(x, y) = x + ay \mod N$. Let $f = g \circ h$. The hidden subgroup of f is $\langle(-a, 1)\rangle$. Boneh and Lipton [**BL95**] show that even if the linear structure of h is hidden, we can recover the parameter a.

Self-Shift-Equivalent Polynomials: Given a polynomial P in l variables $X_1, X_2, \ldots, X_l$ over $\mathbb{F}_q$, the function f which maps $(a_1, a_2, \ldots, a_l) \in \mathbb{F}_q^l$ to $P(X_1 - a_1, X_2 - a_2, \ldots, X_l - a_l)$ is constant on cosets of a subgroup K of $\mathbb{F}_q^l$. This subgroup K is the set of shift-self-equivalences of the polynomial P. Grigoriev [**Gri97**] shows how to compute this subgroup.

Abelian Stabiliser Problem: Let G be any group acting on a finite set X. That is each element of G acts as a map from X to X in such a way that for any two elements $a, b \in G$, $a(b(x)) = (ab)(x)$ for all $x \in X$. For a particular element $x \in X$, the set of elements which fix x (that is the elements $a \in G$ such that $a(x) = x$) form a subgroup. This subgroup is called the stabiliser of x in G, denoted $St_G(x)$. Let f_x denote the function from G to X which maps $g \in G$ to $g(x)$. The hidden subgroup corresponding to f_x is $St_G(x)$. The finitely generated

Abelian case of this problem was solved by Kitaev [**Kit95**], and includes finding orders and discrete logarithms as special cases.

When K is normal in G, we could in fact decompose f as $g \circ h$, where h is a homomorphism from G to some finite group H, and g is some 1-1 mapping from H to a set X which 'hides' the homomorphism structure of h. In this case, K corresponds to the kernel of h and H is isomorphic to G/K.

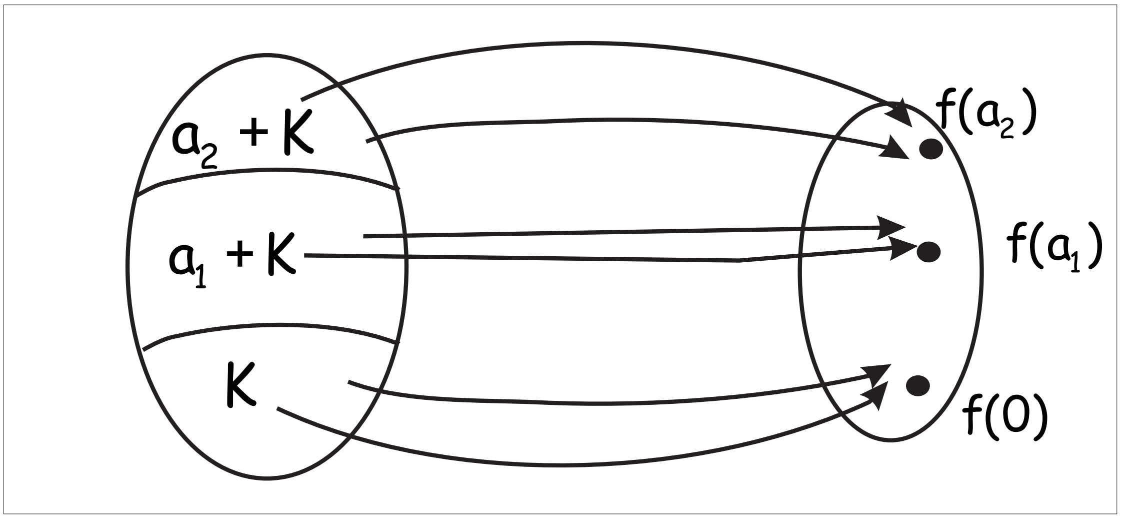

FIGURE 2.6. The function f is constant on cosets of K and distinct on each coset.

We give the complexity in terms of $n = \lceil \log_2 [G : K] \rceil$ which is a lower bound on $\log_2 |X|$ (note that underestimating the input size results in an overestimated complexity, so any upper bound is still valid).

Let us start with a simple example, namely $G = \mathbb{Z}$. This special case is almost identical to the order-finding algorithm, and will be useful for reducing the case of finitely generated G to finite G. This algorithm is needed for finding the order of $a \in G$ when a^{-1} is not known.

2.8.1. Finding a period. Let us suppose we have a function $f : \mathbb{Z} \to X$, where $|X| \leq N < \infty$, and f has the property that $f(x) = f(y)$ if and only if $x - y \in rZ$. Thus the sequence $f(0), f(1), \ldots$ is periodic with period r. The case where f is not $1 - 1$ in this range is addressed in [**BL95**] and in the appendix of [**ME99**]. Consider the 'eigenstates'

$$|\,\Psi_k\rangle = \sum_{j=0}^{r-1} e^{-2\pi i j \frac{k}{r}}\,|\,f(j)\rangle\,.$$

Why do I call these 'eigenstates'? Eigenstates of what? They are eigenstates of the shift operation $|\,f(x)\rangle \to |\,f(x+y)\rangle$. The respective eigenvalues are $e^{2\pi i y \frac{k}{r}}$, analogous to the eigenvalues of the $|\,\Psi_k\rangle$ from equation (15).

The period-finding algorithm goes as follows.

ALGORITHM 30.

Input:

- *An integer* N*,*
- *An integer* M*.*
- *A unitary operator* U_f *which acts on* $\mathbb{H}_M \times \mathbb{H}_N$ *and maps* $|\,x\rangle\,|\,0\rangle \to |\,x\rangle\,|\,f(x)\rangle$*.*

Output:

- *A positive integer* t*.*

Complexity:

- 1 *application of* U_f *on inputs of size* $\lceil \log_2 M \rceil$*.*
- $O(\log^2 M)$ *other elementary operations.*

Procedure:

The procedure is the same as in Algorithm 13 except that instead of using Algorithm Eig_Est we use the following (almost identical) procedure:

1a: *Start in the state* $|00\ldots0\rangle|00\ldots0\rangle$.

1b: *Apply* $QFT(M)$ *to the first register.*

1c: *Apply* U_f.

1d: *Apply* $QFT(M)^{-1}$ *to the first register.*

1e: *Measure the first register and output the measured integer* y.

THEOREM 31. *If* $M > 2r^2$*, then Algorithm 30 finds the correct period* r *of* f *with probability at least* $\frac{32}{\pi^4}$*. If it does not output FAIL, then it outputs a multiple of* r.

PROOF. After step 1c we have the state

$$\begin{aligned} &\sum_{x=0}^{M-1} |x\rangle|f(x)\rangle \\ = &\sum_{k=0}^{r-1}\left(\sum_{x=0}^{M-1} e^{\frac{2\pi i x k}{r}}|x\rangle\right)|\Psi_k\rangle . \end{aligned}$$

Thus after step 1d, measuring the first register produces the same distribution as Algorithm 7 ($Eig\_Est(U_a, \rho, M)$), as described in Proposition 14, and the result follows similarly. □

ALGORITHM 32. *(Find_Period(f))*

Input:

- *A uniform algorithm which outputs for any integer* M *a quantum network* U_f *which acts on* $\mathbb{H}_M \times \mathbb{H}_N$ *and maps* $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$.

Output:

- *A positive integer* t.

Complexity:

- *Expected* $O(\log_2 r)$ *applications of* U_f.
- *Expected* $O(n^2)$ *other elementary operations where* $n = \lceil \log_2 [G : K] \rceil = \lceil \log_2 r \rceil$.

Procedure:

Same as Algorithm 15 (Find_Order(a)) except using Algorithm 30) instead of Algorithm 13.

COROLLARY 33. *Algorithm 30 finds a multiple of* r. *With probability at least* $\frac{2}{3}$ *this multiple is* r. *The expected complexity is* $O(\log r)$ *applications of* U_f *and* $O(\log^2 r)$ *other elementary operations.*

Having thus described the one-dimensional infinite order case, we are ready to reduce the general finitely generated Abelian case to the finite Abelian case.

2.8.2. The general case. Let us assume that we have a set of generators for G. Since f has finite index, K must contain some power of each of these generators. We can use the period-finding algorithm to find the period of f on each of these generators. We can then assume each of the generators has finite order. The algorithm described here is for G identified with a product of cyclic groups, namely $\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2} \times \cdots \times \mathbb{Z}_{N_2}$. Although all Abelian groups are isomorphic to such a product of cyclic groups, finding such a product and the isomorphism can be very difficult (at least as hard as factoring integers when $G = \mathbb{Z}_N^*$). In the next section, we show how a reasonable representation of a finite Abelian group G can be decomposed in this way by using this quantum algorithm.

With the factoring algorithm of Shor, we can factor each N_j into its prime power factors, and then efficiently find the isomorphism between the additive group $\mathbb{Z}_{N_j}$ and a product of cyclic groups of prime power order. For example, if $N = pq$, where p and q are coprime, and $\langle a \rangle$ is identified with $\mathbb{Z}_N$, then $\mathbb{Z}_N \approx \langle a \rangle = \langle a^p \rangle \oplus \langle a^q \rangle \approx$

$\mathbb{Z}_q \times \mathbb{Z}_p$. We can thus restrict our attention to such products of groups where the N_j are prime powers.

Furthermore, any subgroup K of an Abelian $G = G_{p_1} \times G_{p_2} \times \cdots \times G_{p_k}$ where G_{p_j} is the p_j-subgroup (the p_j are distinct primes) of G, is of the form $K_{p_1} \times K_{p_2} \times \cdots \times K_{p_k}$ where $K_{p_j} \leq G_{p_j}$. We can thus find the hidden subgroup K of f piecewise in the following way. For $j = 1, 2, \ldots, k$, we find the hidden subgroup K_{p_j} of the function $f_{p_j} : G_{p_j} \to X$, where $f_{p_j}(x) = f(0, 0, \ldots, 0, x, 0, \ldots, 0)$ (x appears in the jth entry), and then set $K = K_{p_1} \times K_{p_2} \times \cdots \times K_{p_k}$.

So we restrict attention to the case of finite groups G equal to $G_p \approx Z_{p^{a_1}} \times Z_{p^{a_2}} \times \cdots \times Z_{p^{a_k}}$ for some prime p and positive integers a_j. Let $a = \max\{a_j\}$.

Let us define more general 'eigenstates'. Let T be the set of tuples $(t_1, t_2, \ldots, t_k) \in \mathbb{Z}_{p^{a_1}} \times \mathbb{Z}_{p^{a_2}} \times \cdots \times \mathbb{Z}_{p^{a_k}}$ that satisfy

$$p^a \sum_{j=1}^{k} \frac{h_j t_j}{p^{a_j}} = 0 \mod p^a \text{ for all } h \in K. \tag{24}$$

For each such $\mathbf{t} \in T$, define

$$\left|\, \Psi_{(t_1, t_2, \ldots, t_k)} \right\rangle = \sum_{\mathbf{s} \in G/K} e^{-2\pi i \sum_{j=1}^{k} \frac{t_j s_j}{p^{a_j}}} \,|\, f(\mathbf{s}) \rangle \,.$$

The sum is over a set $\mathbf{s}$ of coset representatives for K in G. Since the $\mathbf{t}$ satisfy equation (24), the $|\, \Psi_{\mathbf{t}} \rangle$ are well-defined. Again these are eigenvectors of the shift operations $|\, f(x) \rangle \to |\, f(x+y) \rangle$ with respective eigenvalue

$$e^{2\pi i \sum_{j=1}^{k} \frac{y_j t_j}{p^{a_j}}}.$$

From these eigenvalues the following algorithm determines a uniformly random $\mathbf{t} \in T$, and by equation (24), we can use enough random t to determine K using linear algebra. For simplicity, we assume here that we can perform $QFT(p^a)$

perfectly. In practice, we could use $QFT(2^n)$ where n is in $O(a\log p)$, or other approximate versions of it, and succeed with probability close to 1.

ALGORITHM 34. *(Find_Hidden_Subgroup(f))*

Input:

- *Integers* $p, a_1, a_2, \dots, a_k$.
- *A quantum network for implementing* U_f.

Output:

- *A* k*-tuple* $\mathbf{t} \in \mathbb{Z}_{p^{a_1}} \times \mathbb{Z}_{p^{a_2}} \times \cdots \times \mathbb{Z}_{p^{a_k}}$.

Procedure:

1. *Start with the state* $|\,0\rangle\,|\,0\rangle \cdots |\,0\rangle\,|\,00\dots0\rangle \in \mathbb{H}_{p^{a_1}} \times \mathbb{H}_{p^{a_2}} \times \cdots \times \mathbb{H}_{p^{a_k}} \times \mathbb{H}$.
2. *Apply* $QFT(p^{a_1}) \otimes QFT(p^{a_2}) \otimes \cdots \otimes QFT(p^{a_k}) \otimes I$.
3. *Apply* U_f.
4. *Apply* $QFT(p^{a_1})^{-1} \otimes QFT(p^{a_2})^{-1} \otimes \cdots \otimes QFT(p^{a_k})^{-1} \otimes I$.
5. *Measure the control registers and output the measured values* $t_1, t_2, \dots, t_k$.

PROPOSITION 35. *Algorithm 34 (Find_Hidden_Subgroup(f)) outputs an element* $\mathbf{t}$ *satisfying*

$$p^a \sum_{j=1}^{k} \frac{h_j t_j}{p^{a_j}} = 0 \mod p^a \text{ for all } h \in K.$$

PROOF. After step 3 we have the state

$$\sum_{x \in Z_{p^{a_1}} \times Z_{p^{a_2}} \times \cdots \times Z_{p^{a_k}}} |\,x\rangle\,|\,f(x)\rangle\,,$$

which can be reexpressed in a different basis (in the same way as we did in the proof of Theorem 31) as

$$= \sum_{\mathbf{t} \in T} \left(\sum_{x_1=0}^{p^{a_1}-1} e^{2\pi i \frac{x_1 t_1}{p^{a_1}}} \,|\,x_1\rangle \right) \cdots \left(\sum_{x_k=0}^{p^{a_k}-1} e^{2\pi i \frac{x_k t_k}{p^{a_k}}} \,|\,x_k\rangle \right) |\,\Psi_t\rangle\,.$$

It is then clear that after step 4 we will have

$$\sum_{\mathbf{t}\in T} |\, t_1\rangle \,|\, t_2\rangle \cdots |\, t_k\rangle \,|\, \Psi_t\rangle \,.$$

□

2.8.3. Decomposing Abelian Groups. The hidden subgroup algorithm described above requires that the finite Abelian group G be identified with $Z_{N_1} \times \cdots \times Z_{N_k}$ and not just isomorphic to such a group with no efficiently computable isomorphism available.

Here we show that using the Hidden Subgroup Algorithm itself, we can find this isomorphism with high probability for some 'reasonably' presented finite groups G. We will assume the following about the presentation of G.

1. We have a unique binary representation for each element of G and we can efficiently recognise if a binary string represents an element of G or not.
2. Using the binary representation, for any $a \in G$, we can efficiently construct a quantum network for implementing $U_a : |\, y\rangle \to |\, ay\rangle$. This might seem like a strong assumption, since this means we can compute a^{-1}. But in light of the quantum period-finding algorithm, which allows us to find the order r of a and then easily compute $a^{-1} = a^{r-1}$, this becomes a fair assumption.
3. We can efficiently find a generating set for G.

The last assumption might seem strong, but for finite groups G note that it suffices that we have an upper bound 2^k on the size of G and that we can efficiently select elements of G uniformly at random. It is easy to show that $O(k)$ randomly selected elements of G will span G with probability $1 - o(1)$.

We can again assume the orders of the generators are of prime power order. Further, we can first find generators for each of the p-subgroups of G, and then take their product to obtain G. Thus we restrict attention to p-groups G.

ALGORITHM 36. *(Decompose_Group$(a_1, a_2, \ldots, a_k)$)*

Input:

- *Generators $a_1, a_2, \ldots, a_k$ of the p-group G.*
- *The maximum order $q = p^r$ of the elements $a_1, \ldots, a_k$.*

Output:

- *A set of elements $g_1, g_2, \ldots, g_l$, $l \leq k$, from the group G.*

Complexity:

- *$O(k \log q)$ quantum group multiplications.*
- *$O(k^2 \log q)$ classical group multiplications.*
- *$O(k^3 \log^2 q)$ other elementary operations.*

Procedure:

1. *Define $g : \mathbb{Z}_q^k \to G$ by mapping $(x_1, x_2, \ldots, x_k) \to g(\mathbf{x}) = a_1^{x_1} a_2^{x_2} \cdots a_k^{x_k}$. Use Find_Hidden_Subgroup(g) to find generators for the hidden subgroup K of $\mathbb{Z}_q^k$ as defined by the function g.*
2. *Compute a set $\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_l} \in \mathbb{Z}_q^k/K$ of generators for $\mathbb{Z}_q^k/K$ (for example, see Section 2.4.4 and Algorithm 2.4.14 of* [**Coh93**]*).*
3. *Output $\{g(\mathbf{y_1}), g(\mathbf{y_2}), \ldots, g(\mathbf{y_l})\}$.*

THEOREM 37. *The group G is equal to $\langle g_1 \rangle \oplus \langle g_2 \rangle \oplus \ldots \oplus \langle g_l \rangle$ and Algorithm Decompose_Group$(a_1, a_2, \ldots, a_k)$ finds the generators $g_1, g_2, \ldots, g_l$ using $O(k \log q)$ quantum group multiplications, $O(k^2 \log q)$ classical group multiplications, and $O(k^3 \log^2 q)$ other elementary operations.*

PROOF. Evaluating the function g requires at most k exponentiations with exponents of size at most p^a (these require $O(a \log p)$ group operations each), plus $k - 1$ multiplications to multiply the results together. This accounts for the $O(k \log q)$ (quantum) group operations.

Standard matrix reductions over the integers will find the elements $\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_k}$ at a cost of $O(k^3)$ arithmetic operations with integers of size at most q, which accounts for the $O(k^3 \log^2 q)$ other operations. From these we can compute each $g_j = g(\mathbf{y_j})$ using at most $O(k \log q)$ group operations.

The function g is a surjective homomorphism from $\mathbb{Z}_q^k$ to G with kernel K, and thus the First Isomorphism Theorem implies that $\mathbb{Z}_q^k/K$ is isomorphic to G, and we can identify the generators $\mathbf{y_j} + K$ of $\mathbb{Z}_q^k/K$ with generators g_j of G via the equation $g(\mathbf{y_j}) = \mathbf{g_j}$. □

2.8.4. Discussion: What about non-Abelian groups? If the group G is not Abelian, then it is not a product of cyclic groups. Algorithm $Find\_Hidden\_Subgroup(f)$ does not apply and it is not clear how to proceed.

For example, suppose G is S_n, the group of permutations of n points and X is the set of n-vertex graphs. Let $f(\pi) = \pi(A)$, where π is a permutation and $\pi(A)$ is the graph obtained from A by relabelling vertex j with $\pi(j)$. The function f is constant on left cosets of the automorphism group $Aut(A)$ of A. Beals [**Bea97**] showed how to implement a quantum version of the Fourier transform for S_n, but this has not lead to an algorithm for finding the automorphisms of graphs.

Ettinger [**Ett98**] showed how to find a hidden subgroup of a di-hedral group G if K is normal in G using only a polynomial number of applications of f. For K not necessarily normal, Ettinger and Høyer [**EH98**] show that with a polynomial number of applications of f they have enough information to find K, however, no efficient way of performing the post-processing is known.

Rötteler and Beth [**RB98**] show how to find hidden subgroups of certain wreath product groups. Zalka [**Zal99**] showed how to reduce this problem to several instances of an Abelian hidden subgroup problem.

QUESTION 38. *Can other non-Abelian hidden subgroup problems be efficiently unravelled into a polynomial number of instances of the Abelian hidden subgroup algorithm?*

2.9. Equivalence of Shor and Kitaev approaches

I will first show the equivalence between Shor's factoring algorithm and our version of the factoring algorithm following Kitaev's eigenvalue estimation approach. I will then generalise the equivalence to the hidden subgroup algorithm.

This section starts with a review of the quantum part of Shor's algorithm (with some trivial modifications to highlight the similarities).

ALGORITHM 39. *($Shor\_Factor(N)$)*

Input:

- *An integer N.*
- *An element $a \in \mathbb{Z}_N^*$.*

Output:

- *An integer t which approximates $\frac{kN}{r}$ for a uniformly random integer k from $\{1, 2, \ldots, r\}$, where r is the order of a.*

Complexity:

- *One application of $\Lambda_M(U_a)$, $M = 2N^2$.*
- *$O(\log^2 M)$ other operations.*

Procedure

1. *Start with two registers $|\,00\ldots0\rangle\,|\,1\rangle$.*
2. *Apply $QFT(M)$ to the first register.*
3. *Evaluate the function $f(x) = a^x$ using the operator $U_a : |\,x\rangle\,|\,y\rangle \to |\,x\rangle\,|\,a^x y\rangle$.*

4. *Apply $QFT(M)^{-1}$ to the first register.* [1]

After step 4, we have the state

$$\sum_{j=0}^{M-1} |\, x\rangle \,|\, a^x\rangle \,. \tag{25}$$

Shor describes this in the computational basis as

$$\sum_{k=0}^{r-1} \left(\sum_{j=0}^{\lfloor \frac{M-1-k}{r} \rfloor} |\, rj+k\rangle \right) |\, a^k\rangle \,.$$

He then mentions a measurement of the second register to 'collapse' the first register into a periodic superposition

$$\left(\sum_{j=0}^{\lfloor \frac{M-1-k}{r} \rfloor} |\, rj+k\rangle \right)$$

for some random k. **Since we never make use of the measured value a^k, this 'measurement' was only an illustrative tool.** We ignore it.

The final $QFT(M)^{-1}$ maps the periodic superposition to a superposition where the amplitudes are concentrated near values of x where $\frac{x}{M}$ is close to $\frac{j}{r}$ for some integer j between 0 and $r-1$. The information about k is encoded primarily in the phases:

$$\sum_{k=0}^{r-1} QFT(M)^{-1} \left(\sum_{j=0}^{\lfloor \frac{M-1-k}{r} \rfloor} |\, rj+k\rangle \right) |\, a^k\rangle = \sum_{k=0}^{r-1} e^{-2\pi i \frac{k}{M}} QFT(M)^{-1} \left(\sum_{j=0}^{\lfloor \frac{M-1-k}{r} \rfloor} |\, rj\rangle \right) |\, a^k\rangle \,. \tag{26}$$

[1] Shor actually uses $QFT(M)$. The net result is the same.

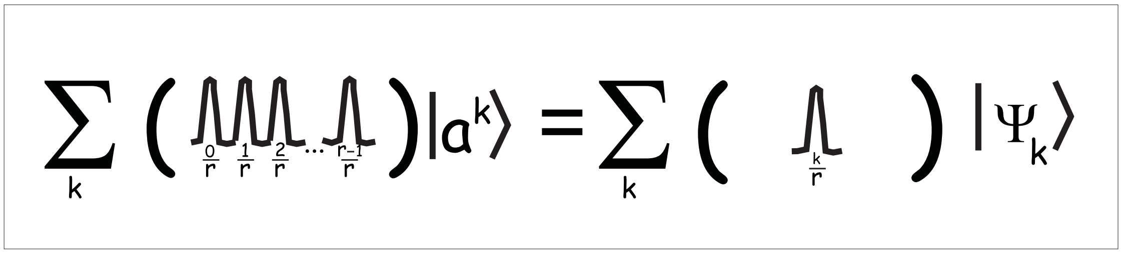

FIGURE 2.7. Shor analysed the above state in the computational basis, whereas we analysed it in the eigenvector basis.

Since we ignore (or "trace out") the second register, the first state is in a mixture of states of the form

$$QFT(M)^{-1}\left(\sum_{j=0}^{\lfloor\frac{M-1-k}{r}\rfloor} |\, rj\rangle\right) |\, a^k\rangle$$

which each have most of the probability amplitude near states $|\, y\rangle$ such that $\frac{y}{M}$ is close to $\frac{k}{r}$ for some integer k, as shown by Shor [**Sho95b**].

Using our eigenvalue estimation approach, we also have the state (25), and we also apply $QFT(M)^{-1}$ to the first register to produce the same state as in equation (26) but we described it in a different basis as

$$\sum_{k=0}^{r} \left|\, \widetilde{\frac{k}{r}}\right\rangle |\, \Psi_k\rangle$$

as illustrated in figure 2.7.

We can similarly compare the common analysis of the hidden subgroup algorithm in the computational basis, where after applying the QFT^{-1} to the first register in the state

$$\sum_{(x_1,x_2,\dots,x_l)} |\, \mathbf{x}\rangle\, |\, f(\mathbf{x})\rangle = \sum_{\mathbf{y}\in G/K}\left(\sum_{\mathbf{s}\in K} |\, \mathbf{s}+\mathbf{y}\rangle\right) |\, f(\mathbf{y})\rangle\,,$$

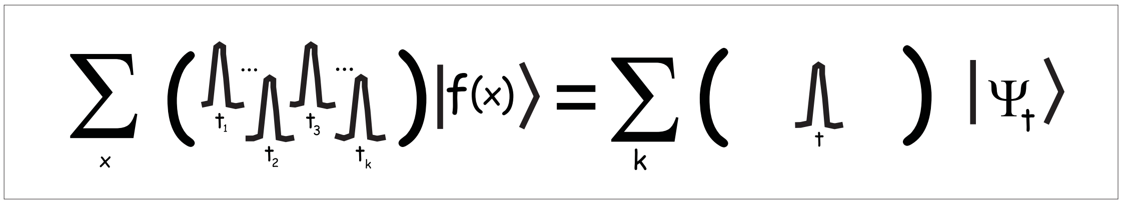

FIGURE 2.8. One common approach is to analysis the above state in the computational basis, but analysing the second register in the 'eigenvector' basis produces a convenient description of the final state.

we get

$$\sum_{\mathbf{y}\in G/K}\left(\sum_{\mathbf{t}\in T} e^{-2\pi i \sum \frac{t_j y_j}{p^{a_j}}}\,|\,\mathbf{t}\rangle\right)|\,f(\mathbf{y})\rangle$$

(the first summation is over a set of coset representatives $\mathbf{y}$ of G/K) whereas in our analysis we write this in the 'eigenvector' basis as

$$\sum_{\mathbf{t}\in T}|\,\mathbf{t}\rangle\,|\,\Psi_t\rangle$$

as illustrated in figure 2.8.

2.10. Finding Hidden Affine Functions

As pointed out in the previous section, Deutsch's problem (section 2.1) is a very special instance of the hidden subgroup problem. It can also be rephrased as follows.

PROBLEM 40. *Given an integer N and a function $f : x \to mx + b$, where $x, m, b \in \mathbb{Z}_2$, find m.*

We can partially generalise this problem as follows.

PROBLEM 41. *Given an integer N function $f : x \to mx + b$, where $x, m, b \in \mathbb{Z}_N$, find m.*

Since b could be any integer from $\mathbb{Z}_N$, one classical evaluation of f gives no information about m. The next algorithm, however solves Problem 41 with just one application of U_f.

ALGORITHM 42.

Input:

- *An integer N.*
- *A black-box that implements $U_f : |\, x\rangle \,|\, y\rangle \to |\, x\rangle \,|\, y + f(x)\rangle$.*

Output:

- *An integer $m \in \mathbb{Z}_N$.*

Complexity:

- 1 *application of U_f*
- 1 *application of $QFT(N)$ and* 2 *applications of $QFT(N)^{-1}$.*

Procedure:

1. *Start with two registers in the state $|\, 00\ldots 0\rangle \,|\, \Psi_1\rangle \in \mathbb{H}_N \times \mathbb{H}_N$ where $|\, \Psi_1\rangle = QFT(N)^{-1} \,|\, 1\rangle$.*
2. *Apply $QFT(N)$ to the first register.*
3. *Apply U_f.*
4. *Apply $QFT(N)^{-1}$ to the first register.*
5. *Measure the first register and output the measured value.*

PROPOSITION 43. *Algorithm 42 solves Problem 41.*

PROOF. The second register is in the state

$$|\,\Psi_1\rangle = \sum_{j=0}^{N-1} e^{-2\pi \frac{j}{N}}\,|\,j\rangle$$

which is an eigenstate of the operation $|\,y\rangle \rightarrow |\,y + f(x)\rangle$ with eigenvalue $e^{2\pi i \frac{f(x)}{N}}$.

After step 3 we will have the state

$$\begin{aligned} & \sum_{x=0}^{N-1} e^{2\pi i \frac{f(x)}{N}}\,|\,x\rangle\,|\,\Psi_1\rangle \\ = \; & e^{2\pi i \frac{b}{N}} \left(\sum_{x=0}^{N-1} e^{2\pi i \frac{mx}{N}}\,|\,x\rangle \right) |\,\Psi_1\rangle \end{aligned}$$

Thus after applying $QFT(N)^{-1}$ to the first register we get $e^{2\pi i \frac{b}{N}}\,|\,m\rangle\,|\,\Psi_1\rangle$. □

As with the hidden subgroup algorithm, we are measuring the eigenvalues of shift functions $|\,f(x)\rangle \rightarrow |\,f(x+y)\rangle$, except in this case we know the eigenvectors and can efficiently construct one, and the respective eigenvalue directly gives us the desired solution (in contrast, the hidden subgroup algorithms only output elements 'orthogonal' to the solution, from which the solution is derived by linear algebra).

Bernstein and Vazirani [**BV97**] generalised the Deutsch problem another way that we describe here.

PROBLEM 44. *Given a function* $f : x \rightarrow \mathbf{m}^T \cdot \mathbf{x} + b$, *where* $\mathbf{x}, \mathbf{m} \in \mathbb{Z}_2^n$, $b \in \mathbb{Z}_2$, *find* m.

This following algorithm is a refinement of the one proposed in [**BV97**] for solving Problem 44.

ALGORITHM 45.

Input:

- *An integer* N.
- *A black-box that implements* $U_f : |\,\mathbf{x}\rangle\,|\,y\rangle \to |\,\mathbf{x}\rangle\,|\,y + f(\mathbf{x})\rangle$.

Output:

- *A tuple* $\mathbf{m} \in \mathbb{Z}_2^n$.

Complexity:

- 1 *application of* U_f.
- $2n+1$ *applications of* $H = QFT(2) = QFT(2)^{-1}$.

Procedure:

1. *Prepare the state* $|\,00\ldots0\rangle\,(|\,0\rangle - |\,1\rangle)$.
2. *Apply* $QFT(2) \times QFT(2) \times \cdots \times QFT(2)$ *to the first register.*
3. *Evaluate* U_f.
4. *Apply* $QFT(2)^{-1} \times QFT(2)^{-1} \times \cdots \times QFT(2)^{-1}$ *to the first register.*
5. *Measure the first register and output the measured values.*

PROPOSITION 46. *Algorithm 45 solves Problem 44.*

PROOF. After step 2 we have the state

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |\,x\rangle\,(|\,0\rangle - |\,1\rangle)$$

and after step 3 we have

$$\begin{aligned}
& \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}\,|\,x\rangle\,(|\,0\rangle - |\,1\rangle) \\
= & \frac{(-1)^b}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (-1)^{m\cdot x}\,|\,x\rangle\,(|\,0\rangle - |\,1\rangle) \\
= & \frac{(-1)^b}{\sqrt{2^{n+1}}} (|\,0\rangle + (-1)^{m_1}\,|\,1\rangle)\,(|\,0\rangle + (-1)^{m_2}\,|\,1\rangle)\cdots(|\,0\rangle + (-1)^{m_n}\,|\,1\rangle)\,(|\,0\rangle - |\,1\rangle)\,.
\end{aligned}$$

Thus after applying a Hadamard transform $H = QFT(2)^{-1}$ on each of the first n qubits we get $\frac{(-1)^b}{\sqrt{2}}\,|\,m_1\rangle\,|\,m_2\rangle\ldots|\,m_n\rangle\,(|\,0\rangle - |\,1\rangle)$. □

We now combine problems 41 and 44 and get the following.

PROBLEM 47. *Given a function* $f : \mathbf{x} \to \mathbf{m}^T \cdot \mathbf{x} + b$*, where* $\mathbf{x}, \mathbf{m} \in \mathbb{Z}_N^n$, $b \in \mathbb{Z}_N$*, find* $\mathbf{m}$.

Algorithm 45, replacing $QFT(2)$ with $QFT(N)$, and replacing $(|\,0\rangle - |\,1\rangle) = QFT(2)^{-1}\,|\,1\rangle$ with $|\,\Psi_1\rangle = QFT(N)^{-1}\,|\,1\rangle$ will solve this problem. The proof follows similarly, so it is omitted.

PROBLEM 48. *Given a function* $f : \mathbf{x} \to \mathbf{M}\mathbf{x} + \mathbf{b}$*, where* $M \in M_{m\times n}(N)$*,* $\mathbf{x}, \mathbf{b} \in \mathbb{Z}_N^n$*, find* $\mathbf{M}$.

Let us denote by $\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_m}$ the rows of $\mathbf{M}$ and $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_n}$ the columns. So

$$\mathbf{M} = \left(\begin{array}{c} \mathbf{r_1} \\ \hline \mathbf{r_2} \\ \hline \cdots \\ \hline \mathbf{r_m} \end{array}\right) = \left(\begin{array}{c|c|c|c} \mathbf{c_1} & \mathbf{c_2} & \ldots & \mathbf{c_n} \end{array}\right).$$

We describe an algorithm (illustrated in figure 2.9) for determining $\mathbf{d}^T \cdot \mathbf{M}$ for any $\mathbf{d} \in \mathbb{Z}_N^n$.

ALGORITHM 49.

Input:

- *The integers* N*,*n*, and* m*.*
- *A black-box that implements* $U_f : |\,\mathbf{x}\rangle\,|\,\mathbf{y}\rangle \to |\,\mathbf{x}\rangle\,|\,\mathbf{y} + f(\mathbf{x})\rangle$*.*
- *A tuple* $\mathbf{d} \in \mathbb{Z}_N^m$*.*

Output:

- *A tuple* $\mathbf{t} \in \mathbb{Z}_N^n$*.*

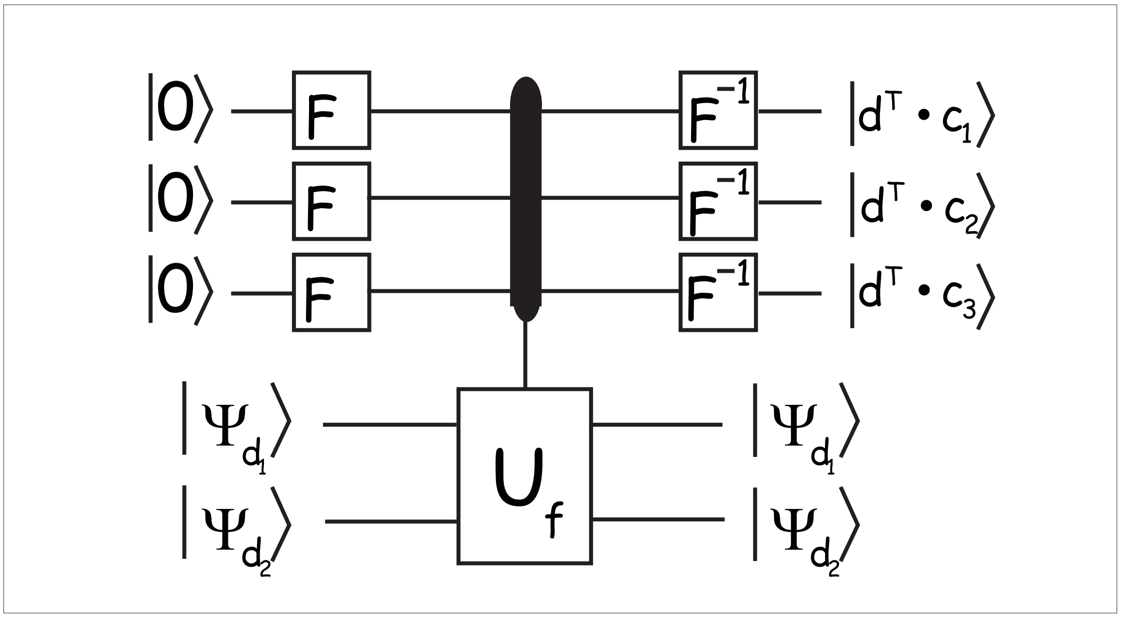

FIGURE 2.9. The values $\mathbf{d}^T \cdot \mathbf{c_j}$ are encoded in the control register and deciphered by the final F^{-1} transformations. A global phase of $e^{\mathbf{d}^T \cdot \mathbf{b}}$ is also present, but not illustrated in the diagram.

Complexity:

- *1 application of U_f.*
- *n applications of $QFT(N)$ and $n+m$ applications of $QFT(N)^{-1}$.*

Procedure:

1. *Prepare $n+m$ registers in the state*

$$|\,0\rangle\,|\,0\rangle \ldots |\,0\rangle\,|\,\Psi_{d_1}\rangle\,|\,\Psi_{d_2}\rangle \ldots |\,\Psi_{d_m}\rangle$$

2. *Apply $QFT(N) \times QFT(N) \times \cdots \times QFT(N)$ to the control registers.*
3. *Apply U_f.*
4. *Apply $QFT(N)^{-1} \times QFT(N)^{-1} \times \cdots \times QFT(N)^{-1}$ to the control registers.*
5. *Measure the first n registers and output the value.*

PROPOSITION 50. *Algorithm 49 outputs* $\mathbf{t} = \mathbf{d}^T \cdot \mathbf{M}$. *With* m *iterations with* $\mathbf{d} = (1,0,0,\dots,0),(0,1,0,0,\dots,0),\dots,(0,0,\dots,0,1)$, *we can determine* $\mathbf{M}$.

PROOF. Step 1 requires preparing m registers in the states $|\,c_i\rangle$ and then applying $QFT(N)^{-1}$ to them. The remaining registers are initialised to $|\,0\rangle$. The second register is in an eigenstate of the operator $|\,\mathbf{y}\rangle \to |\,\mathbf{y}+\mathbf{z}\rangle$ with eigenvalue $e^{2\pi i \mathbf{d}^T\cdot\mathbf{z}}$. Thus after step 3, which applies the operator $|\,\mathbf{y}\rangle \to |\,\mathbf{y}+\mathbf{f}(\mathbf{x})\rangle$ when the first register is in the state $|\,x\rangle$, the first register is then in the state

$$\begin{aligned}
&\sum_{x\in\mathbb{Z}_N^n} e^{2\pi i\frac{\mathbf{d}^T\cdot\mathbf{f}(\mathbf{x})}{N}}\,|\,\mathbf{x}\rangle \\
=\;& e^{2\pi i\frac{\mathbf{d}^T\cdot\mathbf{b}}{N}} \sum_{x\in\mathbb{Z}_N^n} e^{2\pi i\frac{\mathbf{d}^T\cdot\mathbf{Mx}}{N}}\,|\,\mathbf{x}\rangle \\
=\;& e^{2\pi i\frac{\mathbf{d}^T\cdot\mathbf{b}}{N}} \left(\sum_{x_1\in\mathbb{Z}_N} e^{2\pi i\frac{\mathbf{d}^T\cdot\mathbf{c_1}x_1}{N}}\,|\,\mathbf{x_1}\rangle\right)\cdots\left(\sum_{x_n\in\mathbb{Z}_N} e^{2\pi i\frac{\mathbf{d}^T\cdot\mathbf{c_n}x_n}{N}}\,|\,\mathbf{x_n}\rangle\right)
\end{aligned}$$

and thus after step 4 we have

$$\begin{aligned}
& e^{2\pi i\frac{\mathbf{d}^T\cdot\mathbf{b}}{N}}\,\left|\,\mathbf{d}^T\cdot\mathbf{c_1}\right\rangle\left|\,\mathbf{d}^T\cdot\mathbf{c_2}\right\rangle\cdots\left|\,\mathbf{d}^T\cdot\mathbf{c_n}\right\rangle \\
=\;& e^{\frac{2\pi i\mathbf{d}^T\cdot\mathbf{b}}{N}}\,|\,t_1\rangle\,|\,t_2\rangle\dots|\,t_n\rangle
\end{aligned}$$

where $\mathbf{t} = \mathbf{d}^T\cdot\mathbf{M}$. If $\mathbf{d} = \mathbf{e_j}$ then $\mathbf{d}^T\cdot\mathbf{M} = \mathbf{c_j}$, so we can clearly determine M using only n applications of U_f.

$\square$

Algorithm 49 appears in [**CEMM98**] for $N = 2$, and a similar algorithm was noted independently in [**Høy97**] but for homomorphisms acting on $\mathbb{Z}_{N_1}\times\ldots\times\mathbb{Z}_{N_k}$ for arbitrary N_j (that is without the shift component b).

However, unlike the case with the hidden subgroup problem, I can think of no practical examples where we have an efficient way of computing f but do not already know M.

PROBLEM 51. *Find an example where we know how to evaluate a function of the form* $f : \mathbf{x} \to \mathbf{M}\mathbf{x} + \mathbf{b}$ *but do not already effectively possess a description of* M.

[**HR90**]

CHAPTER 3

Limitations of Quantum Computers

What can quantum computers not do? Like any 'classical' computer, they are subject to the laws of physics. With a computational task expressed in more physical terms, it might become easier to determine the limitations of any computer in performing the task.

We start this chapter by defining some complexity classes and other relevant notions, like the classical classes P, BPP and NP, the quantum class BQP, and notions like NP-complete and NP-hard. By *classical computer* we mean any device which is polynomially equivalent to a universal Turing machine, and a *classical algorithm* is one that runs on a classical computer.

Finding non-trivial lower bounds on the computational complexity of computational tasks have proved very difficult. We hope that this more general framework provided by quantum computation will help find non-trivial lower bounds and some new relationships between computational complexity classes. For example, suppose our physical intuition inspires a proof that a quantum computer cannot solve an NP-complete problem. This will not only prove that NP does not contain BQP, but will have corollaries like $P \neq NP$. This is a rather ambitious task of course. It seems quite difficult to prove non-trivial lower bounds in the quantum setting as well.

Section 3.2 describes the *black-box* model of computation, in which we have proved some lower bounds and relationships between various complexity measures (sections 3.4, 3.5). The main ingredient is the relationship between the amplitudes of states in a quantum network and polynomials (section 3.3).

3.1. What is a complexity class?

As we discussed in the introduction, *complexity* refers to some measure of the resources required to solve a problem, and we will restrict attention to *decision* problems. Decision problems can be treated as the problem of recognising elements of a *language*. To define a language, we first fix an alphabet, say $\Sigma = \{0, 1\}$. A language $L \subseteq \Sigma^*$ (the set of finite strings over the alphabet Σ) is a collection of strings. An algorithm solves the language recognition problem for L if it accepts (by outputting 1) any string $x \in L$ and rejects (by outputting 0) any string $x \notin L$. For example, let the string x represent an integer, and consider the language $COMPOSITE$ which contains all representations of composite integers. The number x has non-trivial factors if and only if $x \in COMPOSITE$. So $7 \notin COMPOSITE$ and $6 \in COMPOSITE$. For another example, let the string x represent a graph, and we have $x \in 3\text{-}COLOURABLE$ if and only if there is a way of assigning one of three colours to the vertices of x so that no adjacent vertices are coloured the same. Note that both of these language recognition problems have the property that if $x \in L$, then I can prove it to you efficiently (though finding the proof might be very hard). For example, as illustrated in figure 3.1, we know $110111 \in 3\text{-}COLOURABLE$ since $CHECK\_3\_COLOURING(110111, RBRG) = 1$, where $CHECK\_3\_COLOURING(x, y)$ is an algorithm which verifies that y is a proper colouring of x. It is easy to implement $CHECK\_3\_COLOURING$ to run in polynomial time. This property of being checkable in polynomial time on a classical computer defines a whole class of problems. More specifically, following [**MR95**], DEFINITION 52. *The class NP (non-deterministic polynomial time) consists of all languages L that have a polynomial time classical algorithm A such that for any input $x \in \Sigma^*$*

- $x \in L \Rightarrow$ *there exists a* $y \in \Sigma^*$ *such that* $A(x, y)$ *accepts, (*$|y|$ *is bounded by a polynomial in* $|x|$*).*

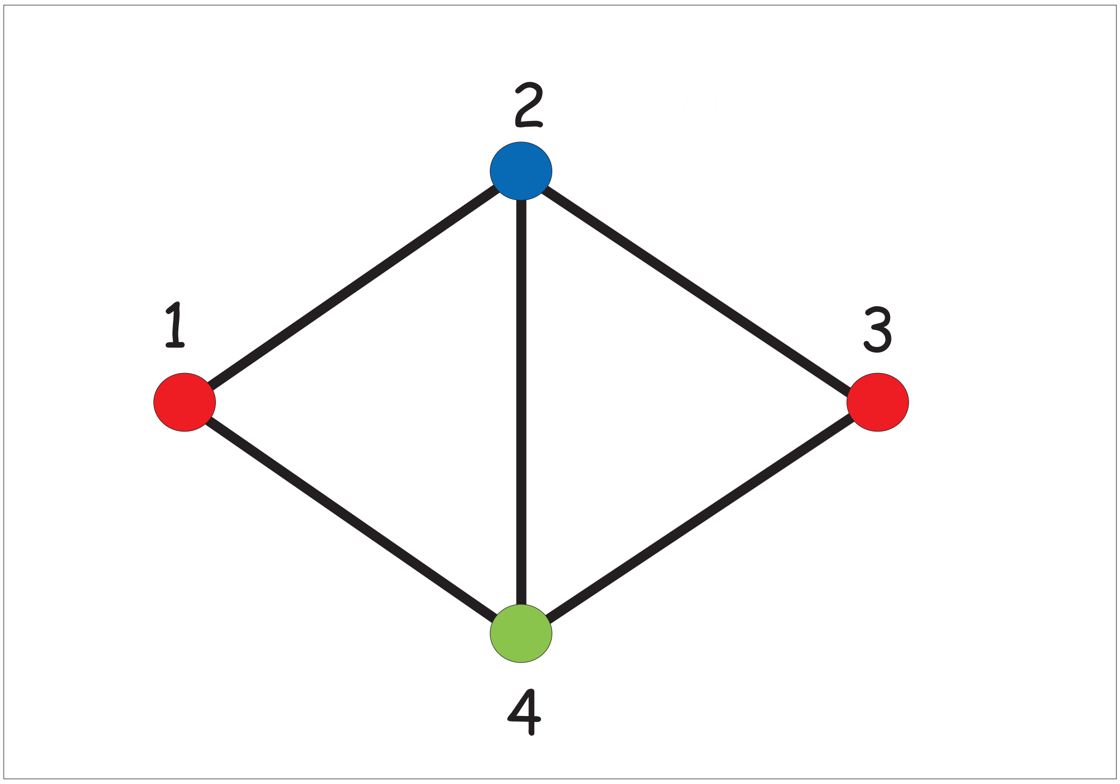

FIGURE 3.1. There are 2^6 four-vertex graphs, in a 1-1 correspondence with 6-bit strings that tell us which of the pairs of vertices $\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}$ are connected. The graph (101111) is illustrated here, with the colouring $RBRG$, one of the *proper* colourings (that is, a colouring were no connected vertices are coloured by the same colour).

- *$x \notin L \Rightarrow$ for all $y \in \Sigma^*$, $A(x, y)$ rejects.*

We can similarly define the complementary class to NP, namely co-NP, to be all the languages L such that $\Sigma^* \backslash L \in NP$.

Therefore 3-$COLOURABLE$ and $COMPOSITE$ are in NP while NOT-3-$COLOURABLE$ and $PRIME$ are in co-NP. In fact $COMPOSITE$ and $PRIME$ are in both NP and co-NP, denoted $NP \cap$ co-NP (proved in [**Pra75**]). One of the biggest open problems in theoretical computer science is whether or not every problem in NP can be solved in polynomial time.

DEFINITION 53. *The class P consists of all languages L that have a polynomial time classical algorithm A such that for any input $x \in \Sigma^*$,*

- *$x \in L \Leftrightarrow A(x)$ accepts.*

Most people believe that $P \neq NP$. Furthermore, most people believe that even probabilistic algorithms with some chance of error will not be able to decide all NP problems in polynomial time. The class of languages that can be solved with a classical computer in polynomial time with a small probability of error is called BPP.

DEFINITION 54. *The class BPP (bounded-error probabilistic polynomial time) consists of all languages L that have a randomised classical algorithm A running in worst-case polynomial time such that for any input $x \in \Sigma^*$,*

- *$x \in L \Rightarrow Pr[A(x)$ accepts $] \geq \frac{2}{3}$.*
- *$x \notin L \Rightarrow Pr[A(x)$ accepts $] \leq \frac{1}{3}$.*

By repeating this algorithm n times and taking the majority answer, we get the correct result with probability at least $1 - \epsilon^n$ for some ϵ, $0 < \epsilon < 1$ (by the Chernoff bound; e.g. [**MR95**]). The problems in BPP are held to be the tractable ones on a classical computer. Similarly, the problems in the quantum class BQP are widely held to be the tractable ones on a quantum computer.

DEFINITION 55. *The class BQP (bounded-error quantum polynomial time) consists of all languages L that have a quantum algorithm A running in worst-case polynomial time such that for any input* $x \in \Sigma^*$,

- $x \in L \Rightarrow Pr[A(x) \text{ accepts }] \geq \frac{2}{3}$.
- $x \notin L \Rightarrow Pr[A(x) \text{ accepts }] \leq \frac{1}{3}$.

Two important notions are those of being "NP-complete" and "NP-hard". Consider the problem where the inputs x encode (classical) acyclic circuits with one output bit.

DEFINITION 56. *The language $CIRCUIT\_SAT$ consists of all strings x which encode classical acyclic circuits (using some reasonable encoding as discussed in* [**Wel88, MR95**]*) for which there exists an input string y to the circuit x for which the circuit outputs a* 1.

The problem $CIRCUIT\_SAT$ is in NP since there is an algorithm that runs in time polynomial in $|x|$ that implements the circuit to verify that the circuit described by x outputs 1 on input y. Further, it can be shown [**Coo71, Lev73, GJ79**] that for any language L in NP, we can efficiently transform the polynomial time algorithm $A(x, y)$ that certifies membership in L into an acyclic circuit x' (of size polynomial in $|x|$) with the property that there exists a y such that $A(x, y) = 1$ if and only if there exists a y' such that x' outputs 1 on input y'. In other words, if we can solve $CIRCUIT\_SAT$ in polynomial time, then we can solve any NP decision problem in polynomial time. Such a language is called NP-complete.

Definition 57. *A language L is NP-*hard *if for any language $L' \in NP$ there is a polynomial time algorithm that for any input $x \in \Sigma^*$ outputs y satisfying $y \in L \Leftrightarrow x \in L'$.*

Definition 58. *A language L is NP-*complete *if L is NP-hard and L is in NP.*

In general, a problem (perhaps not even a decision problem) is called NP-hard if a polynomial time algorithm for solving that problem implies the existence of a polynomial time algorithm for deciding some NP-hard language L.

It has been shown that 3-$COLOURABLE$ is NP-complete [**GJS76**], and thus being able to solve this problem in polynomial time means we can solve every NP problem in polynomial time. See [**GJ79**] for many other examples of NP-complete problems. Note that $FACTOR$ (which decides if an integer x has a non-trivial factor less than y), $COMPOSITE$, and $PRIME$ are not believed to be NP-complete.

If $P \neq NP$, then for any NP-complete language L with corresponding polynomial time classical algorithm $A(x, y)$ and a polynomial time classical algorithm $B(x)$ that seeks to find y such that $A(x, y) = 1$, there must be some $x \in L$ for which $B(x)$ does not succeed. No proof is known however. In the next section, we discuss a related problem in the *black-box* model of computation.

3.2. Black-boxes

In the previous chapter, section 2.7.1, we discussed the problem of finding, for a given function $f : \{1, 2..., N\} \rightarrow \{0, 1\}$ a solution to $f(y) = 1$. For specific f it is usually quite hard to find a lower bound on the difficulty of this problem. For example, in the previous section we defined the class NP. Consider any $L \in NP$ with the polynomial-time checking algorithm $A(x, y)$ satisfying $A(x, y) = 1$ if and only if $x \in L$. If we let $f_x(y) = A(x, y)$, then the problem of deciding L is equivalent to deciding if there exists a y satisfying $f_x(y) = 1$. We mentioned earlier, that for this family of functions f it is widely believed that this problem cannot be solved in time polynomial in $|x|$.

For any NP-complete language L, say 3-$COLOURABLE$, for worst case input x, there is no known algorithm which works more than polynomially better than simply trying the colourings $y = c_1c_2 \dots c_n$ in order to see if they are proper colourings, that is if $CHECK\_3\_COLOURING(x, y) = 1$. In this case we are only using f_x, where $f_x(c)$ is the output of $CHECK\_3\_COLOURING(x, c)$, as a black-box in order to decide if there exists a c such that $f_x(c) = 1$.

When we use $f = f_x$ as a black-box and ignore any information we have about it (other than the fact that it maps $\{1, 2, \dots, N\} \to \{0, 1\}$), we could just as well define $X_j = f(j)$ and rephrase the problem as follows. There is a binary string $\mathbf{X} = X_1, X_2, ..., X_N$ and we are given a black-box which takes as input an index j and outputs X_j. In the quantum setting, we will assume that we have a black-box operator O_X which maps $|j\rangle\,|b\rangle \to |j\rangle\,|b \oplus X_j\rangle$. The question addressed by Grover [**Gro96**] was to find an index j where $X_j = 1$. We can also consider the decision problem: does there exist a j such that $X_j = 1$?

It was first shown in [**BBBV97**] that any quantum algorithm using the function $O_X : |j\rangle\,|b\rangle \to |j\rangle\,|b \oplus X_j\rangle$ as a black-box requires $\Omega(\sqrt{N})$ applications of O_X in order to find a solution with probability at least $\frac{2}{3}$ for worst case X.

Another way to phrase this decision problem is to evaluate the OR function on the string $\mathbf{X}$, where $OR(\mathbf{X}) = X_1 \vee X_2 \vee \cdots \vee X_N$. In general, we can consider any $\{0, 1\}$-valued function F of the variables $X_1, X_2, \dots, X_N$. Recall that f is a function $\{1, 2, \dots, N\} \to \{0, 1\}$ satisfying $f(j) = X_j$, and $X_1, X_2, \dots, X_N$ are inputs to the function $F : \{0, 1\}^N \to \{0, 1\}$.

In the following sections, we will study the limitations of quantum computers in determining properties of a binary string $X = X_1X_2 \dots X_N$. For example, the numbers $0, 1, \dots, N-1$ could represent the 3-colourings of a graph G (so $N = 3^n$), and $X_j = 1$ if and only if j is a proper 3-colouring of G. We will only consider the limitations in the black-box model of computation.

This restriction might seem rather convenient and raises the question of how relevant these lower bounds are to a 'real' problem, where we do have some information about how X is evaluated. If we consider an X corresponding to an NP problem, then $X_j = f(j)$ for some function f which we usually know how to compute. So what is the relevance? These black-box lower bounds tell us the limitations of certain algorithmic approaches as we now illustrate.

DEFINITION 59. *The* deterministic query complexity $D(F)$ of F *is the minimum number of queries of bits of X required by a deterministic classical strategy for computing $F(X)$ (the jth index to be queried can depend on the outcome of the previous $j-1$ queries).*

The quantum equivalent of $D(F)$ is the *exact quantum query complexity $Q_E(F)$ of F.*

DEFINITION 60. *The* exact quantum query complexity $Q_E(F)$ of F *is the minimum number of black-box O_X queries required by a quantum algorithm which correctly finds $F(X)$ with probability* 1 *for every string X.*

A more practical quantity is the *2-sided error quantum query complexity $Q_2(F)$ of F.*

DEFINITION 61. *The* 2-sided error quantum query complexity $Q_2(F)$ of F *is the minimum number of black-box queries required by a quantum algorithm which,* *on any input X, outputs a $\{0,1\}$ value that has probability at least $\frac{2}{3}$ of being equal to $F(X)$.*

In [**BBC+98**] we prove the following relationship.

THEOREM 62. *If F is a Boolean function, then $D(F)^{\frac{1}{6}} \leq 4Q_2(F)$.*

What does Theorem 62 mean for the quantum complexity of computing F given an algorithm for computing $X_j = f(j)$? Suppose the best deterministic classical strategy for evaluating $F(X)$ requires in the worst case $T = D(F)$ queries of the bits of X. Theorem 62 tells us that for any quantum algorithm to evaluate F using less that $\frac{T^{\frac{1}{6}}}{4}$ steps, it must exploit additional properties of X. Further, even any non-constructive proof that such an algorithm does or does not exist must make use of the additional properties of $X = f(1)f(2)\ldots f(N)$, other than that it is a binary string of length N.

3.3. Relation between quantum networks, black-boxes and polynomials

In this section we will show how a quantum gate array which queries the string X a total of T times will have amplitudes that are polynomials of degree T in the variables $X_1, X_2, \ldots, X_N$. In the subsequent section we describe several applications of this fact.

LEMMA 63. *Let $\mathcal{N}$ be a quantum network that uses a total of m qubits and makes T queries to a black-box O_X. Then there exist complex-valued N-variate multi-linear polynomials $p_1, p_2, \ldots, p_{2^m}$, each of degree at most T, such that the final state of the network is the superposition*

$$\sum_{y=1}^{2^m-1} p_y(X)\,|\,y\rangle$$

for any black-box X.

PROOF. We can assume that $N = 2^n$. We can assume that the black-boxes are used sequentially. Let U_j denote the unitary transformation which we apply between the jth and $(j+1)$th black-box query. We can assume that we always apply O_X to the first $n+1$ qubits (incorporate any permutations of the qubits into the U_j operations). We thus have the network illustrated in figure 3.2. For the proof, it will help to consider the register in three parts: the first n qubits, the 1 output bit, and the remaining $l = m - n - 1$ ancilla bits.

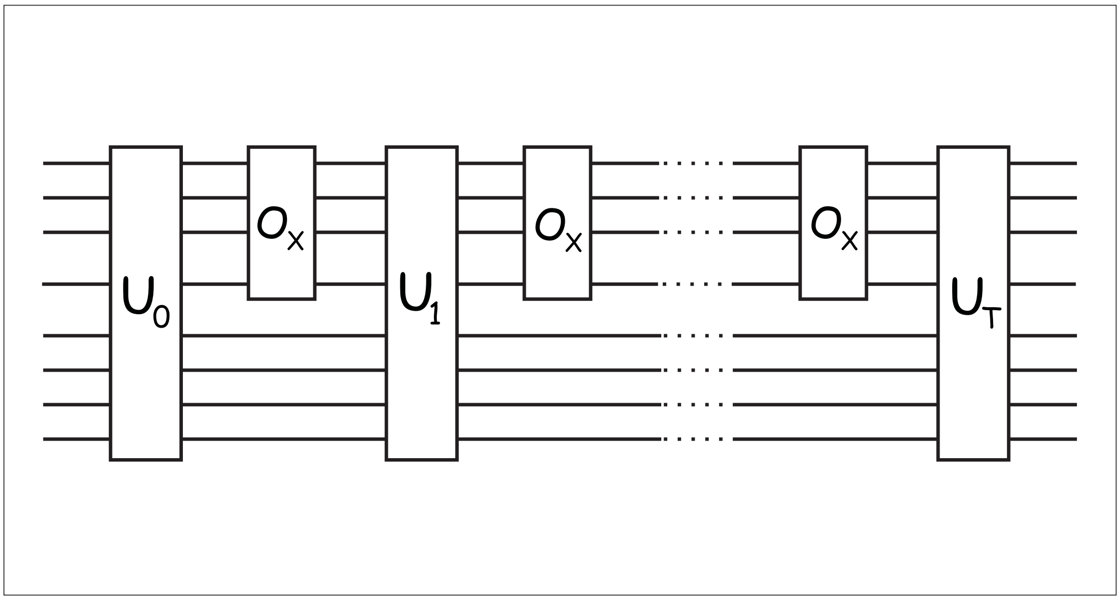

FIGURE 3.2. Without loss of generality, any network which makes T black-box queries is equivalent to a network which starts with the state $|00\ldots0\rangle$, applies a unitary operator U_0, followed by a black-box query on the first $n+1$ qubits, followed by a unitary operation U_1 and so on, with a final unitary operation U_T after the last black-box call.

Just before the first black-box application the m-qubits will be in some state

$$\sum_{j,k} \alpha_{j0k} \, |j0k\rangle + \alpha_{j1k} \, |j1k\rangle \, ,$$

where $0 \leq j < 2^n$, $0 \leq k < 2^l$, and the $\alpha_{j,b,k}$, $b \in \{0,1\}$ are independent of the string X. In other words the α_j are polynomials of degree 0 in $X_1, X_2, \ldots, X_N$. For $b \in \{0,1\}$ we also use the notation $\overline{b} = NOT(b) = 1 - b$. After the first black-box call, we have the state

$$\begin{aligned} & \sum_{j,k} \alpha_{j0k} \, |jX_jk\rangle + \alpha_{j1k} \, \left|j\overline{X_j}k\right\rangle \\ = & \sum_{j,k} [(1-X_j)\alpha_{j0k} + X_j\alpha_{j1k}] \, |j0k\rangle + [(1-X_j)\alpha_{j1k} + X_j\alpha_{j0k}] \, |j1k\rangle \, . \end{aligned}$$

Therefore the amplitudes are polynomials in the X_j of degree at most 1. The unitary operation U_1 is linear, and thus the amplitudes just after U_1 is applied are still polynomials of degree at most 1. We can easily see that, for $j \geq 0$, if just after U_{j-1} is applied the amplitudes are polynomials of degree at most $j-1$, then the jth black-box call adds at most 1 to the degree of the amplitude polynomials so they are of degree at most j. The U_j replaces the amplitude polynomials with linear combinations of amplitude polynomials and thus the degrees remain at most j. Since $x^2 = x$ for $x \in \{0,1\}$, we can assume the polynomials are multi-linear. The proof follows by a simple induction. □

We get the following corollary [**BBC**$^+$**98**].

Corollary 64. *Let $\mathcal{N}$ be a quantum network that makes T queries to a black-box X, and B be a set of basis states. Then there exists a real-valued multi-linear polynomial P of degree at most $2T$, which equals the probability of observing a state from the set B after applying the network $\mathcal{N}$ using black-box O_X.*

3.4. Applications to lower bounds

Let us start by defining three quantities, $\deg(F)$, $\widetilde{\deg}(F)$, and $bs(F)$, related to the N-variate function F. Although the function F is only defined on values of 0 and 1 it is useful to extend this function to the reals.

DEFINITION 65. *An N-variate polynomial $p : R^N \to R$* represents *F if $p(X) = F(X)$ for all $X \in \{0,1\}^N$.*

LEMMA 66. *Every N-variate function $F : \{X_1, \dots, X_N\} \to \{0,1\}$, has a unique multi-linear polynomial $p : \mathbb{R}^N \to \mathbb{R}$ which represents it.*

PROOF. The existence of a representing polynomial is easy: let

$$p(X) = \sum_{Y \in \{0,1\}^N} F(Y) \prod_{k=1}^{N} [1 - (Y_k - X_k)^2].$$

To prove uniqueness, let us assume that $p_1(X) = p_2(X)$ for all $X \in \{0,1\}^N$. Then $p(X) = p_1(X) - p_2(X)$ is a polynomial that represents the zero function. Assume that $p(X)$ is not the zero polynomial and without loss of generality, let $\alpha X_1 X_2 \dots X_k$ be a term of minimum degree, for some $\alpha \neq 0$. Then the string X with $X_1 = X_2 = \ldots = X_k = 1$ and the remaining X_j all 0 has $p(X) = \alpha \neq 0$. This contradiction implies that $p(X)$ is indeed the zero polynomial and $p_1 = p_2$. $\square$

The degree of such a p is a useful measure of the complexity of F.

DEFINITION 67. *The degree of the polynomial p which represents F is denoted $\deg(F)$.*

For example, the OR function is represented by the polynomial $1 - \prod_{j=1}^{N} (1 - X_j)$ which has degree N. Thus $\deg(OR) = N$.

In practice, it would suffice to have a polynomial p which approximates F at every $X \in \{0,1\}^N$. For example $OR(X_1, X_2) \approx \frac{2}{3}(X_1 + X_2)$.

DEFINITION 68. *An N-variate polynomial $p : R^N \to R$* approximates *F if $|p(X) - F(X)| \leq \frac{1}{3}$ for all $X \in \{0,1\}^N$.*

Again, the minimum degree of such a polynomial p is a useful measure of the complexity of F.

DEFINITION 69. *The minimum degree of a p approximating F is denoted $\widetilde{\deg}(F)$.*

Lastly, intuitively it seems obvious that functions which are very sensitive to changes of the values of almost any of the bits in the string X will require us to probe more bits of X than functions which are relatively indifferent to such changes. One way of rigorously capturing this concept of sensitivity is by the notion of the *block sensitivity* of F.

DEFINITION 70. *Let $F : \{0,1\}^N \to \{0,1\}$ be a function, $X \in \{0,1\}^N$, and $B \subseteq \{1, 2, \ldots, N\}$ be a set of indices.*

Let X^B denote the vector obtained from X by flipping the variables in B.

The function F is sensitive to B on X *if $f(X) \neq f(X^B)$.*

The block sensitivity $bs_X(F)$ of F on X *is the maximum number t for which there exist t disjoint sets of indices $B_1, \ldots, B_t$, such that F is sensitive to each B_i on X.*

The block sensitivity $bs(F)$ of F *is the maximum of $bs_X(F)$ over all $X \in \{0,1\}^N$.*

We get the following three theorems relating the quantum query complexities $Q_E(F)$ and $Q_2(F)$ to $\deg(F)$, $\widetilde{\deg}(F)$, and $bs(F)$.

THEOREM 71. *If F is a Boolean function, then $Q_E(F) \geq \frac{\deg(F)}{2}$.*

PROOF. Consider the result of a quantum algorithm for evaluating F exactly using $Q_E(F)$ queries. By corollary 64, the probability of observing 1 is $p_1(X)$, a polynomial of degree at most $2Q_E(F)$. We will observe 1 if and only if $F(X) = 1$. In other words $p_1(X) = F(X)$ for all $X \in \{0,1\}^N$. This implies that $2Q_E(F) \geq \deg(F)$. $\square$

THEOREM 72. *If F is a Boolean function, then $Q_2(F) \geq \frac{\widetilde{\deg}(F)}{2}$.*

PROOF. Consider the result of a quantum algorithm for evaluating F approximately using $Q_2(F)$ queries. By corollary 64, the probability of observing 1 is $p_1(X)$, a polynomial of degree at most $2Q_E(F)$. If $F(X) = 1$, then $p_1(X) \geq \frac{2}{3}$. Similarly if $F(X) = 0$ then $1 - p_1(X) \geq \frac{2}{3}$. In other words $|p_1(X) - F(X)| \leq \frac{1}{3}$ for all $X \in \{0,1\}^N$, which means p_1 approximates F. This implies that $2Q_E(F) \geq \widetilde{deg}(F)$. $\square$

This last theorem is less obvious.

THEOREM 73. *If F is a Boolean function, then $Q_E(F) \geq \sqrt{\frac{bs(F)}{8}}$ and $Q_2(F) \geq \sqrt{\frac{bs(F)}{16}}$.*

The proof [**BBC+98**] makes clever use of the following theorem (from [**EZ64, RC66**]).

THEOREM 74. *Let $p : R \to R$ be a polynomial such that $b_1 \leq p(i) \leq b_2$ for every integer i from 0 to N, and $|p'(x)| \geq c$ for some real x, $0 \leq x \leq N$. Then* $\deg(p) \geq \sqrt{\frac{cN}{c+b_2+b_1}}$.

The notion of block sensitivity extends nicely to the study of *partial* Boolean functions. The functions F we have considered so far are called *total* functions since they are defined on all of $\{0,1\}^N$. Partial functions are only defined on a subset $S \subset \{0,1\}^N$. For example in [**DJ92**] Deutsch and Jozsa restricted the domain to the set S of strings which were either all 0 or all 1 or which had an equal number of 0s and 1s. They defined the function F to be 1 if and only if $X_1 = X_2 = \ldots = X_N$ (i.e. if X is constant) and 0 otherwise (i.e. if X is balanced). The block sensitivity of a partial function is simply the minimum $bs_X(F)$ over all $X \in S$ and Theorem 73 holds for partial functions as well. The Deutsch-Jozsa function has block sensitivity 2 giving a lower bound of 1, which is tight since the algorithm in [**CEMM98**] solves the problem with only one query (see Algorithm 45).

3.5. Relating quantum and deterministic query complexity

Using the relationship between the quantum query complexity $Q_2(F)$ and $bs(F)$ and the result $D(F) \le bs(F)^3$ (before this the best known result was $D(F) \le bs(F)^4$ [**Nis91**]), we are able to prove the relationship $2^{12}Q_2(F)^6 \ge D(F)$ (Theorem 62). The proof requires the introduction of another property of a function F, namely its *certificate complexity.*

Definition 75. *Let* $F : \{0,1\}^N \to \{0,1\}$ *be a function. A* 1-certificate *is an assignment* $C : S \to \{0,1\}$ *of values to some subset* S *of the* N *variables, such that* $f(X) = 1$ *whenever* X *is consistent with* C*. The* size *of* C *is* $|S|$*. We similarly define a* 0-certificate*. The* certificate complexity $C_X(F)$ of F on X *is the size of a smallest* $f(X)$*-certificate that agrees with* X*. The* certificate complexity $C(F)$ of F *is the maximum over all* X *of* $C_X(F)$*. The* 1-certificate complexity $C^{(1)}(F)$ of F *is the maximum of* $C_X(F)$ *over all* X *with* $f(X) = 1$.

Roughly speaking, a 0- or 1-certificate is a subset of the string which guarantees or certifies the value of the function on the whole string.

EXAMPLE 76. *Consider the OR function, which satisfies $OR(X) = 0$ if and only if $X_j = 0$ for all j. Any substring $X_j = 1$ is enough to guarantee that $OR(X) = 1$. Thus the certificate complexity $C_X(OR)$ for any non-zero string X is 1. However the certificate complexity of the all-zero string is N, since every single bit of a string X must be 0 for $OR(X) = 1$ to hold. Thus the certificate complexity $C^{(0)}(OR) = N$. The maximum of all the $C_X(OR)$ gives us the certificate complexity of (OR), namely, $C(OR) = N$. The 1-certificate complexity is $C^{(1)}(OR) = 1$ and the 0-certificate complexity is $C^{(0)}(OR) = N$.*

The following lemma is a result of Nisan [**Nis91**].

LEMMA 77. $C^{(1)}(F) \leq C(F) \leq bs(F)^2$.

The next lemma is proved in [**BBC$^+$98**].

LEMMA 78. $D(F) \leq C^{(1)}(F) bs(F)$.

Combining these two lemmas gives $D(F) \leq bs(F)^3$, and adding in Theorem 73 gives Theorem 62.

3.6. Some examples and applications

We have related $Q_2(F)$ and $Q_E(F)$ to the quantities $\deg(F)$, $\widetilde{\deg}(F)$, and $bs(F)$. How do we evaluate $\deg(F)$, $\widetilde{\deg}(F)$ and $bs(F)$? By Lemma 66 we can find $\deg(F)$ by finding a representing polynomial. We can lower bound block sensitivity $bs(F)$ by finding particular strings that have high block sensitivity. Bounding $\widetilde{\deg(F)}$ requires a little more work.

Let us restrict attention to *symmetric* polynomials, that is those for which permuting the variables X_j does not affect the value of the function. Such functions F are thus reducible to functions $q : \{0, 1, \dots, N\} \to \mathbb{R}$ of the Hamming weight of X. Thus any such symmetric function F effectively partitions the integers $\{0, 1, 2, \ldots, N\}$, into two disjoint sets, and the task at hand in determining $F(X)$ is to decide which set the Hamming weight of X lies in. A related quantity is the function $\Gamma(F) = \min\{|2k - N + 1| : q(k) \neq q(k+1) \text{ and } 0 \leq k \leq N - 1\}$. The value of $\Gamma(F)$ is small if $q(k)$ changes for k close to $\frac{N}{2}$. We make use of the following theorem of Paturi [**Pat92**].

THEOREM 79. *(Paturi) If F is a non-constant symmetric Boolean function on $\{0, 1\}^N$, then $\widetilde{deg}(F) \in \Theta(\sqrt{N(N - \Gamma(F))})$.*

Some examples:

- **OR**

  The OR function maps all non-zero strings to 1 and the all-zero string to 0. As a polynomial we have $OR(X) = 1 - (1 - X_1)(1 - X_2)\ldots(1 - X_N)$, and thus $\deg(OR) = N$.

  Thus by Theorem 71 $Q_E(OR) \geq \frac{N}{2}$. This can in fact be tightened to $Q_E(OR) = N$ [**BBC$^+$98**].

  It is easy to see that the block sensitivity of the OR function is N, thus Theorem 73 gives us $Q_2(OR) \geq \frac{1}{4}\sqrt{N}$. Another way to prove an $\Omega(\sqrt{N})$ lower bound is to make use of Theorems 79 and 72, and of the fact that $\Gamma(OR) = N - 1$.

  Note that Grover's algorithm or the quantum counting algorithms will evaluate the OR function with probability at least $\frac{2}{3}$ of being correct using $O(\sqrt{N})$ queries, so our lower bound is tight up to a constant factor. Such a lower bound was first given in [**BBBV97**]. Several subsequent proofs have also been given.

- **PARITY**

The PARITY function maps all strings with an even number of 1s to 0 and all strings with an odd number of 1s to 1. The PARITY function can be compactly described as $\frac{1-\Pi_{j=1}^{N}(-1)^{X_j}}{2}$. By substituting $(-1)^{X_j} = (1-2X_j)$ we see that $\deg(PARITY) = N$, and thus $Q_E(PARITY) \geq \frac{N}{2}$. This fact was shown independently in [**FGGS98**] and [**BBC$^+$98**]. This is in fact tight due to the following algorithm (we assume N is even, otherwise set $X_{N+1} = 0$ and apply this algorithm).

ALGORITHM 80.

Input:

– *A black-box which evaluates U_f.*

Output:

– $PARITY(X)$

Complexity:

– $\frac{N}{2}$ *applications of* U_f

– $O(N)$ *other elementary operations.*

Procedure:

1. *Use the Deutsch algorithm (section 2.1) to compute $X_{2j-1} \oplus X_{2j}$ for $j = 1, 2, \ldots, N/2$. This uses $N/2$ queries.*
2. *Compute* $PARITY(X) = (X_1 \oplus X_2) \oplus (X_3 \oplus X_4) \oplus \cdots \oplus (X_{N-1} \oplus X_N)$.

Approximating the $PARITY$ function however is not much easier than determining it exactly, since $\widetilde{\deg}(PARITY) \in \Theta(N)$. This follows from Theorem 79 and the fact that $\Gamma(OR) \leq 1$. However we will now show that $\widetilde{deg}(PARITY) = N$ [**MP95, BBC$^+$98**]. We will need the following definition.

DEFINITION 81. *Let* $p : \mathbb{R}^N \to \mathbb{R}$ *be a polynomial. For any* $\pi \in S_N$*, let* $\pi(X) = \pi(X_1)\pi(X_2)\ldots\pi(X_N)$ *and define the polynomial*

$$p^{sym}(X) = \frac{\sum_{\pi \in S_N} p(\pi(X))}{N!}.$$

THEOREM 82. $\widetilde{\deg}(PARITY) = N$

PROOF. Note that if p represents or approximates a function F then so does p^{sym}. Since p^{sym} is symmetric, it can be written as

$$a_0 + a_1V_1 + a_2V_2 + \ldots + a_dV_d$$

where d is the degree of p and V_j is the sum of all multi-linear polynomials of degree j in $X_1, X_2, \ldots, X_N$. Note that when we plug in values for the variables X_i, the value of V_j will be $\binom{|X|}{j}$, where $|X|$ is the Hamming weight of the string X. Therefore the value

$$p^{sym}(X) = a_0 + a_1\binom{|X|}{1} + a_2\binom{|X|}{2} + \ldots a_n\binom{|X|}{d}.$$

Define the polynomial $q : \mathbb{R} \to \mathbb{R}$ by $q(x) = a_0 + a_1\binom{x}{1} + a_2\binom{x}{2} + \ldots a_d\binom{x}{d}$. Note that $p^{sym}(X) = q(|X|)$ and the degree of q in x is at most the degree of p^{sym} in $X_1, X_2, \ldots, X_N$. Therefore if p approximates F, the $\deg(q) \leq \deg(p^{sym}) = \widetilde{deg}(F)$.

Note that for the PARITY function, $q(x) = 0$ for $x = 0, 2, 4, \ldots$ and $q(x) = 1$ for $x = 1, 3, \ldots$. So the function $q(x) - \frac{1}{2}$ has N roots and thus $N \leq \deg(q) \leq \widetilde{deg}(F)$. Therefore $\widetilde{\deg}(PARITY) = \deg(PARITY) = N$. $\square$

COROLLARY 83. $Q_2(PARITY) = Q_E(PARITY) = \lceil \frac{N}{2} \rceil$.

- **THRESHOLD$_\mathbf{M}$**

The $THRESHOLD_M$ function maps strings X with at least M 1s to 1 and the others to 0. The $MAJORITY$ function is a special case with $M = \lceil \frac{N}{2} \rceil$, OR is a special case with $M = 1$ and AND is a special case with $M = N$. For non-trivial M, we have $\deg(THRESHOLD_M) = N$ and thus $Q_E(THRESHOLD_M) \geq \lceil \frac{N}{2} \rceil$. We know that for special cases $M = 1$ and $M = N$, $Q_E(THRESHOLD_M) = N$ and for $M = \lceil \frac{N}{2} \rceil$, algorithms [**HKM98**] have been found which use at most $N + 1 - e(N)$ queries, where $e(N)$ is the number of 1s in the binary expansion of N. Since $\Gamma(THRESHOLD_M) = |2M - N + 1|$, by theorem 79 we know that for $M > 0$, $Q_2(THRESHOLD_M) \in \Omega(\sqrt{N(N - \Gamma(THRESHOLD_M))}) = \Omega(\sqrt{M(N - M + 1)})$. We rewrote the lower bound in a way to coincide with the upper bound shown in section 2.7 (since exact counting will solve the $THRESHOLD$ problem). Thus we have a tight (up to a constant factor) algorithm for the $THRESHOLD_M$ function, $Q_2(THRESHOLD_M) \in \Theta(\sqrt{M(N - M + 1)})$ (this also holds for $M = 0$).

In [**Dam98**] van Dam shows that we can determine the entire string X with probability greater than $\frac{2}{3}$ using only $\frac{N}{2} + \sqrt{N}$ queries, in which case we can correctly evaluate any function $F(X)$.

The following table summarises the complexities of evaluating certain functions in terms of the number of black-box calls O_X.

3.7. Controlled-O_X and some open problems

The type of algorithms we considered applied our black-box O_X exactly T times in our quantum network. What if we wish only to conditionally apply the O_X? How can we implement a controlled-O_X if we are not given a break-down of O_X into elementary gates?

	$Q_2(F)$	$Q_E(F)$
OR, AND	$\Theta(\sqrt{N})$	N
$PARITY$	$N/2$	$N/2$
$THRESHOLD_M$	$\Theta(\sqrt{M(N-M+1)})$	$\Theta(N)$

TABLE 3.2. This table gives some bounded-error and exact quantum query complexities.

Note that if the target qubit of O_X is in the state $|0\rangle + |1\rangle$, the O_X has no effect. We can thus implement a controlled-O_X by implementing a 0-controlled-SWAP (i.e. SWAP the qubits when the control bit is in state $|0\rangle$) between the target qubit and an auxiliary qubit set to the state $|0\rangle + |1\rangle$. The qubits should be swapped back immediately after then O_X is applied (see figure 3.3).

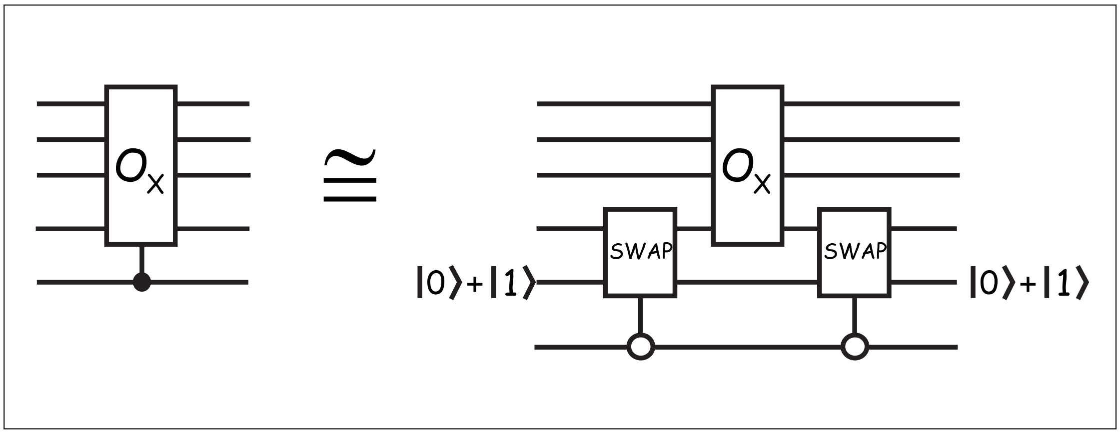

FIGURE 3.3. A network for implementing a controlled black-box operation.

I list a few open problems.

We know that $Q_E(F) \in \Omega(deg(F))$ (Theorem 71). Conversely, given a multilinear polynomial P of degree T, which satisfies $p(X) \in \{0,1\}$ for all $X \in \{0,1\}^N$, is there a quantum algorithm which makes $O(T)$ black-box queries and outputs $|1\rangle$ with probability $P(X)$?

QUESTION 84. *Is* $Q_E(F) \in \Theta(deg(F))$*?*

QUESTION 85. *Is* $Q_2(F) \in \Theta(\widetilde{deg}(F))$*?*

Nisan and Szegedy [**NS94**] have shown that

$$bs(F) \leq 6\widetilde{deg}(F)^2.$$

It thus follows (Corollary 5.5 of [**BBC$^+$98**]) that

$$Q_2(F) \leq D(F) \leq bs(F)^3 \leq 216\widetilde{deg}(F)^6.$$

So question 85 is rather ambitious and an answer to the following question would already be interesting.

QUESTION 86. *Is* $Q_2(F) \in O(\widetilde{deg}(F)^k)$ *for any* $k < 6$.

The difficulty with proving these claims by a simple recursion is related to the difficulty of computing functions *cleanly* (a term borrowed from Buhrman and de Wolf). For example, early attempts at using the Deutsch algorithm, which computes $X_0 \oplus X_1$ using only one query suffered the fatal flaw that the state created is not just $|X_0 \oplus X_1\rangle$ but $(-1)^{X_0}|X_0 \oplus X_1\rangle$. That is, the extra phase information $(-1)^{X_0}$ complicates any recursive use of this method. It seems very tricky indeed to compute the value of the desired function and nothing else without cumbersome 'uncomputing'. For example, try to compute the AND of two bits X_0 and X_1 using only two queries. The best I know of is an algorithm which uses 3 queries.

ALGORITHM 87.

- *Start with the state* $|\,0\rangle\,|\,0\rangle\,|\,0\rangle$.
- *Apply a query using the first two qubits. This produces* $|\,0\rangle\,|\,X_0\rangle\,|\,0\rangle$.
- *Apply a query using the last two qubits. This produces the state* $|\,0\rangle\,|\,X_0\rangle\,|\,X_{X_0}\rangle$.
- *Again apply a query using the first two qubits. This uncomputes the value of* X_0 *to produce* $|\,0\rangle\,|\,0\rangle\,|\,X_{X_0}\rangle$.

Observe that $X_{X_0} = 1$ if and only if $X_0 = X_1 = 1$. This algorithm can be easily generalised to *cleanly* compute the AND of N variables using $2N-1$ queries. Note that this algorithm is also a reversible *classical* one. I know of no better way that does so classically reversibly or quantumly *cleanly*.

QUESTION 88. *What is the black-box complexity of* cleanly *computing the AND of N variables?*

At present we only have an example of at most a quadratic separation between $D(F)$ and $Q_2(F)$, so it is natural to ask the following.

QUESTION 89. *Show that* $D(F) \in O(Q_2(F)^k)$ *for any* k *satisfying* $2 \leq k < 6$. *Alternatively, find a function* F *where* $Q_2(F)^k \in o(D(F))$, $2 \leq k < 6$

CHAPTER 4

Implementations

In section 4.1 we briefly describe why it is difficult to realise a large scale quantum computation and ways we hope to deal with errors once our quantum components are sufficiently reliable. In section 4.2 we describe some methods for maximising what we can do with a limited amount of quantum resources. In section 4.3 we describe how current NMR technology is used to implement quantum algorithms. Finally, in section 4.4 we describe some interesting algorithms that are suitable for implementations on small quantum computers and describe some of the first implementations of quantum algorithms.

4.1. Dealing with errors and faults

Quantum computers, like classical computers, suffer from errors and faults. Quantum information is especially susceptible to corruption by interacting with the environment. This corruption is called *decoherence*. For example, suppose we have a superposition $|\,0\rangle + |\,1\rangle$ independent of the environment in the state $|\,E\rangle$. If the environment in any way interacts with the qubit and obtains enough information to distinguish the state $|\,0\rangle$ from the state $|\,1\rangle$, it will evolve into orthogonal states $|\,E_0\rangle$ and $|\,E_1\rangle$ and the joint system is described by $|\,0\rangle\,|\,E_0\rangle + |\,1\rangle\,|\,E_1\rangle$. If we trace out the environment we see that our qubit is in the state

$$\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}.$$

If we apply a Hadamard transform to this state we still have the state

$$\frac{1}{2}\,|\,0\rangle\,\langle\,0| + \frac{1}{2}\,|\,1\rangle\,\langle\,1| = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

whereas applying the Hadamard transform to the original state

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

gives us the state

$$|\,0\rangle\,\langle\,0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

In the classical case, we have codes, in particular, linear codes (see e.g. [**Wel88**] for an introduction) that help us detect and correct errors.

A small example is the 3-qubit code which encodes 0 as 000 and 1 as 111. One-bit errors can be corrected by resetting the three bits to the value that occurs most often. If the bit flip errors occur independently at random with probability p then we get more than one error with probability $3p^2 - 2p^3$. Thus this scheme will reduce the probability that an uncorrected error corrupts our computation, provided p is small enough that $3p^2 - 2p^3 < p$ (which is true if $0 \leq p < \frac{1}{2}$). Note that the error can be detected and corrected without probing all of the bits individually. It suffices to learn the XOR of bits 1 and 2 and of bits 1 and 3. This 2-bit string is called the error *syndrome* and has the property that for all strings in the code, 000 and 111, the syndrome is 00. Corrupted strings 001 and 110 both have the same syndrome 01 which tells us that the most likely error was a bit flip in the rightmost bit, corresponding to the error vector 001. Corrupted strings 010 and 101 have syndrome 10 which identifies error vector 010 as the most likely one, and similarly strings 100 and 011 have syndrome 11 to identify the error vector

100. From the syndrome we can easily identify the most likely error and correct for that error by flipping the appropriate bit.

In general, linear codes embed the computation in a subspace of a larger space. The subspace is carefully chosen to have several properties:

- legitimate computations keep the state of the computer in the subspace
- unwanted operations (errors, the net effect between an imprecise operation and the desired operation) are likely to map the computer outside the subspace
- we can efficiently compute the most likely way a state has deviated from the computational subspace (this is done by looking at the error *syndrome*)
- we can efficiently "uncompute" these errors.

A quantum version of the 3-bit repetition code could consist of three parallel copies of the quantum register, initialised to the same state. The computation would be carried out on all three registers in parallel. An error-free computation would leave the state of the three quantum registers in the subspace that is symmetric under permutations of the three registers. This encoding, along with the corresponding error-detection (in order to suppress, but not correct, errors) is described in [**BBD$^+$97**]. A quantum version of linear codes with *error-correction* soon followed [**Sho95a, CS96, Ste96**]. There is also the problem of introducing and propagating errors during the process of error-correction. John von Neumann addressed this in the classical case [**Neu56**], and a quantum version has been developed [**Sho96, Ste97**].

The current state of the art in quantum error-correction and fault tolerant computing holds that under reasonable models of decoherence, once the quality of our elementary operations is above some *threshold*, we have efficient families of fault-tolerant error correcting codes that allow us to reliably perform computations requiring T steps using $Tpolylog(T)$ operations [**Got98, KLZ97, ABO97**].

4.2. Maximising exploitation of quantum resources

Classical computers store information in various media, including floppy diskettes, the RAM and the hard-drive. Good algorithms try to store information that is accessed very often in the RAM, since this information is most quickly accessed. Different bits of information are stored in different media at different times depending on their role in the computation. Quantum registers are expensive, fragile, and really useful only for certain tasks. Therefore they should be used with prudence. Let us assume that we have a quantum computer with n qubits, and we wish to maximise what we can do with these n qubits. We should not use these qubits to store information that could be encoded quantumly much later on in the computation, or to perform classical calculations that could have been carried out classically "on the side". We can often modify quantum algorithms to reduce the amount of quantum memory required to carry it out. In section 4.2.1 I will describe how the discrete logarithm problem can be reduced to several smaller instances reducing the necessary size of the control register. If the implementation permits, when performing eigenvalue estimation, we can reduce the control register down to a single reusable control qubit (section 4.2.2). The last 'trick' I mention is to not bother preparing qubits in a specific starting state if the algorithm does not really require it. In liquid state NMR, preparing a qubit in the state $|0\rangle$ turns out to be very difficult, and thus we can avoid this for many of the qubits in the factoring algorithm for example.

4.2.1. Classical reductions to several smaller instances: Discrete Logarithms. Suppose we have a bounded amount of quantum memory, or a bounded number of operations before decoherence reduces the probability of success to an unacceptable level. In this case, it will be useful, if possible, to break up large quantum algorithms into several instances of smaller quantum algorithms that can be run separately and then combined later on a classical computer.

For example, suppose we wish to find the discrete logarithm of b to the base $a \in G$ where a is of order M and M is composite. Let us first assume $M = pq$, where $p < q$ are coprime. If we directly apply Algorithm 18 ($Discrete\_Log(b, a)$) to find the discrete log of b to base a, we need $\lceil \log_2 2M \rceil + 1$ control qubits and $\lceil \log_2 2M \rceil + 1$ group multiplications. We can however find a logarithm modulo p and another modulo q and combine the answers classically to find the logarithm modulo M. We find the logarithm $s_1 \mod p$ of b^q to the base a^q, and the logarithm $s_2 \mod q$ of b^p to the base a^p. We then use the Chinese Remainder theorem to find $s \mod pq$ satisfying $s \equiv s_1 \mod p$ and $s \equiv s_2 \mod q$.

If we remove the restriction that p and q are coprime, say M is a prime power p^2, we cannot apply this technique. We can apply a different one however. The logarithm s has a decomposition $s = ps_1 + s_2$ where s_1 and s_2 are between 0 and $p - 1$. We first find the logarithm $s_1 \mod p$ of b^p to the base a^p. We then find the logarithm $s_2 \mod p$ of $ba^{-ps_1} = a^{s_2}$ to the base a. The logarithm of b to the base a is then $s = ps_1 + s_2$.

Combining these two simple methods allows us to reduce finding logarithms in a group of composite order to finding logarithms in subgroups of prime order. The quantum memory requirements for the control register thus gets reduced to $\lceil \log_2 2p \rceil + 1$ qubits, where p is the largest prime dividing N. Suppose $M = \prod_{j=1}^{n} p_j^{a_j}$, then the total number of group multiplications is in

$$O\left(\sum_{j=1}^{n} a_j (\log p_j + 1)\right) = O(\log M).$$

The cost of combining all the answers to find the answer modulo M is $O(\log^2 M)$ elementary classical operations.

4.2.2. Replacing the control register with a reusable control qubit. In section 2.4 it was shown how to obtain an n-bit estimate of an eigenvalue of a unitary operator U using a control register containing n qubits and using the inverse quantum Fourier transform $QFT(2^n)^{-1}$ which was illustrated in figure 2.3. If we take a closer look at that figure, we notice that we could in fact have observed the first qubit immediately before we used it as a control bit for the rotations later (see figure 4.1). This observation was illustrated in [**GN96**] and they call this part-

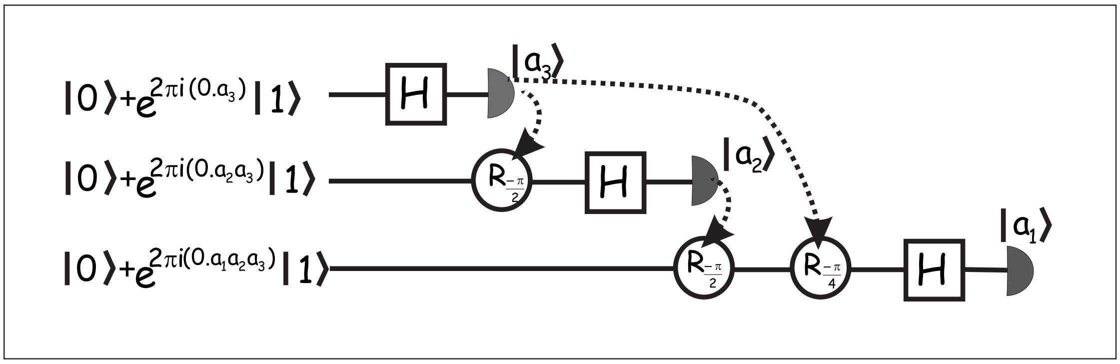

FIGURE 4.1. A "Semi-classical" Fourier transform. The first qubit could be observed first and the result used to classically control the rotation on the next qubits. The same could be done with the second qubit.

classical part-quantum version of the QFT a "semi-classical" Fourier transform. This does not require any 2-qubit quantum gates. However, the advantages go further if the experimental realisation of our quantum computer permits. Note that the preparation of the second qubit could occur *after* the first qubit has been measured. In fact, if we could reset our first qubit to $|0\rangle$, we could then reuse this qubit instead of requiring an additional one in our system. The same goes for the third qubit and so on. Alternatively, we could throw away each qubit once it is measured, and then introduce the subsequent control qubit. This lends itself to a system which has *flying qubits* [**THL$^+$96**]. The advantage is that we do not need to maintain a coherent superposition of more than one-qubit in a control register. In other words, the physical requirements of the control qubits are different from those in the target register (as emphasised to me by David DiVincenzo). That is the control register needs to be some system that can be quickly and reliably prepared, measured and reset. The target register can take lots of time to prepare, but should not decohere much over time (as is typical with systems that can be easily measured), and it does not need to be easily measured (since we never measure it).

REMARK 90. *Algorithm 7 (Eig_Est) can be implemented using only one* flying *control qubit at a time.*

This has consequences on the memory demands of a system used to implement the factoring algorithm for example. This observation was made independently by E. Knill as noted in [**Zal98**] (where he also discusses the effects of this technique on the factoring algorithm).

4.2.3. Cool only if necessary. For most current attempts at implementing quantum computers, cooling the starting state down to the all-zero state is quite a challenging task. Some algorithms do not require this.

For example, the order-finding algorithm in $\mathbb{Z}_N^*$ will work almost as well if the target register is in the maximally mixed state or some natural equilibrium states (we describe one in the next section). It is therefore not worth any significant effort to cool the target register down to the $|\,00\ldots0\rangle$ state and then set it to the state

$|1\rangle$. In fact, the only state one must intentionally avoid with high probability is the state $|0\rangle$, since this is the only trivial eigenvector of multiplication by a modulo N. The other states that might pose some problem have non-trivial greatest common factors with N - but if these occur with significant amplitude we can efficiently factor N without the order-finding algorithm!

4.3. Quantum Computation using Nuclear Magnetic Resonance

Unlike the ideal two-spin computer described earlier, modern liquid state NMR computer does not manipulate just two spins. Firstly, the two spins are usually part of a larger molecule. For example, they could be two hydrogen atoms in a cytosine molecule (see figure 4.2). Further, they do not work with just one molecule, since the spin state of a single proton is not reliably detectable with existing apparatus. They use an *ensemble* of roughly $10^{17} - 10^{20}$ molecules and perform measurements on this ensemble. These molecule-sized computers are in a solution, often with water (H_2O) or heavy water (D_2O) as a solvent. To distinguish our qubits from other spins in the sample, the frequencies of the qubit spins need to be different from the frequencies of the other spins in the molecule and in the solvent. For this reason we use deuterated cytosine, where we use D_2O as our solvent and the hydrogens of the cytosine, apart from our two qubits, are replaced with deuterium. We can still distinguish the two different hydrogens from each other since their energy eigenvalues (and therefore frequencies) are shifted according to their chemical neighbourhoods in slightly different ways.

The size of the output signal from a particular spin (ignoring noise and other errors) is proportional to the sum of the signals of that spin from each molecule. Let us consider a molecule with just one spin. We can assume that $|1\rangle$ gives a signal of size -1 and $|0\rangle$ gives a signal of size 1, where the minus sign means the signal is in the opposite direction. Ideally, when we have two molecules, $|1\rangle \otimes |1\rangle$

(we keep the $\otimes$ to remind us that these spins are on different molecules, and not just different spins on the same molecule) gives a signal of size -2, $|\,1\rangle \otimes |\,0\rangle$ and

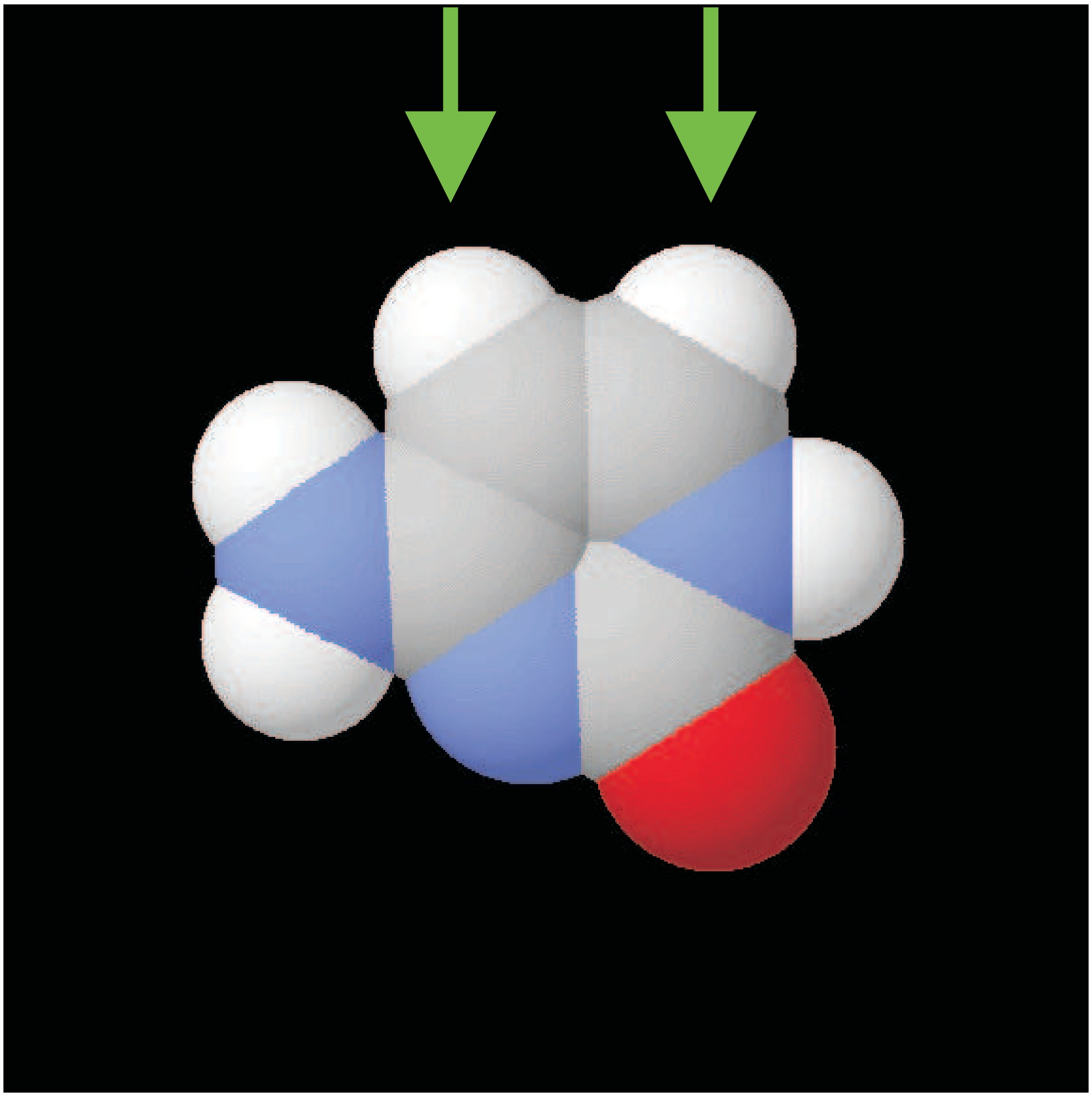

FIGURE 4.2. This picture illustrates a cytosine molecule. The green arrows point at the two hydrogen atoms we used in our computation. The remaining white atoms are deuterium. The grey atoms are carbon, blue atoms are nitrogen, and the red one is oxygen.

$|\,0\rangle\otimes|\,1\rangle$ give signals of size 0 and $|\,0\rangle\otimes|\,0\rangle$ gives a signal of size $+2$. In general when we have n molecules the ideal signal size varies from n to $-n$, and is proportional to the difference between the number of $|\,0\rangle$ states and the number of $|\,1\rangle$ states. Let us for a moment treat the output signal quantumly and note that if we observe the state $\frac{1}{\sqrt{2}}\,|\,0\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle$ with the measuring apparatus initialised to the state $|\,0\rangle$, after measuring we would have

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

If we start with the state $\frac{1}{2}\,|\,0\rangle\otimes|\,0\rangle + \frac{1}{2}\,|\,0\rangle\otimes|\,1\rangle + \frac{1}{2}\,|\,1\rangle\otimes|\,0\rangle + \frac{1}{2}\,|\,1\rangle\otimes|\,1\rangle$ and measure we would get

$$\frac{1}{2}\left(|\,0\rangle\otimes|\,0\rangle\right)|\,{+2}\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\,|\,0\rangle\otimes|\,1\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle\otimes|\,0\rangle\right)|\,0\rangle + \frac{1}{2}\left(|\,1\rangle\otimes|\,1\rangle\right)|\,{-2}\rangle\,.$$

This means, for example, that if we observe the apparatus, then with probability $\frac{1}{2}$, we observe an output of $|\,0\rangle$ and are left with the state $\frac{1}{\sqrt{2}}\,|\,0\rangle\otimes|\,1\rangle + \frac{1}{\sqrt{2}}\,|\,1\rangle\otimes|\,0\rangle$. If we have N copies of the spin, then the output signal for the state $|\,x_1\rangle\otimes|\,x_2\rangle\otimes\ldots\otimes|\,x_N\rangle$ is proportional to $\sum_{j=1}^{N}(-1)^{x_j} = N - 2H(x)$, where $H(x)$ is the number of 1s in the string $x = x_1x_2\ldots x_N$. If we observe a single qubit in the state $\rho = \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix}$, then the expected size of the output signal is

$$\rho_{00} - \rho_{11} = Tr(\rho\sigma_z)$$

where

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

When we have N copies of this spin, the expected size of the output signal is

$$N(\rho_{00} - \rho_{11}) = NTr(\rho\sigma_z).$$

The expected size of the output signal is relevant since the law of large numbers implies that as N gets large then the size of the output signal we see tends to this expected size, $NTr(\rho\sigma_z)$. Since our N are roughly between 10^{17} and 10^{20}, this is an extremely accurate approximation and we will simply say that the signal size is proportional to $NTr(\rho\sigma_z)$.

The measurements do not distinguish different molecules, which we assume are independent of each other, so we can describe the system as being in the state $\rho \otimes \rho \otimes \cdots \otimes \rho$ where ρ describes the state of one molecule, which in general can have more than one spin. If we have N molecules, each with n distinguishable spins in the n-qubit state ρ, the output signal when probing the jth spin will be proportional to $NTr(\rho\sigma_z^j) = 2NTr(\rho\mathbb{I}_z^j)$ (where $\sigma_z^j = \mathbb{I} \otimes \cdots \otimes \mathbb{I} \otimes \sigma_z \otimes \mathbb{I} \otimes \cdots \otimes \mathbb{I}$ with the σ_z on the jth spin).

So what is the natural starting state ρ? We will assume the initial distribution is the natural equilibrium point eventually reached by the sample after it has been placed in the magnetic field oriented in the z direction. This natural equilibrium point is a function of the Hamiltonian of the molecules. For the purpose of estimating the equilibrium distribution, we can ignore the coupling constants J (since they are much less than any of the ω terms). Let us suppose that all the spins have roughly the same frequency $\frac{\omega}{2\pi}$. The natural equilibrium point for that spin is $p_0 \,|\, 0\rangle \langle\, 0| + p_1 \,|\, 1\rangle \langle\, 1|$ where p_0 is proportional to $e^{\frac{\hbar\omega}{kT}}$ and p_1 is proportional to $e^{-\frac{\hbar\omega}{kT}}$ (k is Boltzmann's constant and T is the temperature). Since ω is proportional to the strength of the magnetic field, we could make p_0 close to 1 by increasing the field strength and lowering the temperature. However there are practical reasons why we cannot increase the field strength to a point where p_0 is close to 1. Further, we cannot lower the temperature arbitrarily close to 0 since the sample will freeze well before that point, and this poses problems, in particular the molecules can no longer be treated as independent computers. In current NMR quantum computation, the term $\frac{\hbar\omega}{kT}$ is quite small, roughly 10^{-5}, and thus we can approximate p_0 and p_1 by $\frac{1}{2}+\delta$ and $\frac{1}{2}-\delta$ where $\delta \approx \frac{\hbar\omega}{2kT}$.

The natural equilibrium distribution for the whole system is then roughly a binomial distribution. In other words, we get the term we really want $|\,0000\rangle\,\langle\,0000|$ with probability roughly $\left(\frac{1}{2}+\delta\right)^n \approx \frac{1}{2^n}+\frac{n\delta}{2^{n-1}}$.

Ideally, we would like to somehow prepare some qubits that are almost entirely in the state $|\,00\ldots0\rangle\,\langle\,00\ldots0|$.

One way of better approximating the state $|\,00\ldots0\rangle\,\langle\,00\ldots0|$ [**Tap98**] is to apply a transformation that sorts our n qubit strings according to their Hamming weights. This way the more likely strings have a longer string of leading 0s in their binary representation. This encoding means that with probability $1-o(1)$, the leftmost $O(\delta^2 n)$ bits are all 0. We could use these first $O(\delta^2 n)$ qubits for our computation.

However the straightforward ways of sorting by Hamming weight require an ancilla initialised to $|\,0\rangle$ states, which brings us back to our original problem! Schulman and Vazirani found a clever method of permuting the computational basis states that produces roughly the same result, and this method does not require an ancilla [**SV98**]. In current experiments δ is roughly 10^{-5}, thus to get even 100 qubits mostly in the $|\,0\rangle$ state requires molecules with about 10^{12} spins, which is not practical. Further, the algorithm assumes the operations are performed perfectly. As far as I am aware, no careful treatment of this technique in the presence of errors and imprecise operations has been done. We cannot simply use error-correcting codes since they require ancilla bits initialised to $|\,0\rangle$! However this algorithm gives hope that there is nothing in principle stopping us from initialising a register to the desired starting state.

It suffices however to produce a starting n-qubit state of the form $\epsilon\,|\,00\ldots0\rangle\,\langle\,00\ldots0| + (1-\epsilon)\frac{1}{2^n}\mathbb{I}$, where ϵ is large enough to produce an observable signal. The reason is that when we apply our algorithm A on this state, we will get

$$\epsilon A\,|\,00\ldots0\rangle\,\langle\,00\ldots0|\,A^* + (1-\epsilon)\frac{1}{2^n}\mathbb{I}.$$

Let $|\,\Psi\rangle = A\,|\,00\ldots0\rangle$ be the output of the algorithm. When we measure spin j, we get an output signal proportional to $\epsilon N Tr(|\,\Psi\rangle\,\langle\,\Psi|\,\sigma_z^j)$ (note that $Tr(\mathbb{I}\sigma_z^j = 0)$. This equals $\epsilon N(p_0 - p_1)$ where p_0 is the probability of measuring $|\,0\rangle$ and p_1 is the probability of measuring $|\,1\rangle$ if we measure the jth qubit of $|\,\Psi\rangle$.

No unitary transformation can transform n independent qubits in the state ρ into

$$\epsilon\,|\,00\ldots0\rangle\,\langle\,00\ldots0| + (1-\epsilon)\frac{1}{2^n}\mathbb{I}$$

(see e.g. [**HSTC98**]), so we must somehow use an ancilla and trace it out.

Let me now describe the methods that have been used in practice and work for small systems.

Ignoring terms of second order in δ, the natural equilibrium state of two spins is

$$\left(\frac{1}{4}+\delta\right)|\,00\rangle\,\langle\,00| + \frac{1}{4}\,|\,01\rangle\,\langle\,01| + \frac{1}{4}\,|\,10\rangle\,\langle\,10| + \left(\frac{1}{4}-\delta\right)|\,11\rangle\,\langle\,11| \tag{27}$$

$$= \frac{1}{4}\mathbb{I} + \delta\,|\,00\rangle\,\langle\,00| - \delta\,|\,11\rangle\,\langle\,11|\,. \tag{28}$$

Note that

$$\begin{aligned}\epsilon\,|\,00\rangle\,\langle\,00| + (1-\epsilon)\frac{1}{4}\mathbb{I} = \quad & \tfrac{1}{3}\left(\tfrac{1}{4}\mathbb{I} + \tfrac{3}{4}\epsilon\,|\,00\rangle\,\langle\,00| - \tfrac{3}{4}\epsilon\,|\,11\rangle\,\langle\,11|\right) \\ & + \tfrac{1}{3}\left(\tfrac{1}{4}\mathbb{I} + \tfrac{3}{4}\epsilon\,|\,00\rangle\,\langle\,00| - \tfrac{3}{4}\epsilon\,|\,10\rangle\,\langle\,10|\right) \\ & + \tfrac{1}{3}\left(\tfrac{1}{4}\mathbb{I} + \tfrac{3}{4}\epsilon\,|\,00\rangle\,\langle\,00| - \tfrac{3}{4}\epsilon\,|\,01\rangle\,\langle\,01|\right).\end{aligned}$$

This equation suggests the following strategy. Leave the top third of the sample alone (call this operation U_0), map the middle third (using a controlled-NOT) from

$$\left(\frac{1}{4}\mathbb{I} + \delta\,|\,00\rangle\,\langle\,00| - \delta\,|\,11\rangle\,\langle\,11|\right) \rightarrow \left(\frac{1}{4}\mathbb{I} + \delta\,|\,00\rangle\,\langle\,00| - \delta\,|\,01\rangle\,\langle\,01|\right)$$

(call this U_1) and the bottom third from

$$\left(\frac{1}{4}\mathbb{I} + \delta\,|\,00\rangle\,\langle\,00| - \delta\,|\,11\rangle\,\langle\,11|\right) \rightarrow \left(\frac{1}{4}\mathbb{I} + \delta\,|\,00\rangle\,\langle\,00| - \delta\,|\,10\rangle\,\langle\,10|\right)$$

(call this U_2), and then average over all the molecules again. This gives us the state

$$\frac{4}{3}\delta\,|\,00\rangle\,\langle\,00| + (1 - \frac{4}{3}\delta)\frac{1}{4}\mathbb{I}.$$

One way of looking at this is as adding a virtual two-qubit register to each molecule, a location register, preparing the state

$$\frac{1}{3}\,|\,0\rangle\,\langle\,0| + \frac{1}{3}\,|\,1\rangle\,\langle\,1| + \frac{1}{3}\,|\,2\rangle\,\langle\,2|\,,$$

applying a controlled-U_j on the two systems, and then tracing out the location qubits giving us the desired state. In practice we can spatially address different parts of the sample, so that one third of them (say the top third of the test tube) are designated to be in region $|\,0\rangle\,\langle\,0|$, the middle third are in region $|\,1\rangle\,\langle\,1|$ and the bottom third are in region $|\,2\rangle\,\langle\,2|$. We can thus apply U_0 to the top third, U_1 to the middle third, and U_2 to the bottom third. Cory et. al [**CFH96**] describe such a method (field gradients). Knill et. al [**KCL98**] designed a similar scheme where instead of averaging over spatial configurations they would run U_0, U_1 and U_2 at three separate times and combine up the output signals afterwards (temporal averaging). Gershenfeld and Chuang [**GC97**] have also described an elegant method called logical labelling that uses additional spins in the molecule as the additional control bits. Each of these methods allows us to prepare pseudo-pure states.

Unfortunately, the simple generalisations of these methods to large systems only produce states with ϵ roughly $\frac{n\delta}{2^n}$. To have any hope of observing a signal we need the number of copies N of the molecule to be of size on the order of $\frac{1}{\epsilon}$, which implies that N must be exponential in n, the number of qubits we wish to have. This means that these techniques alone will not suffice in providing efficiently scalable quantum computation, but must be combined with other methods, such as those in [**SV98**]. We are still seeking a method that is both practical and whose complexity scales at most polynomially in the number of qubits we wish to have.

4.4. Interesting algorithms for implementation with few qubits

4.4.1. Deutsch algorithm. As an example of the hidden subgroup problem, this problem is the most interesting two-qubit algorithm to implement and can also be implemented on ensemble computers using only 2 qubits. This algorithm was implemented by Jones and myself using [**JM98**] two hydrogen nuclei in deuterated cytosine (see figure 4.2). More specifically, we used a 50 mM solution of cytosine in D_2O. We worked at room temperature, $20°C$ with a magnetic field of frequency $500MHz$ and a J-coupling term of $7.2Hz$ between the two ${}^1H$ atoms. This J-coupling corresponds to the term $14.4\pi\hbar\mathbb{I}_z \otimes \mathbb{I}_z$ in the Hamiltonian. The resonant frequencies of the two hydrogen atoms differed by $763Hz$, which allowed us to differentiate the two.

We used the gradient techniques of Cory et al. to prepare (approximately) the pseudo-pure state

$$\rho_{01} = \epsilon \,|\, 01\rangle \, \langle\, 01| + (1-\epsilon)\frac{1}{4}\mathbb{I}.$$

We applied a 90°_y pulse to both spins to effect the pseudo-Hadamard operator and thus produced a pseudo-pure state corresponding to the pure state

$$(|\,0\rangle + |\,1\rangle)\,(|\,0\rangle - |\,1\rangle)\,.$$

We then implemented one of the four functions $f : \{0,1\} \to \{0,1\}$. For example, to implement $f_{01}(0) = 0, f_{10}(1) = 1$, via the operator

$$U_{01} : |\,x\rangle\,|\,y\rangle \to |\,x\rangle\,|\,y \oplus f(x)\rangle$$

we could apply the following sequence (based on the one in [**JM98**]; see appendix A.7 for details):

$$(29) \qquad 90^\circ \mathbb{I}^2_y - \frac{1}{4J_{12}} - 180^\circ_x - \frac{1}{4J_{12}} - 180^\circ_x - 90^\circ \mathbb{I}^1_z - 90^\circ \mathbb{I}^2_z - (-90^\circ)\mathbb{I}^2_y$$

where $\frac{1}{4J_{12}}$ indicates simply waiting for a period of time $\frac{1}{4J_{12}}$ and 180°_x indicates a $180^\circ \mathbb{I}_x$ on all spins (when the frequencies of the two spins are close together, these *hard* pulses are easier to implement than *selective* pulses, that is pulses that only target one of the spins). In general a pulse sequence for U_f produces a pseudo-pure state corresponding to the pure state

$$\left((-1)^{f(0)}\,|\,0\rangle + (-1)^{f(1)}\,|\,1\rangle\right)(|\,0\rangle - |\,1\rangle)\,,$$

as described in section 2.1.

A pseudo-Hadamard gate applied to both qubits would give us a pseudo-pure state corresponding to the pure state

$$|\,f(0) \oplus f(1)\rangle\,|\,1\rangle$$

and a subsequent measurement of the first qubit would give us a signal proportional to $Tr(|\,b\rangle\,\langle b|\,\sigma_z) = (-1)^b$ where $b = f(0) \oplus f(1)$ and measuring the second qubit would give us a signal proportional to -1. We implemented similar pulse sequences for all four possible functions $f : \{0,1\} \to \{0,1\}$. We first implemented each U_f

twice, once with a pseudo-pure starting state corresponding to $|0\rangle|0\rangle$ and again with a pseudo-pure state corresponding to $|1\rangle|0\rangle$ in order to simply compute $f(x)$ on every possible classical input. The output signals for the 'classical' computations appear in figure 4.3. We then implemented the Deutsch algorithm to find the value of $f(0) \oplus f(1)$ with only one application of U_f. The output signals from these experiments are shown in figure 4.4. More details are provided in [**JM98**]. A similar implementation with a chloroform molecule was announced shortly after [**CVLL98**].

4.4.2. Quantum searching. Quantum searching in a space of size $N = 4$ can also be implemented with only two qubits. On an ensemble computer, it is useful to promise that there is only one solution to $f(x) = 1$. Otherwise, if there is more than one solution, we observe an average of the corresponding signals, which does not necessarily give useful information. It also convenient when exactly one fourth of the inputs are solutions, since in that case one iteration of the searching iterate will produce exactly a superposition of the satisfying assignments (see section 2.7. When $N = 4$ and we have only one solution, we enjoy both of these advantages.

This algorithm was implemented in [**GCK98**] and [**JMH98**].

4.4.3. Quantum counting. Quantum counting can also be done with as few as 2 qubits. One qubit is used for the "semi-classical" phase estimation and the other is the input to a function f. It gets more interesting of course as the domain of f increases. The averaging in NMR that we describe in section 4.3 turns out to be an advantage! This implementation is described in [**JM99**].

The quantum counting, or more generally, amplitude estimation algorithm, seeks to estimate an eigenvalue of the unitary operator $G = -AU_0A^{-1}U_f$ defined in section 2.7. We adapted this algorithm to work with one control bit by implementing, for increasing r, a controlled-G^r. We know that G has two

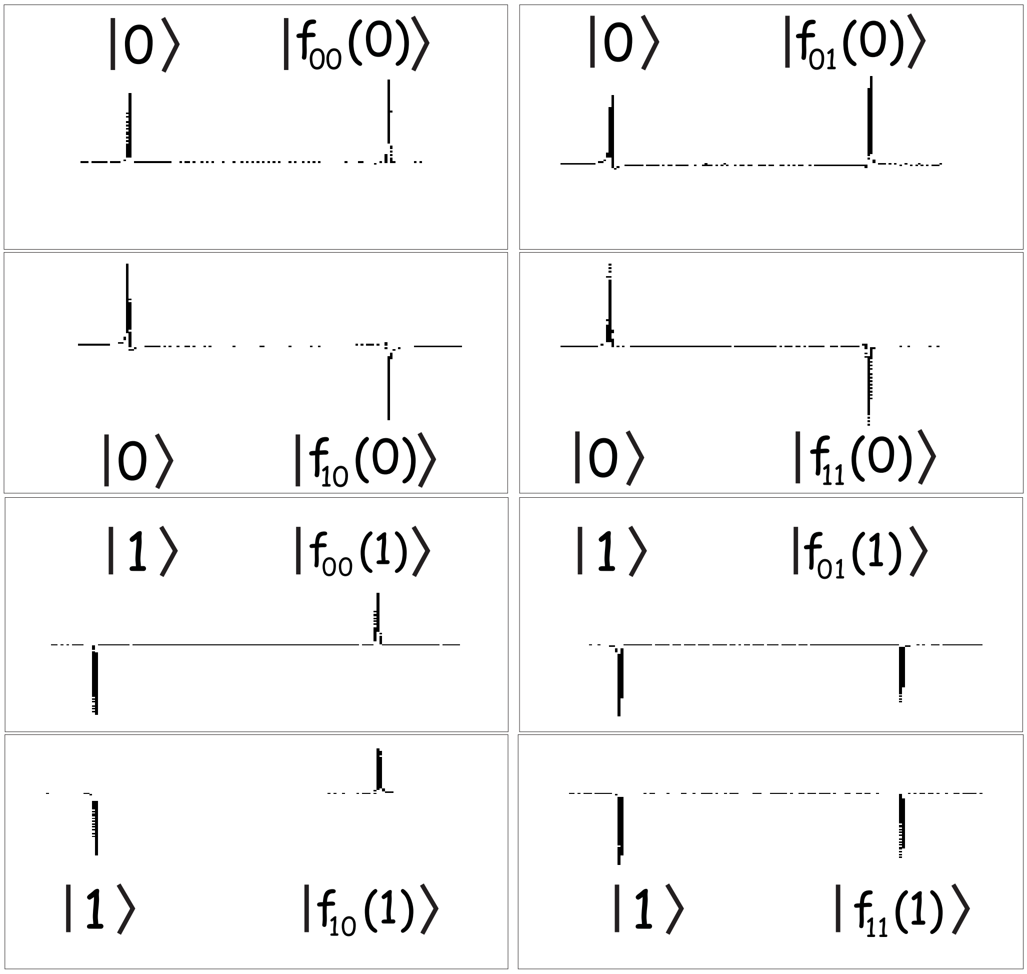

FIGURE 4.3. An upward peak corresponds to the state $|0\rangle$ and a downward peak corresponds to the state $|1\rangle$. The starting state has both peaks pointing up. The left qubit stores the input to f and the answer is added to the right qubit. All four functions were evaluated on both possible input values.

eigenvectors $|\Psi_+\rangle$ and $|\Psi_-\rangle$ with eigenvalues $e^{2\pi i\omega}$ and $e^{-2\pi i\omega}$ where $A|0\rangle = \sin(\pi\omega)|X_1\rangle + \cos(\pi\omega)|X_0\rangle$ and $U_f|X_1\rangle = -|X_1\rangle$ and $U_f|X_0\rangle = |X_0\rangle$. If we

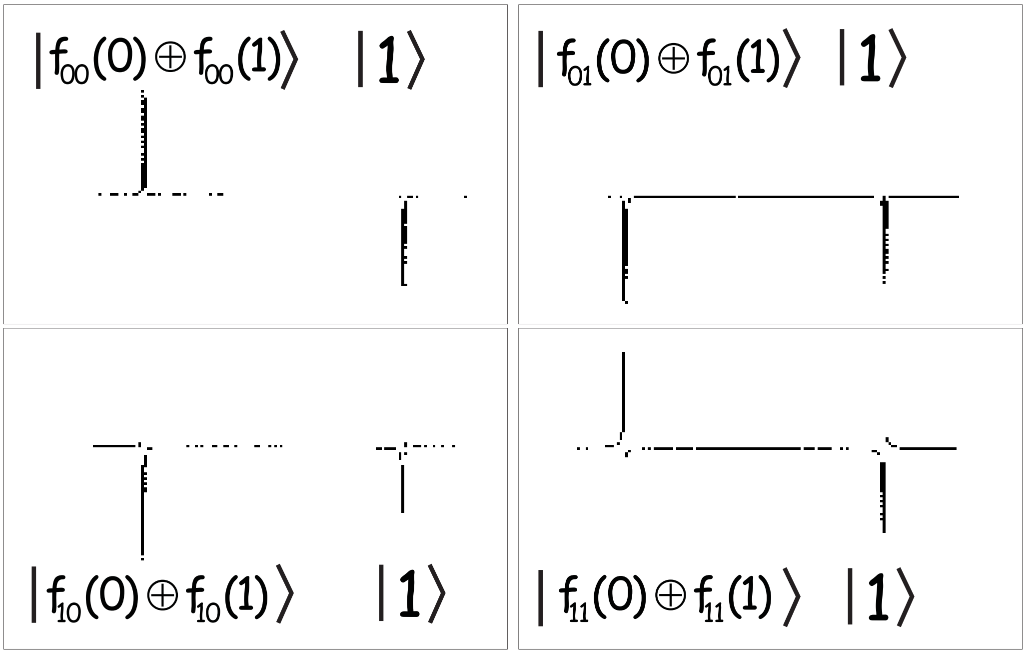

FIGURE 4.4. The four output signals correspond to the result of applying the Deutsch algorithm with the four functions $f : \{0,1\} \to \{0,1\}$. The left qubit stored the value of $f(0) \oplus f(1)$. The right qubit is in the state $|1\rangle$.

start with the state $(|0\rangle - |1\rangle)|\Psi_+\rangle$ and apply a controlled-G^r, we get the state $(|0\rangle - e^{2\pi i r\omega}|1\rangle)|\Psi_+\rangle$. An inverse pseudo-Hadamard transform on the first qubit gives us

$$\left(\frac{1+e^{2\pi i r\omega}}{2}|0\rangle + \frac{1-e^{2\pi i r\omega}}{2}|1\rangle\right)|\Psi_+\rangle .$$

Tracing out the target register we see that the first qubit is in the state

$$\begin{pmatrix} 1+\cos(2\pi r\omega) & i\sin(2\pi r\omega) \\ -i\sin(2\pi r\omega) & 1-\cos(2\pi r\omega) \end{pmatrix}.$$

The same result is obtained if we replace $|\,\Psi_+\rangle$ with $|\,\Psi_-\rangle$, except that the two off-diagonal elements are negated. Thus the same diagonal elements are also obtained from any superposition or statistical mixture of the two, such as $A\,|\,0\rangle$. Starting with $A\,|\,0\rangle = e^{i\pi\omega}\,|\,\Psi_+\rangle + e^{-i\pi\omega}\,|\,\Psi_-\rangle$ in the target register, applying the controlled-G^r, and then tracing out the target register gives the state

$$\rho = \frac{1}{2}\begin{pmatrix} 1+\cos(2\pi r\omega) & 0 \\ 0 & 1-\cos(2\pi r\omega) \end{pmatrix},$$

and measuring will produce an output signal of size proportional to $Tr(\rho\sigma_z) = \cos(\pi r\omega)$. The averaging effects actually help provide a more precise estimate of $\cos(\pi r\omega)$! Running this experiment for exponentially increasing r and classically post-processing (similar to the methods in [**Kit95, Bhi98**]) gives us counting algorithms similar to those described in section 2.7. The details of this implementation appear in [**JM99**].

4.4.4. Order-finding. Pick any group whose operation can be realised with few qubits, add one control bit, and you can find the order of that operation using the order-finding algorithm and the modified phase estimation method described in section 4.2.2 and in [**Bhi98**]. For example, multiplication by 2 mod $2^k - 1$ corresponds to a cyclic permutation of the k bits, which can be implemented using only k qubits. The order-finding algorithm can find this period.

It is not worth implementing this algorithm on very small NMR quantum computers (or other ensemble computers) because the average of the signals from the eigenvalue estimations $\left|\,\widetilde{\tfrac{k}{r}}\right\rangle$ does not provide useful information. We could in principle perform the post-processing that computes the order r quantumly (so that we observe an average signal over states of the form $|\,r\rangle\left|\,\widetilde{\tfrac{k}{r}}\right\rangle$), but in practice this greatly increases the required amount of quantum memory.

4.4.5. Simulating Quantum Chaotic Maps. Schack and Brun [**Sch98, BS99**] have suggested using small quantum computers to implement quantum chaotic maps and to observe and study the predicted chaotic features. Once the number of qubits is above a few dozen, we are out of the range of what classical computers can simulate. This is not the case for the factoring problem, where it would take several hundred working logical qubits to outperform classical algorithms.

APPENDIX A

Appendix

A.1. Computing a controlled-U

Given a quantum gate array for computing U, we wish to create a network for computing the controlled-U which maps $|\,0\rangle\,|\,y\rangle \rightarrow |\,0\rangle\,|\,y\rangle$ and $|\,1\rangle\,|\,y\rangle \rightarrow |\,1\rangle\,U\,|\,y\rangle$.

It is important to remember that we do not measure the control bit, and then apply U if the outcome is 1 (in some circumstances, such as when the control qubit never used again, or as described in section 4.2.2, this is possible, but not in general).

In [**BBC$^+$95**], they show the essential ingredients for performing this task. Firstly, for each gate G in the available family of gates $\mathbb{G}$, we should decompose the controlled-G (at least approximately) into a network of gates from G. For example, the controlled-controlled-NOT can be decomposed as shown in figure A.1.

Creating a network for the controlled-U is now simple. Just replace every gate G in U with a gate array for computing the controlled-G, always using the same control bit, as illustrated in figure A.2. With our universal set of gates G we can, for any δ, efficiently approximate every controlled-G with error at most δ using $poly(\frac{1}{\delta})$ gates from G. Suppose we have a network $\mathcal{N}$ with T gates not necessarily from our universal set. We wish to approximate $\mathcal{N}$ with a network $\mathcal{N}'$ such that their respective outputs $|\,\Psi\rangle = \mathcal{N}\,|\,00\ldots0\rangle$ and $|\,\Psi'\rangle = \mathcal{N}'\,|\,00\ldots0\rangle$ are 'close', that is $||\,\Psi'\rangle - |\,\Psi\rangle| < \epsilon$ for some small ϵ. It suffices to approximate each gate G

in $\mathcal{N}$ with an error in $O(\frac{\epsilon}{T})$. We can efficiently do this using $poly(\frac{T}{\epsilon})$ gates from our universal set.

Although it suffices to approximate every controlled-G, it is an interesting mathematical question if it is possible to do so exactly.

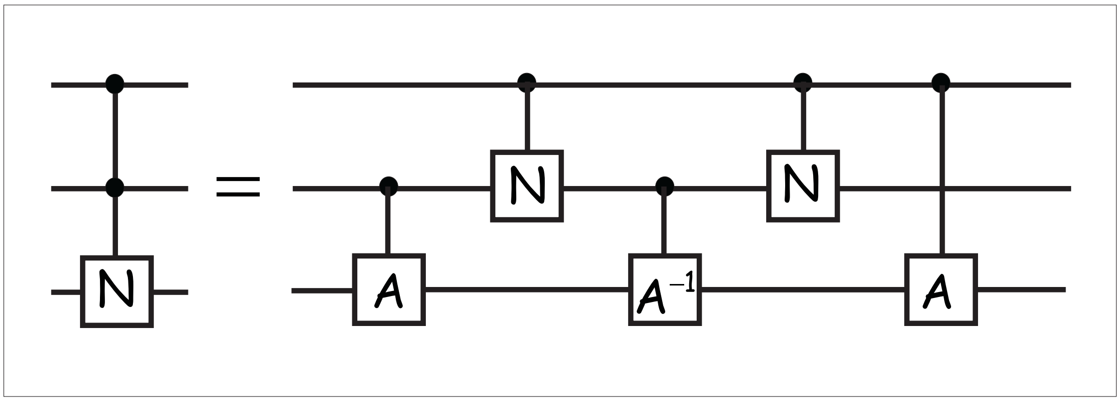

FIGURE A.1. A network for computing the controlled-controlled-NOT using only two-qubit gates. Here N denotes the NOT gate and A denotes the $\sqrt{NOT}$ gate.

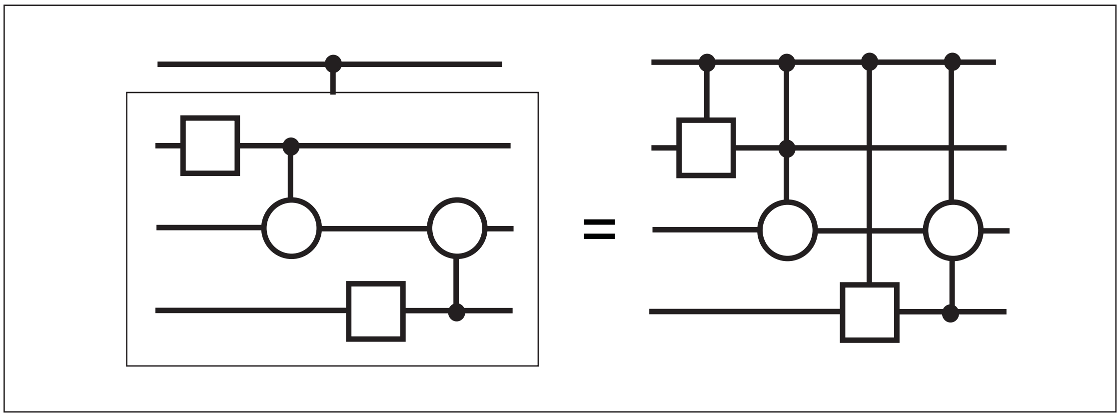

FIGURE A.2. A quantum network for computing the controlled-U given a network for U.

PROBLEM 91. *Does there exist a finite universal family of quantum gates $\mathbb{G}$, such that for each gate $G \in \mathbb{G}$, the controlled-G can be decomposed* exactly *as a finite concatenation of elements in $\mathbb{G}$.*

A.2. Computing $\Lambda_M(U)$

Given a means for computing U, for any positive integer M, we wish to create a gate array for computing $\Lambda_M(U) : |x\rangle |y\rangle \to |x\rangle U^x |y\rangle$. Note that this is a generalisation of the controlled-U.

We will consider two cases. The first case is where we are simply given a quantum network for computing U. For a fixed positive integer k, a controlled-U^{2^k} can easily be created by concatenating 2^k consecutive controlled-U arrays. If the quantum network for U has T gates, then the quantum network for the controlled-U^{2^k} has $O(2^kT)$ gates.

The second case is where it is possible to compute U^{2^k} more efficiently than by just iterating 2^k times the operator U. For example, if U corresponds to multiplication by a in some group G, we can first compute a^2 by squaring a, then compute a^4 by squaring a^2, and continue squaring a total of k times to compute a^{2^k}. This requires k group operations, and we then directly implement the operation $U_{a^{2^k}} = U_a^{2^k}$.

In either case, once we know how to implement the controlled-U, controlled-U^2, controlled-U^4, ..., controlled-U^{2^l}, we are ready to implement $\Lambda_M(U)$ using the well-known 'square and multiply' algorithm (see for example, Algorithms 2.143 and 2.227 in [**MvOV97**]). Using a qubit in the state $|x_j\rangle$ as the control bit for U^{2^j}, we will apply the operator U exactly $x_0 2^0 + x_1 2^1 + \ldots + x_{l-1}2^{l-1}$ times. In other words, U is applied x times where x equals $x_{l-1}\ldots x_1 x_0$ represented in binary. Therefore, such a gate array, as illustrated in figure A.3, will realise the operator $\Lambda_M(U)$.

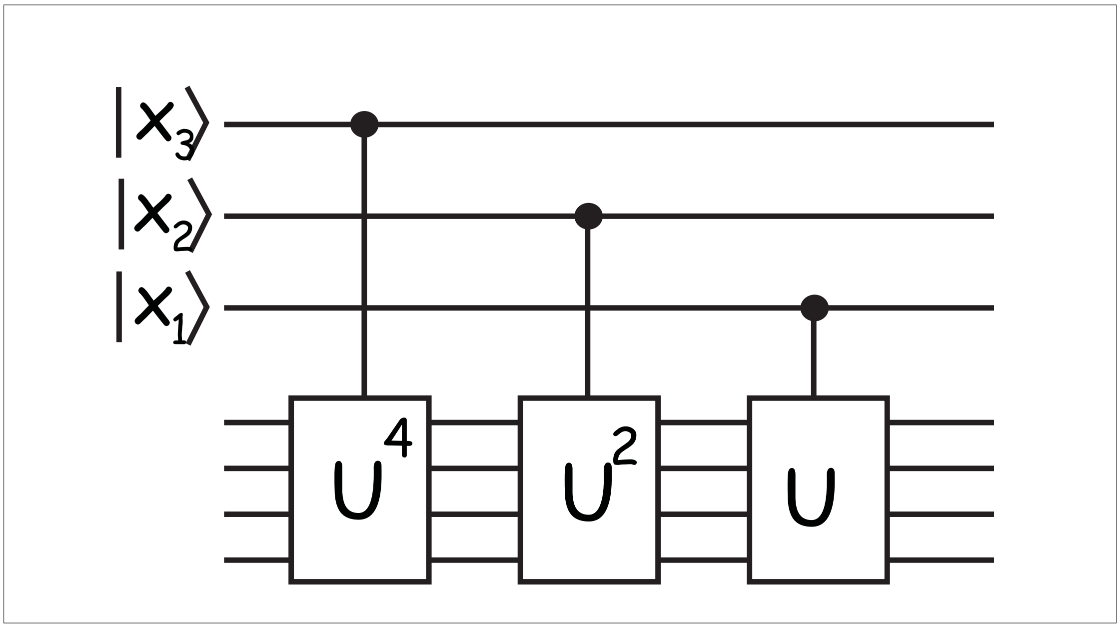

FIGURE A.3. A network for implementing $\Lambda_{2^3}(U) : |x\rangle|\Psi\rangle \to |x\rangle U^x|\Psi\rangle$, where $x = x_1 + 2x_2 + 4x_3$.

LEMMA 92. *Given a quantum network with T elementary gates for implementing U, we can implement $\Lambda_M(U)$ using $O(MT)$ gates. If $U = U_a$, the operator that multiplies by a in a group G where a group multiplication requires $O(T)$ gates, then we can implement $\Lambda_M(U_a)$ using $O(\log MT)$ elementary gates.*

A.3. Reversible Computing without keeping the input

If the function f is logically reversible, and we have a means of computing f^{-1}, then we can implement the operation $|x\rangle \to |f(x)\rangle$ (versus implementing $|x\rangle|y\rangle \to |x\rangle|y + f(x)\rangle$). More specifically, given a quantum network for implementing $|x\rangle|y\rangle \to |x\rangle|y + f(x)\rangle$ and one for implementing $|x\rangle|y\rangle \to |x - f^{-1}(y)\rangle|y\rangle$, we can implement $|x\rangle \to |f(x)\rangle$ as illustrated in figure A.4.

A.4. Fourier Transforms

When we have a non-trivial factorisation for M, say $M = AB$, there are two ways of combining $QFT(A)$ and $QFT(B)$ to produce $QFT(M)$. If A and B are coprime, we can make use of the Chinese remainder theorem, and represent

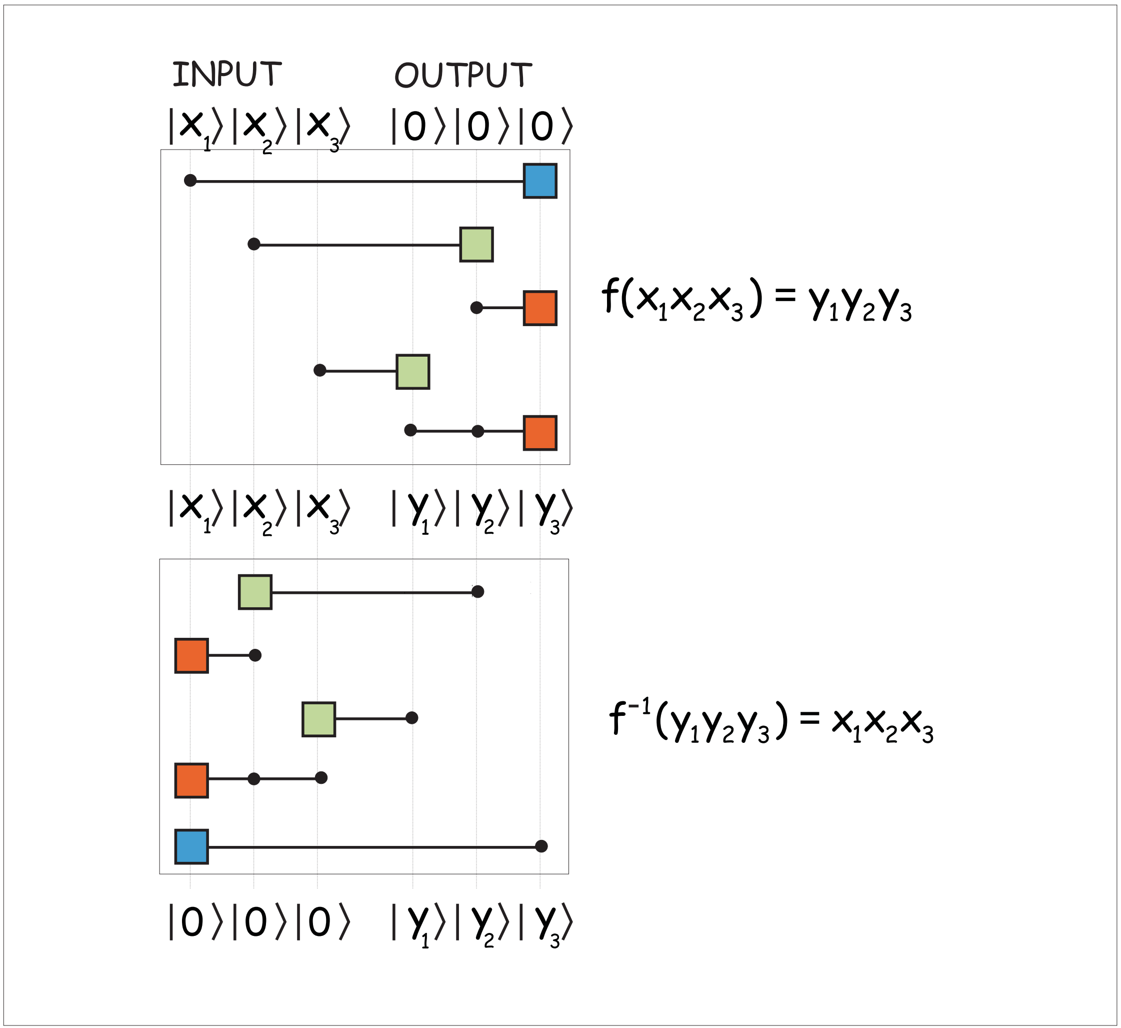

FIGURE A.4. When f is invertible, we can compute $f(x)$ without keeping the input. First compute $|x\rangle\,|f(x)\rangle$, and then uncompute the input using f^{-1}.

integers $k \in \{0,1,...,M-1\}$ as elements in $\{0,1,...,A-1\} \times \{0,1,...,B-1\}$ via the isomorphism $k \bmod M \longleftrightarrow (k \bmod A, k \bmod B)$. Using this representation, it is easy to verify that $QFT(M) = (U_B \otimes U_A)(QFT(A) \otimes QFT(B))$, where $U_B : |x \mod A\rangle \to |xB \mod A\rangle$, and $U_A : |x \mod B\rangle \to |xA \mod B\rangle$.

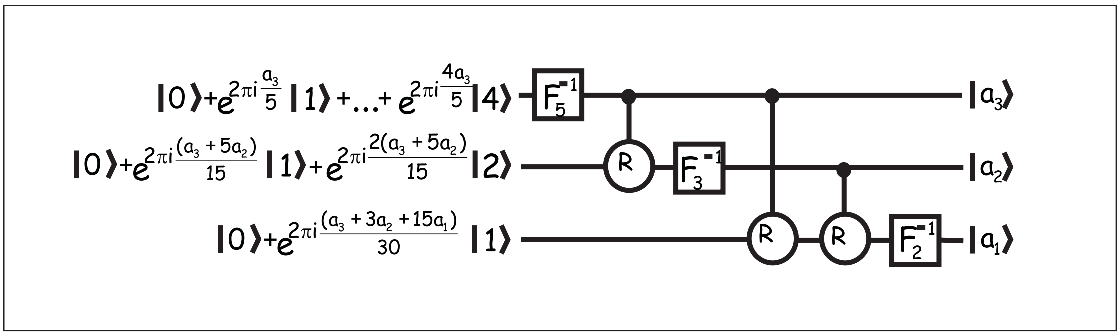

FIGURE A.5. This network realises the $QFT(AB)^{-1}$ when $a \bmod AB$ is represented as $|a \mod A\rangle |b \mod B\rangle$. Apply $QFT(A)$ on the first register followed by a multiplication by $B \bmod A$, and apply $QFT(B)$ on the second register followed by a multiplication by $A \bmod B$.

In general, for any A and B, say $A = 2$ and $B = 2^{n-1}$, we can generalise the technique described in section 2.3. We represent $a \in \{0,1,...,M-1\}$ as elements in $\{0,1,...,A-1\} \times \{0,1,...,B-1\}$ via the isomorphism $a_1B + a_2 \bmod M \longleftrightarrow (a_1 \bmod A, a_2 \bmod B)$, where $a_2 \in \{0,1,...,B-1\}$, and $a_1 \in \{0,1,...,A-1\}$. In the following equations, we will decompose the indices $k \in \{0,1,\ldots,AB-1\}$ as $\{k_2,k_1\}$ where $k = k_2A + k_1$, $k_2 \in \{0,1,\ldots,B-1\}$ and $k_1 \in \{0,1,\ldots,A-1\}$.

Suppose we are given the state

$$
\begin{aligned}
QFT(AB)\,|\,a\rangle &= \sum_{k=0}^{AB-1} e^{2\pi i \frac{ka}{AB}}\,|\,k\rangle = \sum_{k=0}^{AB-1} e^{2\pi i \frac{k(a_1B+a_2)}{AB}}\,|\,k\rangle \\
&= \sum_{0\le k_2<B, 0\le k_1<A} e^{2\pi i \frac{k_1a_1B+k_2a_2A+k_1a_2}{AB}}\,|\,k_2,k_1\rangle \\
&= \left(\sum_{0\le k_2<B} e^{2\pi i \frac{k_2a_2}{B}}\,|\,k_2\rangle\right)\left(\sum_{0\le k_1<A} e^{2\pi i \frac{k_1(a_1B+a_2)}{AB}}\,|\,k_1\rangle\right).
\end{aligned}
$$

Applying $QFT(B)^{-1}$ to the left register will give us

$$
|\,a_2\rangle\left(\sum_{0\le k_1<B} e^{2\pi i \frac{k_1(a_1B+a_2)}{AB}}\,|\,k_1\rangle\right).
$$

Extend the definition of R_w to $\mathbb{H}_B$ by mapping $|\,x\rangle \to e^{-2\pi ixw}\,|\,x\rangle$, $x \in \{0,1,\ldots,B-1\}$. A multi-qubit controlled-$R_{\frac{1}{AB}}$ (i.e. a $\Lambda_A(R_{\frac{2\pi}{AB}})$) will give us

$$
\begin{aligned}
&|\,a_2\rangle\left(\sum_{0\le k_1<A} e^{2\pi i \frac{k_1a_1B}{AB}}\,|\,k_1\rangle\right) \\
&= |\,a_2\rangle\left(\sum_{0\le k_1<A} e^{2\pi i \frac{k_1a_1}{A}}\,|\,k_1\rangle\right).
\end{aligned}
$$

and applying $QFT(A)^{-1}$ gives us $|\,a_2\rangle\,|\,a_1\rangle$.

Note that whereas in the input register $|\,x\rangle\,|\,y\rangle$ corresponded to $xB+y$, in the output register it corresponds to $yA+x$. This accounts for the reversal of the qubits at the end of the $QFT(2^n)$, but it is not quite so simple when a mixed radix representation is used. We will not worry about transforming the outputs back to the same representation as the inputs.

We can recursively apply this technique for any $N = A_1A_2\ldots A_n$ and map, for any $a \in \{0,1,\ldots,N-1\}$, $a = a_1A_2A_3\ldots A_n + a_2A_3A_4\ldots A_n + \ldots + a_{n-1}A_n + a_n$, $0 \le a_j < A_j$. The result is a network which maps, with indices k represented as

$|\,k_n\rangle\,|\,k_{n-1}\rangle\ldots|\,k_1\rangle$, $k = k_nA_{n-1}A_{n-2}\ldots A_1 + k_{n-1}A_{n-2}\ldots A_1 + \ldots + k_2A_1 + k_1$,

$$\sum_{k=0}^{N-1} e^{2\pi i\frac{ka}{N}}\,|\,k\rangle \to |\,a_n\rangle\,|\,a_{n-1}\rangle\ldots|\,a_1\rangle\,.$$

Again, the numbers are encoded differently in the output and input registers.

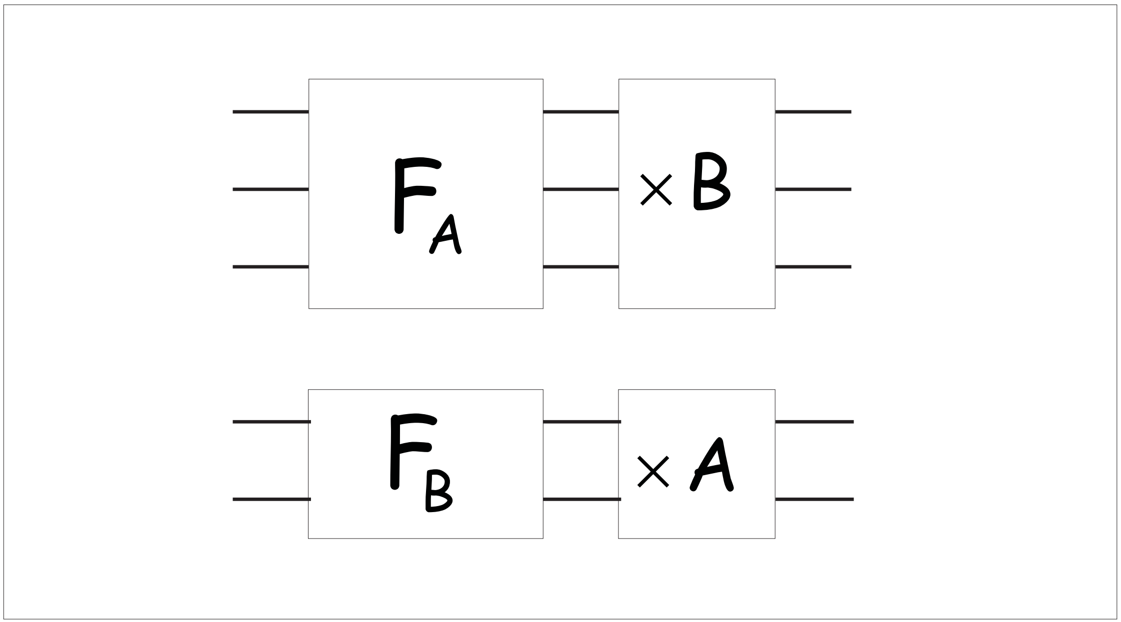

FIGURE A.6. This figure illustrates a network for $QFT^{-1}(ABC)$ where A,B and C are not necessarily coprime. Note that $|\,x\rangle\,|\,y\rangle\,|\,z\rangle$ represents $6x + 2y + z$ at the input, and $15z + 5y + x$ at the output. This corresponds to the necessary reordering of qubits in the $QFT(2^n)$ network described earlier.

A.5. Public Key Cryptography and Quantum Computing

A.5.1. Public Key Cryptography. Up until 1976, most publicly used cryptography was *private* or *symmetric* key cryptography. In private key schemes (such as DES or the one-time pad), the encryption and decryption keys are effectively the same and must somehow be secretly exchanged between participants, say Alice and Bob. In public key cryptography, first described publicly by Diffie and Hellman [**DH76b**] (see also [**Ell70, Ell87**]) the encryption and decryption keys are quite different, since it should be infeasible for one to compute the decryption key given the encryption key.

A reliable copy of everyone's encryption key should be publicly available. If Alice wishes to send a message M to Bob, she encrypts M using Bob's public key E_{Bob} to produce a ciphertext C. Only Bob possesses the decryption key D_{Bob} that allows him to compute P from C. Note than anyone can use the same public key E_{Bob} to send encrypted messages to Bob.

A.5.2. RSA. In 1977, Rivest, Shamir, and Adleman devised a public key scheme now known as RSA [**RSA78**] (similar to that of [**Coc73**]). Alice's public key E_{Alice} is a pair of integers (N, e). The integer N is a product of two distinct large primes p, q known only to Alice. The multiplicative group of integers modulo N, $\mathbb{Z}_N^*$ is the set of $\phi(N) = (p-1)(q-1)$ integers between 1 and $N-1$ that are coprime to N. From the encryption key e Alice computes a decryption key $D_{Alice} = d$ which satisfies $ed \equiv 1 \bmod \phi(N)$. Fermat's theorem implies that for any integer M between 0 and $N-1$, and any integer k, $M^{k\phi(N)+1} \equiv M \bmod N$.

Bob encrypts M by computing $C = M^e \bmod N$. Alice decrypts C by computing $C^d \bmod N$. Note that $C^d \equiv P^{ed} \equiv P^{k\phi(N)+1} \equiv P \bmod N$.

The security of this cryptosystem requires that it is difficult to compute the decryption key $D_{Alice} = d$ given $E_{Alice} = (N, e)$. If we know the factors p and q of N, we could easily compute d. Thus being able to factor suffices to compute P from C. **It is not known if it is necessary.**

A.5.3. Using Algorithm 13 to crack RSA. As we show in the next section, we can crack RSA by using Algorithm 15 (*Find_Order*) to factor N. We can however directly compute P from C without factoring N (we could factor N if we wish by solving this problem for several random C).

ALGORITHM 93. *(Find_RSA_Plaintext(C, N))*

Input:

- *Integers N and C.*

Output:

- *An integer P.*

Complexity:

- *$O(\log N)$ multiplications modulo N.*

Procedure:

1. *Use Algorithm 15 (Find_Order(C)) to find an integer r such that $C^r \equiv 1 \mod N$.*
2. *Use the extended Euclidean algorithm to compute an integer d satisfying $de \equiv 1 mod r$.*
3. *Output $P \equiv C^d \mod N$.*

THEOREM 94. *Algorithm 93 (Find_RSA_Plaintext(C, N)) finds C such that $P^e \equiv C mod N$.*

PROOF. Note that $ed \equiv 1 \bmod r$ means $ed = kr + 1$ for some integer k. Thus $C^d \equiv P^{ed} \equiv P^{kr+1} \equiv (P^r)^k P \equiv P \bmod N$. □

A.5.4. Using Algorithm 13 to factor. As mentioned earlier, we could use Algorithm 15 to factor any composite integer N. It of course suffices to *split* any composite integer N into two non-trivial factors. We assume N is odd, since it is easy to recognise and divide out any power of 2. Further, we can assume N is not a perfect power, since we can easily check to see if it is a perfect rth power for $r = 2, 3, \ldots \log_2 N$.

ALGORITHM 95. *(Split(N))*

Input:

- *An odd composite integer N that is not a prime power.*

Output:

- *A integer t.*

Procedure:

1. *Pick an integer $a \in \{1, 2, 3, \ldots, N-1\}$ uniformly at random.*
2. *Use the Euclidean algorithm to find $d = \gcd(a, N)$. If $d > 1$, output d (i.e. if d is not coprime to N, this is **NOT** a problem!)*
3. *Use Algorithm 13 three times with $M = 2N^2 > 2r^2$. If the three results are FAIL, go to step 1. Otherwise take r to be the minimum non-FAIL output. If r is odd go to step 1. If r is even, let $d = \gcd(a^{\frac{r}{2}} - 1, N)$. If $d = 1$, go to step 1.*
4. *Output d.*

THEOREM 96. *Algorithm Split(N) outputs a non-trivial factor of N and runs with expected running time of $O(\log^3 N)$ multiplications mod N and $O(\log^2 N)$ other elementary operations.*

PROOF. We will prove it for $N = pq$, where $p < q$ are distinct primes. It is easy to generalise. Let a_1 be a generator of $\mathbb{Z}_p^*$ and a_2 be a generator of $\mathbb{Z}_q^*$. By the Chinese Remainder theorem, the $(p-1)(q-1)$ integers in $\mathbb{Z}_N^*$ are in a $1-1$ correspondence with the pairs of integer $(x_1, x_2) \in \{1, 2, \ldots, p-1\} \times \{1, 2, \ldots, q-1\}$ via the mapping $a \mod N \to (a_1^{x_1}, a_2^{x_2})$ with $a \equiv a_1^{x_1} \mod p$ and $a \equiv a_2^{x_2} \mod q$. Selecting an integer a uniformly at random from Z_N^* is equivalent to selecting x_1 uniformly at random from $\{1, 2, \ldots, p-1\}$ and independently selecting x_2 uniformly at random from $\{1, 2, \ldots, q-1\}$.

The order of such an $a \mod p$ is

$$r_1 = \frac{p-1}{\gcd(x_1, p-1)}$$

and modulo q is

$$r_2 = \frac{q-1}{\gcd(x_2, q-1)}.$$

The order of a modulo pq is $r = \operatorname{lcm}(r_1, r_2)$.

Computing $\gcd(a^{\frac{r}{2}} - 1, N)$ will split N if and only if r_1 and r_2 contain the factor 2 with different multiplicities. Since x_1 and x_2 are chosen uniformly at random, the probability that r_1 and r_2 contain the factor two with different multiplicities is at least $\frac{1}{2}$. $\square$

Recursive application of this splitting algorithm gives us a factoring algorithm. By finding certificates for the primes (which can be done since we can now factor), we can make this a zero-error algorithm [**Buh96**].

A.6. Finding logarithms, DSA, and Diffie-Hellman

Diffie and Hellman [**DH76a, DH76b**] devised the following key exchange protocol (which can easily be turned into a public key scheme, such as the El-Gamal protocol; see sections 12.6 and 8.4 of [**MvOV97**]).

There is some publicly known group G and an element $\alpha \in G$.

Alice possesses a secret integer a, and makes public the element α^a. Bob possesses a secret integer b and makes public the element α^b.

Alice takes Bob's public α^b and computes $(\alpha^b)^a = \alpha^{ab}$. Bob takes Alice's public α^a and computes $(\alpha^a)^b = \alpha^{ab}$.

They now share the key α^{ab}. Any adversary knows α, α^a and α^b.

PROBLEM 97. *(Diffie-Hellman Problem) Given* $\alpha, \alpha^a, \alpha^b \in G$*, find* α^{ab}*.*

Note that multiplying α^a and α^b only gives α^{a+b}. It is clear that being able to compute discrete logarithms will allows us to solve the Diffie-Hellman problem and crack any cryptosystems which rely on difficulty of the Diffie-Hellman problem. For example the U.S. Digital Signature Algorithm which is a U.S. Federal Information Processing Standard (see section 11.5.1 of [**MvOV97**]) relies on the difficulty of the Diffie-Hellman problem in $GF(p)$ where p is a 512-bit prime number.

A.7. Implementing a function with a pulse sequence

In I stated that the pulse sequence

$$90°\mathbb{I}_y^2 - \frac{1}{4J_{12}} - 180°_x - \frac{1}{4J_{12}} - 180°_x - 90°\mathbb{I}_z^1 - (-90°)\mathbb{I}_z^2 - (-90°)\mathbb{I}_y^2 \tag{30}$$

(applied from left to right) would implement $U_{f_{01}}$ which corresponds to the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{31}$$

Here I will detail the action of this pulse sequence. The $90°\mathbb{I}_y^2$ effects the $90°\mathbb{I}_y$ pulse (see equation (7)) on the second qubit, corresponding to the matrix

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

The point of the sequence $\frac{1}{4J_{12}} - 180°_x - \frac{1}{4J_{12}} - 180°_x$ is to effect just the coupling term $2\hbar\pi J_{12}\mathbb{I}_z \otimes \mathbb{I}_z$ in the Hamiltonian for a period of time $\frac{1}{2J_{12}}$ in order to effect the operation $e^{-i\pi \mathbb{I}_z \otimes \mathbb{I}_z}$, which (apart from a global phase of $e^{-i\frac{\pi}{4}}$) has matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The problem with just waiting for a period of time $\frac{1}{2J_{12}}$ is that the entire Hamiltonian also has the terms $\hbar\omega_1 \mathbb{I}_z \otimes \mathbb{I}$ and $\hbar\omega_2 \mathbb{I} \otimes \mathbb{I}_z$. The 180°_x operation, which corresponds to the matrix

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

applied after each of two intervals of time $\frac{1}{4J_{12}}$, leaves the coupling term to evolve for a period of time $\frac{1}{2J_{12}}$, and causes the effect of the other two terms to cancel out over the two periods.

The $90^\circ \mathbb{I}_z^1$ is the operation $e^{-i\frac{\pi}{2}\mathbb{I}_z}$ on the first qubit, which corresponds to the matrix (with a global phase of $e^{-i\frac{\pi}{4}}$)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & i \end{pmatrix}.$$

The $(-90°)\mathbb{I}_z^2$ corresponds to the operation $e^{i\frac{\pi}{2}\mathbb{I}_z}$ on the second qubit, which has matrix (with a global phase of $e^{i\frac{\pi}{4}}$)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -i \end{pmatrix}.$$

The $(-90°)\mathbb{I}_y^2$ pulse has matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

If we multiply these matrices we will get the operation U_{01} (apart from a global phase factor).

Bibliography

[ABO97] D. Aharonov and M. Ben-Or. Fault tolerant quantum computation with constant error. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC '97)*, 1997. Also available at quant-ph/9611025.

[ADH97] L. Adleman, J. Demarrais, and M.D. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Massachusetts, 1974.

[AL98] Daniel S. Abrams and Seth Lloyd. A quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors. Technical report, 1998. Also available at quant-ph/9807070.

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

[BBC+95] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, 1995. On the quant-ph archive, report no. 9503016.

[BBC+98] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 352–361, Los Alamitos, California, November 1998. IEEE. On the quant-ph archive, report no. 9802049.

[BBD+97] A. Barenco, A. Berthiaume, D. Deutsch, A. Ekert, R. Jozsa, and C. Macchiavello. Stabilization of quantum computations by symmetrization. *SIAM Journal on Computing*, 26(5):1541–1557, 1997.

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. On the quant-ph archive, report no. 9605034.

[BDEJ95] Adriano Barenco, David Deutsch, Artur Ekert, and Richard Jozsa. Conditional quantum dynamics and quantum gates. *Physical Review Letters*, 74:4083–4086, 1995.

[Bea97] Robert Beals. Quantum computation of fourier transforms over symmetric groups. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC '97)*, pages 48–53, 1997.

[Ben73] Charles H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, November 1973.

[Ben89] Charles H. Bennett. Time/space trade-offs for reversible computing. *SIAM Journal on Computing*, 18(4):766–776, 1989.

[BEST96] Adriano Barenco, Artur Ekert, Kalle-Antti Suominen, and Paivi Torma. Approximate quantum fourier transform and decoherence. *Physical Review A*, 54(1):139–146, 1996.

[BH97] Gilles Brassard and Peter Høyer. An exact quantum polynomial-time algorithm for simon's problem. In *Proceedings of Fifth Israeli Symposium on Theory of Computing and Systems*, pages 12–23. IEEE Computer Society Press, June 1997.

[Bhi98] Wahid Bhimji. Approximate quantum fourier transforms and phase estimations, 1998. M.Phys. dissertation.

[BHMT99] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation, 1999.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In *Proceedings of 25th International Colloquium on Automata, Languages, and Programming (ICALP '98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 820–831. Springer-Verlag, 1998.

[BL95] D. Boneh and R. J. Lipton. Quantum cryptanalysis of hidden linear functions (extended abstract). volume 963 of *Lecture Notes on Computer Science*, pages 424–437, 1995.

[BS99] T. A. Brun and R. Schack. Realizing the quantum baker's map on a nmr quantum computer. *Physical Review A*, 59:2649–2658, 1999.

[Buh96] H. Buhrman. A short note on shor's factoring algorithm. *SIGACT News*, 27(1):89–90, 1996.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, October 1997.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53:17–25, 1995.

[CEH+99] Richard Cleve, Artur Ekert, Leah Henderson, Chiara Macchiavello, and Michele Mosca. On quantum algorithms. *Complexity*, 4:33–, 1999.

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A*, 454:339–354, 1998. quant-ph report no. 9708016.

[CFH96] D. G. Cory, A. F. Fahmy, and T. F. Havel. Nuclear magnetic resonance spectroscopy: An experimentally accessible paradigm for quantum. In *Proceedings of the 4th Workshop on Physics and Computation*, 1996. New England Complex Systems Institute.

[Cle94] Richard Cleve. A note on computing fourier transforms by quantum programs. Available at www.cpsc.ucalgary.ca/ cleve/pubs/fourier _ transform.ps, 1994.

[Cle99] Richard Cleve. An introduction to quantum complexity theory. In C.Macchiavello, G.M.Palma, and A.Zeilinger, editors, *Collected Papers on Quantum Computation and Quantum Information Theory*. World Scientific, 1999. To appear.

[Coc73] C. Cocks. A note on non-secret encryption. Technical report, Communications-Electronics Security Group, U.K., 1973. Available at http://www.cesg.gov.uk/downlds/nsecret/notense.pdf.

[Coh93] Henri Cohen. *A Course in Computational Algebraic Number Theory*. 1993.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (STOC'71)*, pages 151–158, 1971.

[Cop94] Don Coppersmith. An approximate fourier transform useful in quantum factoring. Research report, IBM, 1994.

[CS96] A. R. Calderbank and P. W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54:1098–1105, 1996.

[CTDL77] Claude Cohen-Tannoudji, Bernard Diu, and Franck Laloë. *Quantum Mechanics*, volume 1. John Wiley & Sons, 1977.

[CVLL98] I. L. Chuang, L. M. K. Vandersypen, X. Zhou D. W. Leung, and S. Lloyd. Experimental realization of quantum algorithm. *Nature*, 393:143–146, 1998.

[Dam98] Wim van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS'98)*, pages 362–367, 1998. Also available at quant-ph/9805006.

[Dav82] Martin Davis. *Computability and Unsolvability*. Dover Publications Inc., New York, 1982.

[Deu85] David Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400:97–117, 1985.

[Deu89] David Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A*, 425:73–90, 1989.

[DH76a] W. Diffie and M. E. Hellman. Multiuser cryptographic techniques. In *Proceedings of AFIPS National Computer Conference*, pages 109–112, 1976.

[DH76b] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[Dir58] Paul A.M. Dirac. *The Principles of Quantum Mechanics*. Clarendon Press, Oxford, fourth edition, 1958.

[DiV95] David P. DiVincenzo. Two-bit gates are universal for quantum computation. *Physical Review A*, 51(2):1015–1022, 1995. On the cond-mat archive, report no. 9407022.

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A*, 439:553–558, 1992.

[EH98] Mark Ettinger and Peter Høyer. On quantum algorithms for noncommutative hidden subgroups. quant-ph report 9807029, Los Alamos archive, 1998.

[Ell70] J. H. Ellis. The possibility of non-secret encryption. Technical report, Communications-Electronics Security Group, U.K., 1970. Available at http://www.cesg.gov.uk/downlds/nsecret/possnse.pdf.

[Ell87] J. H. Ellis. The story of non-secret encryption. Technical report, Communications-Electronics Security Group, U.K., 1987. Available at http://www.cesg.gov.uk/downlds/nsecret/ellis.pdf.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum mechanical description of physical reality be considered complete? *Physical Review*, 47:777–780, 1935.

[Ett98] J. M. Ettinger. On noncommutative hidden subgroups, 1998. A lecture at AQIP '98.

[EZ64] H. Ehlich and K. Zeller. Schwankung von polynomen zwischen gitterpunkten. *Mathematische Zeitschrift*, 86:41–44, 1964.

[Fey65] Richard P. Feynman. *The Feynman lectures on physics*, volume III: Quantum Mechanics. Addison-Wesley, 1965.

[Fey82] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6,7):467–488, 1982.

[FGGS98] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation in determining parity. Technical Report 9802045, Los Alamos archive, 1998.

[GC97] N. A. Gershenfeld and I. L. Chuang. Bulk spin-resonance quantum computation. *Science*, 275:350–356, 1997.

[GCK98] N. A. Gershenfeld, I. L. Chuang, and M. Kubinec. *Physical Review Letters*, 80:3408, 1998.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability (A guide to the theory of NP-completeness)*. W.H. Freeman and Company, New York, 1979.

[GJS76] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[GN96] R. B. Griffiths and C. S. Niu. Semi-classical fourier transform for quantum computation. *Physical Review Letters*, pages 3228–3231, 1996.

[Gol99] Oded Goldreich. A sample of samplers – a computational perspective on sampling (survey). Technical report, Electronic Colloquium on Computational Complexity, 199. Available at http://www.eccc.uni-trier.de/eccc/.

[Got98] Daniel Gottesman. A theory of fault-tolerant quantum computation. *Physical Review A*, 57:127–137, 1998. Also available at quant-ph/9702029.

[Gri97] D.Y. Grigoriev. Testing the shift-equivalence of polynomials by deterministic, probabilistic and quantum machines. *Theoretical Computer Science*, 180:217–228, 1997.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC'96)*, pages 212–219, Philadelphia, Pennsylvania, May 1996. ACM. On the quant-ph archive, report no. 9605043.

[Gro98] Lov K. Grover. Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters*, 80:4329–4332, May 1998.

[HKM98] T. Hayes, S. Kutin, and D. van Melkebeek. On the quantum complexity of majority. Technical Report TR-98-11, University of Chicago, Computer Science Department, 1998.

[Høy97] Peter Høyer. Conjugated operators in quantum algorithms. IMADA preprint, 1997.

[HR90] Torben Hagerup and Christine Rüb. Guided tour of Chernoff bounds. *Information Processing Letters*, 33(6):305–308, 1990.

[HSTC98] T. F. Havel, S. S. Somaroo, C.-H. Tseng, and D. G. Cory. Principles and demonstrations of quantum information processing by nmr spectroscopy. Technical report, 1998. Available at quant-ph/9812086.

[HW79] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, Oxford, fifth edition, 1979.

[JM98] J. A. Jones and M. Mosca. Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer. *Journal of Chemical Physics*, 109:1648–1653, 1998. Also available at Los Alamos archive quant-ph/9801027.

[JM99] J. A. Jones and M. Mosca. Approximate quantum counting on an nmr ensemble quantum computer. *Physical Review Letters*, 83:1050–1053, 1999. Also available at Los Alamos archive quant-ph/98008056.

[JMH98] J. A. Jones, M. Mosca, and R. H. Hansen. Implementation of a quantum search algorithm on a quantum computer. *Nature*, 393:344–346, 1998. Also available at Los Alamos archive quant-ph/9805069.

[KCL98] E. Knill, I. Chuang, and R. Laflamme. Effective pure states for bulk quantum computation. *Physical Review A*, 57:3348–3363, 1998.

[Kit95] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem. Available at Los Alamos e-Print achive (`http://xxx.lanl.gov`) as quant-ph/9511026, 1995.

[Kit97] A. Yu. Kitaev. Quantum computations: algorithms and error correction. *Russian Math. Surveys*, 52(6):1191–1249, 1997. Uspekhi Mat. Nauk **52**:6 53-112.

[KLZ97] Emanuel Knill, Raymond Laflamme, and Wojciech Zurek. Resilient quantum computation: Error models and thresholds. Technical report, 1997. Available at quant-ph/9702058.

[Knu98] Donald E. Knuth. *Seminumerical Algorithms*, volume 2. Addison-Wesley, third edition, 1998.

[Kob94] Neal Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag, New York, second edition, 1994.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, CA*, volume 1109 of *Lecture Notes in Computer Science*. Springer, 1996.

[Lev73] L. A. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.

[Llo95] Seth Lloyd. Almost any quantum logic gate is universal. *Physical Review Letters*, 75:346–349, 1995.

[LTV98] M. Li, J. Tromp, and P. Vitanyi. Reversible simulation of irreversible computation. *Physica D*, 120:168–176, 1998.

[ME99] Michele Mosca and Artur Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. volume 1509 of *Lecture Notes in Computer Science*, 1999. Also available at Los Alamos archive, quant-ph/9903071.

[Mos98] Michele Mosca. Quantum searching and counting by eigenvector analysis. In *Proceedings of Randomized Algorithms, A satellite workshop of 23rd International Symposium on Mathematical Foundations of Computer Science*, 1998. Available at http://www.eccc.uni-trier.de/eccc-local/ECCC-LectureNotes/randalg/index.html.

[Mos99] Michele Mosca. Counting by quantum eigenvalue estimation. *Theoretical Computer Science*, 1999. to appear.

[MP95] M. Minsky and S. Papert. *Perceptrons.* MIT Press, second edition, 1995.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* discrete mathematics and its applications. CRC Press, London, 1997.

[Neu56] John von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies.* Princeton University Press, 1956.

[Nis91] N. Nisan. Crew prams and decision trees. *SIAM Journal of Computing*, 20(6):999–1007, 1991.

[NS94] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.

[NW99] Ashwin Nayak and Felix Wu. On the quantum black-box complexity of approximating the mean and related statistics. In *Proceedings of the 21th Annual ACM Symposium on Theory of Computing (STOC 99)*, 1999. Also available at Los Alamos archive, quant-ph/9804066.

[Pap94] C. H. Papadimitriou. Complexity theory, 1994.

[Pat92] R. Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 468–474, 1992.

[Pra75] Vaughan R. Pratt. Every prime has a succinct certificate. *SIAM Journal on Computing*, 4(3):214–220, 1975.

[RB98] Martin Rötteler and Thomas Beth. Polynomial-time solution to the hidden subgroup problem for a class of non-abelian groups. quant-ph 9812070, 1998.

[RC66] T. J. Rivlin and E. W. Cheney. A comparison of uniform approximations on an interval and a finite subset thereof. *SIAM Journal of Numerical Analysis*, 3(2):311–320, 1966.

[Rog87] Hartley Rogers. *Theory of Recursive Functions and Effective Computability.* MIT Press, 1987.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sch98] R. Schack. Using a quantum computer to investigate quantum chaos. *Physical Review A*, 57:1634–1635, 1998.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In Shafi Goldwasser, editor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society Press, November 1994. available at feynman.stanford.edu/qcomp/shor/index.html.

[Sho95a] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52, 1995.

[Sho95b] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. On the quant-ph archive, report no. 9508027., August 1995. Expanded version of [**Sho94**].

[Sho96] P. W. Shor. Fault tolerant quantum computation. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 56–65, Los Alamositos, CA, 1996. IEEE Computer Society Press.

[Sim94] Daniel R. Simon. On the power of quantum computation. In Shafi Goldwasser, editor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 116–123. IEEE Computer Society Press, November 1994. available at feynman.stanford.edu/qcomp/simon/index.html.

[Sol99] R. Solovay. Lie groups and quantum circuits, 1999. preprint.

[Ste96] A. M. Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77:793–797, 1996.

[Ste97] A. M. Steane. Active stabilisation, quantum computation, and quantum state synthesis. *Physical Review Letters*, 78:2252–2255, 1997.

[SV98] Leonard J. Schulman and Umesh Vazirani. Scalable nmr quantum computation. Technical Report 9804060, Los Alamos archive, 1998.

[Tap98] Alain Tapp, 1998. Personal communication.

[THL$^+$96] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble. Measurement of conditional phase shifts for quantum logic. *Physical Review Letters*, 76:3108, 1996.

[Vaz97] Umesh Vazirani, 1997. UC Berkeley Course CS294-2 Quantum Computation Fall 1997.

[vDDE+99] Wim van Dam, Mauro D'Ariano, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Estimation of local phase shifts on a quantum computer, 1999. in preparation.

[Wel88] Dominic Welsh. *Codes and Cryptography.* Oxford University Press, Oxford, 1988.

[Yao93] Andrew Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, pages 352–361, Los Alamitos, California, 1993. Institute of Electrical and Electronic Engineers Computer Society Press. available at feynman.stanford.edu/qcomp/yao/index.html.

[Zal98] Christof Zalka. Fast versions of shor's quantum factoring algorithm. Technical Report 9806084, Los Alamos archive, 1998.

[Zal99] Christof Zalka. preprint, 1999.