Quantum complexity theory*

Ethan Bernstein † Umesh Vazirani ‡

September 8, 1997

Abstract

In this paper we study quantum computation from a complexity theoretic viewpoint. Our first result is the existence of an efficient universal quantum Turing Machine in Deutsch's model of a quantum Turing Machine [20]. This construction is substantially more complicated than the corresponding construction for classical Turing Machines - in fact, even simple primitives such as looping, branching and composition are not straightforward in the context of quantum Turing Machines. We establish how these familiar primitives can be implemented, and also introduce some new, purely quantum mechanical primitives, such as changing the computational basis, and carrying out an arbitrary unitary transformation of polynomially bounded dimension.

We also consider the precision to which the transition amplitudes of a quantum Turing Machine need to be specified. We prove that $O(\log T)$ bits of precision suffice to support a T step computation. This justifies the claim that that the quantum Turing Machine model should be regarded as a discrete model of computation and not an analog one.

We give the first formal evidence that quantum Turing Machines violate the modern (complexity theoretic) formulation of the Church-Turing thesis. We show the existence of a problem, relative to an oracle, that can be solved in polynomial time on a quantum Turing Machine, but requires superpolynomial time on a bounded-error probabilistic Turing Machine; and thus not in the class **BPP**. The class **BQP**, of languages that are efficiently decidable (with small error-probability) on a quantum Turing Machine, satisfies: $\mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{P}^{\sharp\mathbf{P}}$. Therefore there is no possibility of giving a mathematical proof that quantum Turing Machines are more powerful than classical probabilistic Turing Machines (in the unrelativized setting) unless there is a major breakthrough in complexity theory.

1 Introduction

Just as the theory of computability has its foundations in the Church-Turing thesis, computational complexity theory rests upon a modern strengthening of this thesis, which asserts that any "reasonable" model of computation can be *efficiently* simulated on a probabilistic Turing Machine (an efficient simulation is one whose running time is bounded by some polynomial in the running time of the simulated machine). Here, we take reasonable to mean in principle physically realizable. Some models of computation, though interesting for other reasons, do not meet this criterion. For example, it is clear that computers that operate on arbitrary length words in unit time, or that exactly compute with infinite precision real numbers are

*A preliminary abstract of this work appeared as [11].

†Microsoft Corporation, One Microsoft Way, Redmond, WA 98052 (ethanb@microsoft.com). Research supported by N.S.F. Grant CCR-9310214.

‡Computer Science Division, University of California, Berkeley, CA 94720 (vazirani@cs.berkeley.edu). Research supported by N.S.F. Grant CCR-9310214

not realizable. It has been argued that the Turing Machine model (actually, the polynomial time equivalent cellular automaton model) is the inevitable choice once we assume that we can implement only finite precision computational primitives. Given the widespread belief that $\mathbf{NP} \not\subseteq \mathbf{BPP}$, this would seem to put a wide range of important computational problems (the $\mathbf{NP}$-hard problems) well beyond the capability of computers.

However, the Turing Machine fails to capture all physically realizable computing devices for a fundamental reason: the Turing Machine is based on a classical physics model of the Universe, whereas current physical theory asserts that the Universe is quantum physical. Can we get inherently new kinds of (discrete) computing devices based on quantum physics? Early work on the computational possibilities of quantum physics [6] asked the opposite question: does quantum mechanic's insistence on unitary evolution restrict the class of efficiently computable problems? They concluded that as far as deterministic computation is concerned, the only additional constraint imposed by quantum mechanics is that the computation must be reversible, and therefore by Bennett's [7] work it follows that quantum computers are at least as powerful as classical computers. The issue of the extra computational power of quantum mechanics over probabilistic computers was first raised by by Feynman [25] in 1982. In that paper, Feynman pointed out a very curious problem: the natural simulation of a quantum physical system on a probabilistic Turing Machine requires an exponential slowdown. Moreover, it is unclear how to carry out the simulation more efficiently. In view of Feynman's observation, we must re-examine the foundations of computational complexity theory, and the complexity-theoretic form of the Church-Turing thesis, and study the computational power of computing devices based on quantum physics.

A precise model of a quantum physical computer - hereafter referred to as the quantum Turing Machine - was formulated by Deutsch [20]. There are two ways of thinking about quantum computers. One way that may appeal to computer scientists is to think of a quantum Turing Machine as a quantum physical analogue of a probabilistic Turing Machine - it has an infinite tape and a transition function, and the actions of the machine are local and completely specified by this transition function. Unlike probabilistic Turing Machines, quantum Turing Machines allow branching with complex "probability amplitudes", but impose the further requirement that the machine's evolution be time-reversible. This view is elaborated in §3.2. Another way is to view a quantum computer as effecting a transformation in a space of complex superpositions of configurations. Quantum physics requires that this transformation be unitary. A quantum algorithm may then be regarded as the decomposition of a unitary transformation into a product of unitary transformations, each of which makes only simple local changes. This view is elaborated in §3.3. Both formulations play an important role in the study of quantum computation.

One important concern is whether quantum Turing Machines are really analog devices, since they involve complex transition amplitudes. It is instructive to examine the analogous question for probabilistic Turing Machines. There, one might worry that probabilistic machines are not discrete, and therefore not "reasonable", since they allow transition probabilities to be real numbers. However, there is extensive work showing that probabilistic computation can be carried out in a such a way that it is so insensitive to the transition probabilities that they can be allowed to vary arbitrarily in a large range [34, 44, 47]. In this paper, we show in a similar sense, that quantum Turing Machines are discrete devices: the transition amplitudes need only be accurate to $O(\log T)$ bits of precision to support T steps of computation. As Lipton [30] pointed out, it is crucial that the number of bits is $O(\log T)$ and not $O(T)$ (as it was in an early version of this paper), since k bits of precision requires pinning down the transition amplitude to one part in 2^k. Since the transition amplitude is some physical quantity such as the angle of a polarizer or the length of a π pulse, we must not assume that we can specify it to better than one part in some polynomial in T, and therefore the precision must be $O(\log T)$.

Another basic question one may ask is whether it is possible to define the notion of a general purpose quantum computer. In the classical case, this question is answered affirmatively by showing that there is an efficient universal Turing Machine. In this paper, we prove that there is an efficient quantum Turing Machine. When given as input the specification of an arbitrary QTM M, an input x to M, a time bound T, and an accuracy ϵ, the universal machine produces a superposition whose Euclidean distance from the time T superposition of M on x is at most ϵ. Moreover, the simulation time is bounded by a polynomial in T, $|x|$, and $1/\epsilon$. Deutsch [20] gave a different construction of a universal quantum Turing Machine. The simulation overhead in Deutsch's construction is exponential in T (the issue Deutsch was interested in was computability, not computational complexity). The structure of the efficient universal quantum Turing Machine constructed in this paper is very simple. It is just a deterministic Turing Machine with a single type of quantum operation — a quantum coin flip (an operation that performs a rotation on a single bit). The existence of this simple universal quantum Turing Machine has a bearing on the physical realizability of quantum Turing Machines in general, since it establishes that it is sufficient to physically realize a simple quantum operation on a single bit (in addition to maintaining coherence and carrying out deterministic operations, of course). Adleman, et. al. [1] and Solovay and Yao [40] have further clarified this point by showing that quantum coin flips with amplitudes 3/5 and 4/5 are sufficient for universal quantum computation.

Quantum computation is necessarily time reversible, since quantum physics requires unitary evolution. This makes it quite complicated to correctly implement even simple primitives such as looping, branching and composition. These are described in §4.2. In addition, we also require programming primitives, such as changing the computational basis, which are purely quantum mechanical. These are described in §5.1. Another important primitive is the ability to carry out any specified unitary transformation of polynomial dimension to a specified degree of accuracy. In §6 we show how to build a quantum Turing that implements this primitive. Finally, all these pieces are put together in §7 to construct the universal quantum Turing Machine.

We can still ask whether the quantum Turing Machine is the most general model for a computing device based on quantum physics. One approach to arguing affirmatively is to consider various other reasonable models, and to show that the quantum Turing Machine can efficiently simulate each of them. An earlier version of this work [11] left open the question of whether standard variants of a quantum Turing Machine, such as machines with multiple tapes or with modified tape access, are more powerful than the basic model. Yao [46] showed that these models are polynomially equivalent to the basic model, as are quantum circuits (which were introduced in [21]. The efficiency of Yao's simulation has been improved in [10] to show that the simulation overhead is a polynomial with degree independent of the number of tapes. Arguably, the full computational power of quantum physics for discrete systems is captured by the quantum analog of a cellular automaton. It is still an open question whether a quantum cellular automaton might be more powerful than a quantum Turing Machine (there is also an issue about the correct definition of a quantum cellular automaton). The difficulty has to do with decomposing a unitary transformation that represents many overlapping sites of activity into a product of simple, local unitary transformations. This problem has been solved in the special case of linearly bounded quantum cellular automata [24, 45].

Finally, several researchers have explored the computational power of quantum Turing Machines. Early work by Deutsch and Jozsa [22] showed how to exploit some inherently quantum mechanical features of QTMs. Their results, in conjunction with subsequent results by Berthiaume and Brassard [12, 13], established the existence of oracles under which there are computational problems that QTMs can solve in polynomial time with certainty, whereas if we require a classical probabilistic Turing machine to produce the correct answer with certainty, then it must take exponential time on some inputs. On the other hand, these computational problems are in **BPP** — the class of problems that can be solved in polynomial time by probabilistic Turing machines that are allowed to give the wrong answer with small probability. Since **BPP** is widely considered the class of efficiently computable problems, these results left open the question of whether quantum computers are more powerful than classical computers.

In this paper, we give the first formal evidence that quantum Turing Machines violate the modern form of the Church-Turing thesis, by showing that relative to an oracle there is a problem that can be solved in polynomial time on a quantum Turing Machine, but cannot be solved in $n^{o(\log n)}$ time on a probabilistic Turing Machine with any fixed error probability $< 1/2$. A detailed discussion about the implications of these oracle results is in the introduction to §8.4. Simon [39] subsequently strengthened our result in the time parameter by proving the existence of an oracle relative to which a certain problem can be solved in polynomial time on a quantum Turing Machine, but cannot be solved in less than $2^{n/2}$ steps on a probabilistic Turing Machine (Simon's problem is in $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ and therefore does not address the non-determinism issue). More importantly, Simon's paper also introduced an important new technique which was one of the ingredients in a remarkable result proved subsequently by Shor [36]. Shor gave polynomial time quantum algorithms for the factoring and discrete log problems. These two problems have been well-studied, and their presumed intractability forms the basis of much of modern cryptography. These results have injected a greater sense of urgency to the actual implementation of a quantum computer. The class $\mathbf{BQP}$, of languages that are efficiently decidable (with small error-probability) on a quantum Turing Machine, satisfies: $\mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{P}^{\sharp\mathbf{P}}$. This rules out the possibility of giving a mathematical proof that quantum Turing Machines are more powerful than classical probabilistic Turing Machines (in the unrelativized setting) unless there is a major breakthrough in complexity theory.

It is natural to ask whether quantum Turing Machines can solve every problem in NP in polynomial time. Bennett, Bernstein, Brassard and Vazirani [9] give evidence showing the limitations of quantum Turing Machines. They show that relative to an oracle chosen uniformly at random, with probability 1, the class $\mathbf{NP}$ cannot be solved on a quantum Turing Machine in time $o(2^{n/2})$. They also show that relative to a permutation oracle chosen uniformly at random, with probability 1, the class $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ cannot be solved on a quantum Turing Machine in time $o(2^{n/3})$. The former bound is tight since recent work of Grover [28] shows how to accept the class $\mathbf{NP}$ relative to any oracle on a quantum computer in time $O(2^{n/2})$.

Several designs have been proposed for realizing quantum computers [17], [23]. A number of authors have argued that there are fundamental problems in building quantum computers, most notably the effects of the decoherence of quantum superpositions, or the entanglement of the system with the environment. Very recently, there have been a sequence of important results showing how to implement quantum error-correcting codes and also use these codes to make quantum algorithms (quite) robust against the effects of decoherence [16] and [38].

Quantum computation touches upon the foundations of both computer science and quantum physics. The nature of quantum physics was clarified by the Einstein-Podolsky-Rosen paradox and Bell's inequalities (discussed in [25]) which demonstrate the difference between its statistical properties and those of any "classical" model. The computational differences between quantum and classical physics are if anything more striking, and can be expected to offer new insights into the nature of quantum physics. For example, one might naively argue that it is impossible to experimentally verify the exponentially large size of the Hilbert space associated with a discrete quantum system, since any observation leads to a collapse of its superposition. However, an experiment demonstrating the exponential speedup offered by quantum computation over classical computation would establish that something like the exponentially large Hilbert space must exist. Finally, it is important, as Feynman pointed out [25], to clarify the computational overhead required to simulate a quantum mechanical system. The simple form of the universal quantum Turing Machine constructed here – the fact that it has only a single non-trivial quantum operation defined on a single bit – suggests that even very simple quantum mechanical systems are capable of universal quantum computation, and are therefore hard to simulate on classical computers.

This paper is organized as follows. Section 2 introduces some of the mathematical machinery and notation we will use. In §3 we introduce the quantum Turing Machine as a natural extension of classical probabilistic Turing Machines. We also show that quantum Turing Machines need not be specified with an unreasonable amount of precision. In §4 and §5 we demonstrate the basic constructions which will allow us to build up large, complicated quantum Turing Machines in subsequent sections. Many actions which are quite easy for classical machines, such as completing a partially specified machine, running one machine after another, or repeating the operation of a machine a given number of times, will require non-trivial constructions for quantum Turing Machines. In §6, we show how to build a single quantum Turing Machine which can carry out any unitary transformation which is provided as input. Then, in §7, we use this simulation of unitary transformations to build a universal quantum computer. Finally, in §8 we give our results, both positive and negative, on the power of quantum Turing Machines.

2 Preliminaries

Let $\mathbf{C}$ denote the field of complex numbers. For $\alpha \in \mathbf{C}$, we denote by α^* its complex conjugate.

Let V be a vector space over $\mathbf{C}$. An inner-product over V is a complex function $(\cdot,\cdot)$ defined on $V \times V$ which satisfies

1. $\forall x \in V, (x,x) \geq 0$. Moreover $(x,x) = 0$ iff $x = 0$.
2. $\forall x,y,z \in V, (\alpha x + \beta y, z) = \alpha(x,z) + \beta(y,z)$.
3. $\forall x,y \in V, (x,y) = (y,x)^*$.

The inner-product yields a norm given by $\|x\| = (x,x)^{1/2}$. In addition to the triangle inequality $\|x+y\| \leq \|x\| + \|y\|$, the norm also satisfies the Schwarz inequality $\|(x,y)\| \leq \|x\|\|y\|$.

An *inner-product space* is a vector space V together with an inner-product $(\cdot,\cdot)$.

An inner-product space $\mathcal{H}$ over $\mathbf{C}$ is a *Hilbert space* if it is complete under the induced norm; where $\mathcal{H}$ is complete if every Cauchy sequence converges. i.e. if $\{x_n\}$ is a sequence with $x_n \in \mathcal{H}$, such that $\lim_{n,m\to\infty} \|x_n - x_m\| = 0$, then there is an x in $\mathcal{H}$ with $\lim_{n\to\infty} \|x_n - x\| = 0$.

Given any inner-product space V, each vector $x \in V$ defines a linear functional $x^* : V \to \mathbf{C}$ where $x^*(y) = (x,y)$. The set of such linear functionals is also an inner-product space, and will be referred to as the *vector-dual* of V and denoted V^*. In the case that V is a Hilbert space, V^* is called the *dual* of V, and is the set of all continuous linear functionals on V, and the dual space V^* is also a Hilbert space.

In Dirac's notation, a vector from an inner-product space V is identified using the "ket" notation $|\ \rangle$, with some symbol(s) placed inside to distinguish that vector from all others. We denote elements of the dual space using the "bra" notation $\langle\ |$. Thus the dual of $|\phi\rangle$ is $\langle\phi|$, and the inner product of vectors $|\psi\rangle$ and $|\phi\rangle$ which is the same as the result of applying functional $\langle\psi|$ to the vector $|\phi\rangle$, is denoted by $\langle\psi|\phi\rangle$.

Let U be a linear operator on V. In Dirac's notation, we denote the result of applying U to $|\phi\rangle$ as $U|\phi\rangle$. U also acts as a linear operator on the dual space V^* mapping each linear functional $\langle\phi|$ of the dual space to the linear functional which applies U followed by $\langle\phi|$. We denote the result of applying U to $\langle\phi|$ by $\langle\phi|U$.

For any inner-product space V, we can consider the usual vector space basis or Hamel basis: $\{|\theta_i\rangle\}_{i\in I}$. Every vector $|\phi\rangle \in V$ can be expressed as a finite linear combination of basis vectors. In the case that $\|\theta_i\| = 1$ and $(\langle\theta_i|, |\theta_j\rangle) = 0$ for $i \neq j$, we refer to the basis as an orthonormal basis. With respect to an orthonormal basis, we can write each vector $|\phi\rangle \in V$ as $|\phi\rangle = \sum_{i\in I} \alpha_i |\theta_i\rangle$, where $\alpha_i = \langle\theta_i|\phi\rangle$. Similarly each dual vector $\langle\phi| \in V^*$ can be written as as $\langle\phi| = \sum_{i\in I} \beta_i \langle\theta_i|$, where $\alpha_i = \langle\phi|\theta_i\rangle$. Thus each element $|\phi\rangle \in V$ can be thought of as a column vector of the α_i's, and each element $\langle\phi| \in V$ can be thought of as a row vector of the β_i's. Similarly each linear operator U may be represented by the set of *matrix elements* $\{\langle\theta_i|U|\theta_j\rangle\}_{i,j\in I}$, arranged in a "square" matrix with rows and columns both indexed by I. Then, the "column" of U with index i is the vector $U|\theta_i\rangle$ and the "row" of U with index i is the dual vector $\langle\theta_i|U$.

For a Hilbert space $\mathcal{H}$, $\{|\theta_i\rangle\}_{i\in I}$ is a *Hilbert space basis* for $\mathcal{H}$ if it is a maximal set of orthonormal vectors in $\mathcal{H}$. Every vector $|\phi\rangle \in \mathcal{H}$ can be expressed as the limit of a sequence of vectors, each of which can be expressed as a finite linear combination of basis vectors.

Given a linear operator U in an inner product space, if there is a linear operator U^* which satisfies $\langle U^*\phi|\psi\rangle = \langle\phi|U\psi\rangle$ for all ϕ, ψ, then U^* is called the *adjoint* or *Hermitian conjugate* of U. If a linear operator in an inner product space has an adjoint, it is unique. The adjoint of a linear operator in a Hilbert space or in a finite dimensional inner product space always exists. It is easy to see that if the adjoints of U_1 and U_2 exist then $(U_1 + U_2)^* = U_1^* + U_2^*$ and $(U_1 U_2)^* = U_2^* U_1^*$. An operator U is called *Hermitian* or *self-adjoint* if it is its own adjoint ($U^* = U$). The linear operator U is called *unitary* if its adjoint exists and satisfies $U^*U = UU^* = I$.

If we represent linear operators as "square" matrices indexed over an orthonormal basis, then U^* is represented by the conjugate transpose of U. So, in Dirac's notation we have the convenient identity $\langle\phi|U^*|\psi\rangle = (\langle\psi|U|\phi\rangle)^*$.

Recall that if the inner-product space V is the tensor product of two inner-product spaces V_1, V_2, then for each pair of vectors $|\phi_1\rangle \in V_1, |\phi_2\rangle \in V_2$ there is an associated tensor product $|\phi_1\rangle \otimes |\phi_2\rangle$ in V. In Dirac's notation, we denote $|\phi_1\rangle \otimes |\phi_2\rangle$ as $|\phi_1\rangle|\phi_2\rangle$.

The *norm* of U is defined as $\|U\| = \sup_{\||x\rangle\|=1} \|U|x\rangle\|$. A linear operator is called *bounded* if $\|U\|$ is finite. We will freely use the following standard facts about bounded linear operators:

$$\begin{aligned} \text{If } U^* \text{ exists then } \|U^*\| &= \|U\| & (1)\\ \|U_1 U_2\| &\leq \|U_1\|\|U_2\| & (2)\\ \|U_1\| - \|U_2\| &\leq \|U_1 + U_2\| \leq \|U_1\| + \|U_2\| & (3) \end{aligned}$$

Notice that a unitary operator U must satisfy $\|U\| = 1$. We will often use the following fact which tells us that if we approximate a series of unitary transformations with other unitary transformations, the error increases only additively.

Fact 2.0.1 *If U_1, U_1', U_2, U_2' are unitary transformations on an inner-product space then*

$$\|U_1' U_2' - U_1 U_2\| \leq \|U_1' - U_1\| + \|U_2' - U_2\|$$

This fact follows from Statements 3 and 2 above, since

$$\begin{aligned} \|U_1' U_2' - U_1 U_2\| &\leq \|U_1' U_2' - U_1 U_2'\| + \|U_1 U_2' - U_1 U_2\| \\ &\leq \|U_1' - U_1\|\|U_2'\| + \|U_1\|\|U_2' - U_2\| \end{aligned}$$

2.1 Miscellaneous notation

If d is a direction $\in \{L, R\}$, then $\bar{d}$ is the opposite of d.

Given two probability distributions $\mathcal{P}_1$ and $\mathcal{P}_2$ over the same domain I, the *total variation distance* between $\mathcal{P}_1$ and $\mathcal{P}_2$ is equal to $\frac{1}{2}\sum_{i\in I}|\mathcal{P}_1(i)-\mathcal{P}_2(i)|$.

We will refer to the cardinality of a set S as $\mathrm{card}(S)$ and the length of a string x as $|x|$.

3 Quantum Turing Machines

3.1 A physics-like view of randomized computation

Before we formally define a quantum Turing Machine (QTM), we introduce the necessary terminology in the familiar setting of probabilistic computation. As a bonus, we will be able to precisely locate the point of departure in the definition of a QTM.

Quantum mechanics makes a distinction between a system's evolution and its measurement. In the absence of measurement, the time evolution of a probabilistic TM can be described by a sequence of probability distributions. The distribution at each step gives the likelihood of each possible configuration of the machine. We can also think of the probabilistic TM as specifying an infinite dimensional stochastic matrix[1] M whose rows and columns are indexed by configurations. Each column of this matrix gives the distribution resulting from the corresponding configuration after a single step of the machine. If we represent the probability distribution at one time step by a vector $|v\rangle$, then the distribution at the next step is given by the product $M|v\rangle$. In quantum physics terminology, we call the distribution at each step a "linear superposition" of configurations, and we call the coefficient of each configuration (its probability) its "amplitude." The stochastic matrix is referred to as the "time evolution operator".

Three comments are in order. First, not every stochastic matrix has an associated probabilistic TM. Stochastic matrices obtained from probabilistic TM are finitely specified and map each configuration by making only local changes to it. Second, the support of the superposition can be exponential in the running time of the machine. Third, we need to constrain the entries allowed in the transition function of our probabilistic TM. Otherwise, it is possible to smuggle hard to compute quantities into the transition amplitudes, for instance by letting the i^{th} bit indicate whether the ith deterministic TM halts on a blank tape. A common restriction is to allow amplitudes only from the set $\{0, \frac{1}{2}, 1\}$. More generally, we might allow any real number in the interval $[0,1]$ which can be computed by some deterministic algorithm to within any desired 2^{-n} in time polynomial in n. It is easily shown that the first possibility is computationally no more restrictive than the second.

Returning to the evolution of the probabilistic TM, when we observe the machine after some number of steps, we do not see the linear superposition (probability distribution), but just a sample from it. If we "observe" the entire machine, then we see a configuration sampled at random according to the superposition. The same holds if we observe just a part of the machine. In this case, the superposition "collapses" to one that corresponds to the probability distribution conditioned on the value observed. By the linearity of the law of alternatives[2], the mere act of making observations at times earlier than t does not change the probability for each outcome in an observation at time t. So, even though the unobserved superposition may have support that grows exponentially with running time, we need only keep track of a constant amount of information when simulating a probabilistic TM which is observed at each step. The computational possibilities of quantum physics arise out of the fact that observing a quantum system changes its later behavior.

[1] Recall that a matrix is stochastic if it has non-negative real entries that sum to 1 in each column.

[2] The law of alternatives says exactly that the probability of an event A doesn't change if we first check to see whether event B has happened, $P(A) = P(A|B)P(B) + P(A|\overline{B})P(\overline{B})$

3.2 Defining a QTM

Our model of randomized computation is already surprisingly close to Deutsch's model of a QTM. The major change that is required is that in quantum physics, the amplitudes in a system's linear superposition and the matrix elements in a system's time evolution operator are allowed to be complex numbers rather than just positive reals. When an observation is made, the probability associated with each configuration is not the configuration's amplitude in the superposition, but rather the squared magnitude of its amplitude. So instead of always having a linear superposition whose entries sum to 1, we will now always have a linear superposition whose Euclidean length is 1. This means that QTMs must be defined so that their time evolution preserves the Euclidean length of superpositions.

Making these changes to our model, we arrive at the following definitions.

For completeness, let us recall the definition of a deterministic TM. There are many standard variations to the definition of deterministic TM, none of which affect their computational power. In this paper we will make our choices consistent with those in Deutsch's paper [20]: we consider TMs with a two-way infinite tape and a single tape head which must move left or right one square on each step. We also give standard definitions for interpreting the input, output, and running time of a deterministic TM. Note that although we usually restrict our discussion to TMs with tape head movements $\{L, R\}$, we will sometimes consider *generalized* TMs with tape head movements $\{L, N, R\}$ (where N means no head movement).

Definition 3.2.1 *A* deterministic Turing Machine *is defined by a triplet* (Σ, Q, δ) *where:* Σ *is a finite alphabet with an identified blank symbol* $\#$, Q *is a finite set of states with an identified initial state* q_0 *and final state* $q_f \neq q_0$, *and* δ, *the* deterministic transition function, *is a function*

$$\delta \; : \; Q \times \Sigma \rightarrow \Sigma \times Q \times \{L, R\}$$

The TM has a two-way infinite tape of cells indexed by $\mathbf{Z}$ *and a single read/write tape head that moves along the tape.*

A configuration *or* instantaneous description *of the TM is a complete description of the the contents of the tape, the location of the tape head, and the state* $q \in Q$ *of the finite control. At any time only a finite number of tape cells may contain non-blank symbols.*

For any configuration c *of TM* M, *the successor configuration* c' *is defined by applying the transition function to the current state* q *and currently scanned symbol* σ *in the obvious way. We write* $c \rightarrow_M c'$ *to denote that* c' *follows from* c *in one step.*

By convention, we require the initial configuration *of* M *to be satisfy the following conditions: the tape head is in cell 0, called the* start cell, *and the machine is in state* q_0. *An initial configuration has* input $x \in (\Sigma - \#)^*$ *if* x *is written on the tape in positions* $0, 1, 2, \ldots$ *and all other tape cells are blank. The TM* halts *on input* x *if it eventually enters the final state* q_f. *The number of steps a TM takes to halt on input* x *is its* running time *on input* x. *If a TM halts then its* output *is the string in* Σ^* *consisting of those tape contents from the leftmost non-blank symbol to the rightmost non-blank symbol, or the empty string if the entire tape is blank. A TM which halts on all inputs therefore computes a function from* $(\Sigma - \#)^*$ *to* Σ^*.

We now give a slightly modified version of the definition of a QTM provided by Deutsch [20]. As in the case of probabilistic TM, we must limit the transition amplitudes to efficiently computable numbers. Adleman, et. al. [1] and Solovay and Yao [40] have separately shown that further restricting QTMs to rational amplitudes does not reduce their computational power. In fact, they have shown that the set of amplitudes $\{0, \pm\frac{3}{5}, \pm\frac{4}{5}, 1\}$ are sufficient to construct a universal QTM. We give a definition of the computation of a QTM with a particular string as input, but we defer discussing what it means for a QTM to halt or give output until §3.5. Again, we will usually restrict our discussion to QTMs with tape head movements $\{L, R\}$, but will sometimes consider "generalized" QTMs with tape head movements $\{L, N, R\}$. As we pointed out in the introduction, unlike in the case of deterministic TMs, these choices do make a greater difference in the case of QTMs. This point is also discussed later in the paper.

Definition 3.2.2 *Call $\tilde{\mathbf{C}}$ the set consisting of $\alpha \in \mathbf{C}$ such that there is a deterministic algorithm that computes the real and imaginary parts of α to within 2^{-n} in time polynomial in n.*

A quantum Turing Machine *(QTM) M is defined by a triplet (Σ, Q, δ) where: Σ is a finite alphabet with an identified blank symbol $\#$, Q is a finite set of states with an identified initial state q_0 and final state $q_f \neq q_0$, and δ, the* quantum transition function*, is a function*

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

The QTM has a two-way infinite tape of cells indexed by $\mathbf{Z}$ and a single read/write tape head that moves along the tape. We define configurations, initial configurations, and final configurations exactly as for deterministic TMs.

Let $\mathcal{S}$ be the inner-product space of finite complex linear combinations of configurations of M with the Euclidean norm. We call each element $\phi \in \mathcal{S}$ a superposition *of M. The QTM M defines a linear operator $U_M : \mathcal{S} \rightarrow \mathcal{S}$, called the* time evolution operator *of M as follows: If M starts in configuration c with current state p and scanned symbol σ. Then after one step M will be in superposition of configurations $\psi = \sum_i \alpha_i c_i$, where each non-zero α_i corresponds to a transition $\delta(p, \sigma, \tau, q, d)$, and c_i is the new configuration that results from applying this transition to c. Extending this map to the entire space $\mathcal{S}$ through linearity gives the linear time evolution operator U_M.*

Note that we defined $\mathcal{S}$ by giving an orthonormal basis for it: namely the configurations of M. In terms of this standard basis, each superposition $\psi \in \mathcal{S}$ may be represented as a vector of complex numbers indexed by configurations. The time evolution operator U_M may be represented by the (countable dimensional) "square" matrix with columns and rows indexed by configurations where the matrix element from column c and row c' gives the amplitude with which configuration c leads to configuration c' in a single step of M.

For convenience, we will overload notation and use the expression $\delta(p, \sigma, \tau, q, d)$ to denote the amplitude in $\delta(p, \sigma)$ of $|\tau\rangle|q\rangle|d\rangle$.

The next definition provides an extremely important condition that QTMs must satisfy to be consistent with quantum physics. We have introduced this condition in the form stated below for expository purposes. As we shall see later (in §3.3), there are other equivalent formulations of this condition that are more familiar to quantum physics.

Definition 3.2.3 *We will say that M is* well-formed *if the its time evolution operator U_M preserves Euclidean length.*

Well-formedness is a necessary condition for a QTM to be consistent with quantum physics. As we shall see in the next subsection, well-formedness is equivalent to unitary time evolution, which is a fundamental requirement of quantum physics.

Next, we define the rules for observing the QTM M. For those familiar with quantum mechanics, we should state that the definition below restricts the measurements to be in the computational basis of $\mathcal{S}$. This is because the actual basis in which the measurement is performed must be efficiently computable, and therefore we may, without loss of generality, perform the rotation to that basis during the computation itself.

Definition 3.2.4 *When QTM M in superposition $\psi = \sum_i \alpha_i c_i$ is observed or measured, configuration c_i is seen with probability $|\alpha|^2$. Moreover, the superposition of M is updated to $\psi' = c_i$.*

We may also perform a partial measurement, *say only on the first cell of the tape. In this case, suppose that the first cell may contain the values 0 or 1, and suppose the superposition was $\psi = \sum_i \alpha 0_i c0_i + \sum_i \alpha 1_i c1_i$, where the $c0_i$ are those configurations that have a 0 in the first cell, and $c1_i$ are those configurations that have a 1 in the first cell. Measuring the first cell results in $Pr[0] = \sum_i |\alpha 0_i|^2$. Moreover, if a 0 is observed, the new superposition is given by $\frac{1}{\sqrt{Pr[0]}} \sum_i \alpha 0_i c0_i$. i.e. the part of the superposition consistent with the answer, with amplitudes scaled to give a unit vector.*

Note that the well-formedness condition on a QTM simply says that the the time evolution operator of a QTM must satisfy the condition that in each successive superposition, the sum of the probabilities of all possible configurations must be 1.

Notice that a QTM differs from a classical TM in that the "user" has decisions beyond just choosing an input. A priori it is not clear whether multiple observations might increase the power of QTMs This point is discussed in more detail in [10], and there it is shown that one may assume without loss of generality that the QTM is only observed once. Therefore, in this paper, we shall make simplifying assumptions about the measurement of the final result of the QTM. The fact that these assumptions do not result in any loss of generality follows from the results in [10].

In general, the "output" of a QTM is a sample from a probability distribution. We can regard two QTMs as functionally equivalent, for practical purposes, if their output distributions are sufficiently close to each other. A formal definition of what it means for one QTM to simulate another is also given in [10]. As in the case of classical TMs, the formal definition is quite unwieldy. In the actual constructions, it will be easy to see in what sense they are simulations. Therefore we will not replicate the formal definitions from [10] here. We give a more informal definition below:

Definition 3.2.5 *We say that QTM M' simulates M with slowdown f with accuracy ϵ, if the following holds: let $\mathcal{D}$ be a distribution such that observing M on input x after T steps produces a sample from $\mathcal{D}$. let $\mathcal{D}'$ be a distribution such that observing M' on input x after $f(T)$ steps produces a sample from $\mathcal{D}'$. Then we say that M' simulates M with accuracy ϵ if $|\mathcal{D} - \mathcal{D}'| \leq \epsilon$.*

We will sometimes find it convenient to measure the accuracy of a simulation by calculating the Euclidean distance between the target superposition and the superposition achieved by the simulation. The following shows that the variation distance between the resulting distributions is at most 4 times this Euclidean distance.

Lemma 3.2.6 *Let $\phi, \psi \in \mathcal{S}$ such that $\|\phi\| = \|\psi\| = 1$, and $\|\phi - \psi\| \leq \epsilon$. Then the total variation distance between the probability distributions resulting from measurements of ϕ and ψ is at most 4ϵ.*

Proof. Let $\phi = \sum_i \alpha_i |i\rangle$ and $\psi = \sum_i \beta_i |i\rangle$. Observing ϕ gives each $|i\rangle$ with probability $|\alpha_i|^2$, while observing ψ gives each $|i\rangle$ with probability $|\beta_i|^2$. Let $\pi = \phi - \psi = \sum_i (\alpha_i - \beta_i)|i\rangle$. Then the latter probability, $|\beta_i|^2$, can be expressed as:

$$\beta_i \beta_i^* = (\alpha_i + \gamma_i)(\alpha_i + \gamma_i)^* = |\alpha_i|^2 + |\gamma_i|^2 + \alpha_i \gamma_i^* + \gamma_i \alpha_i^*$$

Therefore, the total variation distance between these two distributions is at most

$$\sum_i \|\gamma_i\|^2 + |\alpha_i \gamma_i^*| + |\gamma_i \alpha_i^*| \leq \sum_i |\gamma_i|^2 + \langle \gamma || \alpha \rangle + \langle \alpha || \gamma \rangle \leq \epsilon^2 + 2\|\alpha\|\|\gamma\| \leq \epsilon^2 + 2\epsilon$$

Finally, note that since we have unit superpositions, we must have $\epsilon \leq 2$. □

3.3 Quantum computing as a unitary transformation

In the preceding sections, we introduced QTMs as an extension of the notion of probabilistic TMs. We stated there that a QTM is well-formed if it preserves the norm of the superpositions. In this section, we explore a different, and extremely useful, alternative view of QTMs: in terms of properties of the time evolution operator. We prove that a QTM is well-formed iff its time evolution is unitary. Indeed unitary time evolution is a fundamental constraint imposed by quantum mechanics, and we chose to state the well-formedness condition in the last section mainly for expository purposes.

Understanding unitary evolution from an intuitive point of view is quite important to comprehending the computational possibilities of quantum mechanics. Let us explore this in the setting of a quantum mechanical system that consists of n parts each of which can be in one of two states, labeled $|0\rangle$ and $|1\rangle$ (these could be n particles, each with a spin state). If this were a classical system, then at any given instant it would be in a single configuration which could be described by n bits. However, in quantum physics, the system is allowed to be in a linear superposition of configurations, and indeed the instantaneous state of the system is described by a unit vector in the 2^n dimensional vector space, whose basis vectors correspond to all the 2^n configurations. Therefore to describe the instantaneous state of the system, we must specify 2^n complex numbers. The implications of this are quite extraordinary: even for a small system consisting of 200 particles, nature must keep track of 2^{200} complex numbers just to 'remember' its instantaneous state. Moreover, it must update these numbers at each instant to evolve the system in time. This is an extravagant amount of effort, since 2^{200} is larger than the standard estimates on the number of particles in the visible universe. So if nature puts in such extravagant amounts of effort to evolve even a tiny system at the level of quantum mechanics, it would make sense that we should design our computers to take advantage of this.

However, unitary evolution and the rules for measurement in quantum mechanics place significant constraints on how these features can be exploited for computational purposes. One of the basic primitives that allows these features to be exploited while respecting the unitarity constraints is the discrete fourier transform — this is described in more detail in §8.4. Here we consider some very simple cases: One interesting phenomenon supported by unitary evolution is the interference of computational paths. In a probabilistic computation the probability of moving from one configuration to another is the sum of the probabilities of each possible path from the former to the latter. The same is true of the probability amplitudes in quantum computation, but not necessarily of the probabilities of observations. Consider for example applying the transformation $U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$ twice in sequence to the same tape cell which at first contains the symbol 0. If we observe the cell after the first application of U, we see either symbol with probability 1/2. If we then observe after applying U a second time, the symbols are again equally likely. However, since U^2 is the identity, if we only observe at the end, we see a 0 with probability 1. So, even though there are two computational paths that lead from the initial 0 to a final symbol 1, they interfere destructively cancelling each other out. The two paths leading to a 0 interfere constructively so that even though both have probability 1/4 when we observe twice we have probability 1 of reaching 0 if we observe only once. Such a boosting of probabilities, by an exponential factor, lies at the heart of the QTM's advantage over a probabilistic TM in solving the Fourier sampling problem.

Another constraint inherent in computation using unitary transformations is reversibility. We show in §4.2 that for any QTM M there is a corresponding QTM M^R, whose time evolution operator is the conjugate transpose of the time evolution operator of M, and therefore undoes the actions of M. Sections 3.5 and 4.1 are devoted to defining the machinery to deal with this feature of QTMs.

We prove below in Appendix A that a QTM is well-formed if and only if its time evolution operator is unitary. This establishes that our definition (which did not mention unitary evolution for expository purposes) does satisfy the fundamental constraint imposed by quantum mechanics — unitary evolution. Actually one could still ask why this is consistent with quantum mechanics, since the space $\mathcal{S}$ of *finite* linear combinations of configurations is not a Hilbert space since it is not complete. To see that this doesn't present a problem, notice that $\mathcal{S}$ is a dense subset of the Hilbert space $\mathcal{H}$ of all (not just finite) linear combinations of configurations. Moreover, any unitary operator U on $\mathcal{S}$ has a unique extension $\hat{U}$ to $\mathcal{H}$; and $\hat{U}$ is unitary and its inverse is $\hat{U}^*$. The proof is quite simple. Let $\hat{U}$ and $\hat{U}^*$ be the continuous extensions of U and U^* to $\mathcal{H}$. Let $x, y \in \mathcal{H}$. Then there are sequences $\{x_n\}, \{y_n\} \in \mathcal{S}$ such that $x_n \to x$ and $y_n \to y$. Moreover, for all n, $(Ux_n, y_n) = (x_n, U^* y_n)$. Taking limits, we get that $(\hat{U}x, y) = (x, \hat{U}^* y)$, as desired.

As an aside, we should briefly mention that another resolution of this issue is achieved by following Feynman [26], who suggested that if we use a quantum mechanical system with Hamiltonian $U + U^*$, then the resulting system has a local, time invariant Hamiltonian. It is easy to probabilistically recover the computation of the original system from the computation of the new one.

It is interesting to note that the following theorem would not be true if we defined QTMs using a one-way infinite tape. In that case, the trivial QTM which always moves its tape head right would be well-formed, but its time evolution would not be unitary since it's start configuration could not be reached from any other configuration.

Theorem A.0.12 *A QTM is well-formed iff its time evolution operator is unitary.*

3.4 Precision required in a QTM

One important concern is whether QTMs are really analog devices, since they involve complex transition amplitudes. The issue here is how accurately these transition amplitudes must be specified to ensure that the correctness of the computation is not compromised. In an earlier version of this paper, we showed that $T^{O(1)}$ bits of precision are sufficient to correctly carry out T steps of computation to within accuracy ϵ for any constant ϵ. Lipton [30] pointed out that for the device to be regarded as a discrete device, we must require that its transition amplitudes be specified to at most one part in $T^{O(1)}$ (as opposed to accurate to within $T^{O(1)}$ bits of precision). This is because the transition amplitude represents some physical quantity such as the angle of a polarizer or the length of a π pulse, and we must not assume that we can specify it to better than one part in some polynomial in T, and therefore the number of bits of precision must be $O(\log T)$. This is exactly what we proved shortly after Lipton's observation, and we present that proof in this section.

The next theorem shows that because of the unitary time evolution errors in the superposition made over a sequence of steps will, in the worst case, only add.

Theorem 3.4.1 *Let U be the time evolution operator of a QTM M and $T > 0$. If $|\phi_0\rangle, |\tilde{\phi}_0\rangle, \ldots, |\phi_T\rangle, |\tilde{\phi}_T\rangle$ are superpositions of U such that*

$$\begin{aligned} \| |\phi_i\rangle - |\tilde{\phi}_i\rangle \| &\leq \epsilon \\ |\phi_i\rangle &= U|\tilde{\phi}_{i-1}\rangle \end{aligned}$$

then $\| |\tilde{\phi}_T\rangle - U^T|\phi_0\rangle \| \leq T\epsilon$.

Proof. Let $|\psi_i\rangle = |\tilde{\phi}_i\rangle - |\phi_i\rangle$. Then we have

$$|\tilde{\phi}_T\rangle = U^T|\phi_0\rangle + U^T|\psi_0\rangle + U^{T-1}|\psi_1\rangle + \cdots + |\psi_T\rangle$$

The theorem follows by the triangle inequality since U is unitary and $\||\psi_i\rangle\| \leq \epsilon$. $\square$

Definition 3.4.2 *We say that QTMs M and M' are* ϵ-close *if they have the same state set and alphabet and if the difference between each pair of corresponding transition amplitudes has magnitude at most ϵ. Note that M and M' can be ϵ-close even if one or both are not well-formed.*

The following theorem shows that two QTMs which are close in the above sense give rise to time evolution operators which are close to each other, even if the QTMs are not well-formed. As a simple consequence, the time evolution operator of a QTM is always bounded, even if the QTM is not well-formed.

Theorem 3.4.3 *If QTMs M and M' with alphabet Σ and state set Q are ϵ-close, then the difference in their time evolutions has norm at most $2\ card(\Sigma)\ card(Q)\epsilon$. Moreover, this statement holds even if one or both of the machines are not well-formed.*

Proof. Let QTMs M and M' with alphabet Σ and state set Q be given which are ϵ-close. Let U be the time evolution of M, and let U' be the time evolution of M'.

Now, consider any unit length superposition of configurations $|\phi\rangle = \sum_j \alpha_j |c_j\rangle$. Then we can express the difference in the machines' operation on $|\phi\rangle$ as follows.

$$U|\phi\rangle - U'|\phi\rangle = \sum_j \left(\sum_{i \in P(j)} (\lambda_{i,j} - \lambda'_{i,j}) \alpha_i \right) |c_j\rangle$$

where $P(j)$ is the set of i such that configuration c_i can lead to c_j in a single step of M or M', and where $\lambda_{i,j}$ and $\lambda'_{i,j}$ are the amplitudes with which c_i leads to c_j in M and M'.

Applying the triangle inequality and the fact that the square of the sum of n reals is at most n times the sum of their squares, we have

$$\begin{aligned} \|U|\phi\rangle - U'|\phi\rangle\|^2 &= \textstyle\sum_j \left| \sum_{i \in P(j)} (\lambda_{i,j} - \lambda'_{i,j}) \alpha_i \right|^2 \\ &\leq \textstyle\sum_j 2\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q) \sum_{i \in P(j)} \left| (\lambda_{i,j} - \lambda'_{i,j}) \alpha_i \right|^2 \end{aligned}$$

Then since M and M' are ϵ-close, we have

$$\begin{aligned} \textstyle\sum_j 2\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q) \sum_{i \in P(j)} & \left| (\lambda_{i,j} - \lambda'_{i,j}) \alpha_i \right|^2 \\ &= 2\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q) \textstyle\sum_j \sum_{i \in P(j)} \left| \lambda_{i,j} - \lambda'_{i,j} \right|^2 |\alpha_i|^2 \\ &\leq 2\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q) \epsilon^2 \textstyle\sum_j \sum_{i \in P(j)} |\alpha_i|^2 \end{aligned}$$

Finally since for any configuration c_j, there are at most $2\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q)$ configurations that can lead to c_j in a single step, we have

$$\begin{aligned} 2\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q) \epsilon^2 \textstyle\sum_j \sum_{i \in P(j)} |\alpha_i|^2 &\leq 4\ \mathrm{card}(\Sigma)^2\ \mathrm{card}(Q)^2 \epsilon^2 \textstyle\sum_i |\alpha_i|^2 \\ &= 4\ \mathrm{card}(\Sigma)^2\ \mathrm{card}(Q)^2 \epsilon^2 \end{aligned}$$

Therefore, for any unit length superposition $|\phi\rangle$

$$\|(U - U')|\phi\rangle\| \leq 2\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q) \epsilon$$

$\square$

The following corollary shows that $O(\log T)$ bits of precision are sufficient in the transition amplitudes to simulate T steps of a QTM to within accuracy ϵ for any constant ϵ.

Corollary 3.4.4 *Let $M = (\Sigma, Q, \delta)$ be a well-formed QTM. and let M' be a QTM which is $\frac{\epsilon}{24\ card(\Sigma)\ card(Q)T}$-close to M, where $\epsilon > 0$. Then M' simulates M for time T with accuracy ϵ. Moreover, this statment holds even if M' is not well-formed.*

Proof. Let $b = \frac{1}{24\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q)T}$. Without loss of generality, we further assume $\epsilon < 1/2$.

Consider running M and M' with the same initial superposition. Since M is well-formed, by Theorem A.0.12, its time evolution operator U is unitary. By Theorem 3.4.3 on page 13 the time evolution operator of M', U', is within $\delta = \frac{\epsilon}{12T}$ of U.

Applying U' can always be expressed as applying U and then adding a perturbation of length at most δ times the length of the current superposition. So, the length of the superposition of U' at time t is at most $(1+\delta)^t$. Since $\delta \leq 1/T$, this length is at most e. Therefore, appealing to Theorem 3.4.1 above, the difference between the superpositions of M and M' at time T is a superposition of norm at most $3\delta T \leq \frac{\epsilon}{4}$. Finally, Lemma 3.2.6 on page 10 tells us that observing M' at time T gives a sample from a distribution which is within total variation distance ϵ of the distributions sampled from by observing M at time T. $\square$

3.5 Input/output conventions for QTMs

Timing is of crucial importance to the operation of a QTM, because computational paths can only interfere if they take the same number of time steps. Equally important are the position of the tape head and alignment of the tape contents. In this subsection, we introduce several input/output conventions on QTMs and deterministic TMs which will help us maintain these relationships while manipulating and combining machines.

We would like to think of our QTMs as finishing their computation when they reach the final state q_f. However, it is unclear how we should regard a machine which reaches a superposition in which some configurations are in state q_f but others are not. We try to avoid such difficulties by saying that a QTM halts on a particular input if it reaches a superposition consisting entirely of configurations in state q_f.

Definition 3.5.1 *A* final configuration *of a QTM is any configuration in state q_f. If when QTM M is run with input x, at time T the superposition contains only final configurations and at any time less than T the superposition contains no final configuration, then M* halts *with* running time *T on input x. The superposition of M at time T is called the* final superposition *of M run on input x. A polynomial-time QTM is a well-formed QTM which on every input x halts in time polynomial in the length of x.*

We would like to define the output of a QTM which halts as the superposition of the tape contents of the configurations in the machine's final superposition. However, we must be careful to note the position of the tape head and the alignment relative to the start cell in each configuration since these details determine whether later paths interfere. Recall that the output string of a final configuration of a TM is its tape contents from the leftmost non-blank symbol to the rightmost non-blank symbol. This means that giving an output string leaves unspecified the alignment of this string on the tape and the location of the tape head to be identified. When describing the input/output behavior of a QTM we will sometimes describe this additional information. When we do not, the additional information will be clear from context. For example, we will often build machines in which all final configurations have the output string beginning in the start cell with the tape head scanning its first symbol.

Definition 3.5.2 *A QTM is called* well-behaved *if it halts on all input strings in a final superposition where each configuration has the tape head in the same cell. If this cell is always the start cell, we call the machine* stationary*. Similarly, a deterministic TM is called stationary if it halts on all inputs with its tape head back in the start cell.*

To simplify our constructions, we will often build QTMs and then combine them in simple ways, like running one after the other or iterating one a varying number of times. To do so we must be able to add new transitions into the initial state q_0 of a machine. However, since there may already be transitions into q_0, the resulting machine may not be reversible. But, we can certainly redirect the transitions out of the final state q_f of a reversible TM or a well-behaved QTM without affecting its behavior. Note that for a well-formed QTM, if q_f always leads back to q_0, then there can be no more transitions into q_0. In that case, redirecting the transitions out of q_f will allow us to add new ones into q_0 without violating reversibility. We will say that a machine with this property is in *normal form.* Note that a well-behaved QTM in normal form always halts before using any transition out of q_f and therefore also before using any transition into q_0. This means that altering these transitions will not alter the relevant part of the machine's computation. For simplicity, we arbitrarily define normal form QTMs to step right and leave the tape unchanged as they go from state q_f to q_0.

Definition 3.5.3 *A QTM or deterministic TM $M = (\Sigma, Q, \delta)$ is in* normal form *if*

$$\forall \sigma \in \Sigma \quad \delta(q_f, \sigma) = |\sigma\rangle|q_0\rangle|R\rangle$$

We will need to consider QTMs with the special property that any particular state can be entered while the machine's tape head steps in only one direction. Though not all QTMs are "unidirectional", we will show that any QTM can be efficiently simulated by one that is. Unidirectionality will be a critical concept in reversing a QTM, in completing a partially described QTM, and in building our universal QTM. We further describe the advantages of unidirectional machines after Theorem 5.2.2 in §5.2.

Definition 3.5.4 *A QTM $M = (\Sigma, Q, \delta)$ is called* unidirectional *if each state can be entered from only one direction: In other words, if $\delta(p_1, \sigma_1, \tau_1, q, d_1)$ and $\delta(p_2, \sigma_2, \tau_2, q, d_2)$ are both non-zero, then $d_1 = d_2$.*

Finally, we will find it convenient to use the common tool of thinking of the tape of a QTM or deterministic TM as consisting of several tracks.

Definition 3.5.5 *A* multi-track *TM with k tracks is a TM whose alphabet Σ is of the form $\Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_k$ with a special blank symbol # in each Σ_i so that the blank in Σ is $(\#, \ldots, \#)$. We specify the input by specifying the string on each "track", and optionally by specifying the alignment of the contents of the tracks. So, a TM run on input $x_1; x_2; \ldots; x_k \in \Pi_{i=1}^{k}(\Sigma_i - \#)^*$ is started in the (superposition consisting only of the) initial configuration with the non-blank portion of the i^{th} coordinate of the tape containing the string x_i starting in the start cell. More generally, on input $x_1|y_1; x_2|y_1; \ldots; x_k|y_k$ with $x_i, y_i \in \Sigma_i^*$ the non-blank portion of the i^{th} track is $x_i y_i$ aligned so that the first symbol of each y_i is in the start cell. Also, input $x_1; x_2; \ldots; x_k$ with $x_{l+1}, \ldots, x_k = \epsilon$ is abbreviated as $x_1; x_2; \ldots; x_l$.*

4 Programming a QTM

In this section we explore the fundamentals of building up a QTM from several simpler QTMs. Implementing basic programming primitives, such as looping, branching, and reversing a computation is straightforward for deterministic TMs. However, these constructions are more difficult for QTMs because one must be very careful to maintain reversibility. In fact, the same difficulties arise when building reversible deterministic TMs. However, building reversible TMs up out of simpler reversible TMs has never been necessary. This is because Bennett [7] showed how to efficiently simulate any deterministic TM with one which is reversible. So, one can build a reversible TM by first building the desired computation with a non-reversible machine, and then using Bennett's construction. None of the constructions in this section make any special use of the quantum nature of QTMs, and in fact all techniques used are the same as those required to make the analogous construction for reversible TMs.

We will show in this section that reversible TMs are a special case of QTMs. So, as Deutsch [20] noted, Bennett's result allows any desired deterministic computation to be carried out on a QTM. However, Bennett's result is not sufficient to allow us to use deterministic computations when building up QTMs, because the different computation paths of a QTM will only interfere properly provided that they take exactly the same number of steps. We will therefore carry out a modified version of Bennett's construction to show that any deterministic computation can be carried out by a reversible TM whose running time depends only on the length of its input. Then, different computation paths of a QTM will take the same number of steps provided that they carry out the same deterministic computation on inputs of the same length.

4.1 Reversible Turing Machines

Definition 4.1.1 *A* reversible Turing Machine *is a deterministic TM for which each configuration has at most one predecessor.*

Note that we have altered the definition of a reversible TM from the one used by Bennett [7, 8] so that our reversible TMs are a special case of our QTMs. First, we have restricted our reversible TM to move its head only left and right, instead of also allowing it to stay still. Second, we insist that the transition function δ be a complete function rather than a partial function. Finally, we consider only reversible TMs with a single tape, though Bennett worked with multi-tape machines.

Theorem 4.1.2 *Any reversible TM is also a well-formed QTM.*

Proof. The transition function δ of a deterministic TM maps the current state and symbol to an update triple. If we think of it as instead giving the unit superposition with amplitude 1 for that triple and 0 elsewhere, then δ is also a quantum transition function and we have a QTM. The time evolution matrix corresponding to this QTM contains only the entries 1 and 0. Since Corollary B.0.14 proven below in Appendix B tells us that each configuration of a reversible TM has exactly one predecessor, this matrix must be a permutation matrix. If the TM is reversible then there must be at most one 1 in each row. Therefore any superposition of configurations $\sum_i \alpha_i|c_i\rangle$ is mapped by this time evolution matrix to some other superposition of configurations $\sum_i \alpha_i|c_i'\rangle$. So, the time evolution preserves length, and the QTM is well-formed. □

Previous work of Bennett shows that reversible machines can efficiently simulate deterministic TMs. Of course, if a deterministic TM computes a function which is not one-to-one, then no reversible machine can simulate it exactly. Bennett [7] showed that a generalized reversible TM can do the next best thing, which is to take any input x and compute $x; M(x)$ where $M(x)$ is the output of M on input x. He also showed that if a deterministic TM computes a function which is one-to-one, then there is a generalized reversible TM that computes the same function. For both constructions, he used a multi-tape TM and also suggested how the simulation could be carried out using only a single-tape machine. Morita et.al. [33] use Bennett's ideas, and some further techniques, to show that any deterministic TM can be simulated by a generalized reversible TM with a two symbol alphabet.

We will give a slightly different simulation of a deterministic TM with a reversible machine that preserves an important timing property.

First, we describe why timing is of critical importance. In later sections, we will build QTMs with interesting and complex interference patterns. However, two computational paths can only interfere if they reach the same configuration at the same time. We will often want paths to interfere which run much of the same computation, but with different inputs. We can only be sure they interfere if we know that these computations can be carried out in exactly the same running time. We therefore want to show that any function computable in deterministic polynomial time can be computed by a polynomial time reversible TM in such a way that the running time of the latter is determined entirely by the length of its input. Then, provided that all computation paths carry out the same deterministic algorithms on the inputs of the same length, they will all take exactly the same number of steps.

We prove the following theorem in Appendix B on page 55 using ideas from the constructions of Bennett and Morita et.al.

Theorem 4.1.3 (Synchronization Theorem) *If f is a function mapping strings to strings which can be computed in deterministic polynomial time and such that the length of $f(x)$ depends only on the length of x, then there is a polynomial time, stationary, normal form reversible TM which given input x, produces output $x; f(x)$, and whose running time depends only on the length of x.*

If f is a function from strings to strings that such that both f and f^{-1} can be computed in deterministic polynomial time, and such that the length of $f(x)$ depends only on the length of x, then there is a polynomial time, stationary, normal form reversible TM which given input x, produces output $f(x)$, and whose running time depends only on the length of x.

4.2 Programming primitives

We now show how to carry out several programming primitives reversibly. The Branching, Reversal, and Looping Lemmas will be used frequently in subsequent sections.

The proofs of the following two lemmas are straightforward and are omitted. However, they will be quite useful as we build complicated machines, since they allow us to build a series of simpler machines while ignoring the contents of tracks not currently being used.

Lemma 4.2.1 *Given any QTM (reversible TM) $M = (\Sigma, Q, \delta)$ and any set Σ', there is a QTM (reversible TM) $M' = (\Sigma \times \Sigma', Q, \delta')$ such that M' behaves exactly as M while leaving its second track unchanged.*

Lemma 4.2.2 *Given any QTM (reversible TM) $M = (\Sigma_1 \times \cdots \times \Sigma_k, Q, \delta)$ and permutation $\pi : [1, k] \to [1, k]$, there is a QTM (reversible TM) $M' = (\Sigma_{\pi(1)} \times \cdots \times \Sigma_{\pi(k)}, Q, \delta')$ such that the M' behaves exactly as M except that its tracks are permuted according to π.*

The following two lemmas are also straightforward, but stating them separately makes Lemma 4.2.5 below easy to prove. The first deals with swapping transitions of states in a QTM. We can swap the outgoing transitions of states p_1 and p_2 for transition function δ by defining $\delta'(p_1, \sigma) = \delta(p_2, \sigma)$, $\delta'(p_2, \sigma) = \delta(p_1, \sigma)$ and $\delta'(p, \sigma) = \delta(p, \sigma)$ for $p \neq p_1, p_2$. Similarly, we can swap the incoming transitions of states q_1 and q_2 by defining $\delta'(p, \sigma, \tau, q_1, d) = \delta(p, \sigma, \tau, q_2, d)$, $\delta'(p, \sigma, \tau, q_2, d) = \delta(p, \sigma, \tau, q_1, d)$, and $\delta'(p, \sigma, \tau, q, d) = \delta(p, \sigma, \tau, q, d)$ for $q \neq q_1, q_2$.

Lemma 4.2.3 *If M is a well-formed QTM (reversible TM), then swapping the incoming or outgoing transitions between a pair of states in M gives another well-formed QTM (reversible TM).*

Lemma 4.2.4 *Let $M_1 = (\Sigma, Q_1, \delta_1)$ and $M_2 = (\Sigma, Q_2, \delta_2)$ be two well-formed QTMs (reversible TMs) with the same alphabet and disjoint state sets. Then then the union of the two machines, $M = (\Sigma, Q_1 \cup Q_2, \delta_1 \cup \delta_2)$ and with arbitrarily chosen start state $q_0 \in Q_1 \cup Q_2$ is also a well-formed QTM (reversible TM).*

Using the two preceding lemmas, we can insert one machine in place of a state in another. When we perform such an insertion, the computations of the two machines might disturb each other. However, sometimes this can easily be seen not to be the case. For example, in the insertion used by the Dovetailing Lemma below, we will insert one machine for the final state of a well-behaved QTM, so that the computation of the original machine has completed before the inserted machine begins. In the rest of the insertions we carry out the two machines will operate on different tracks, so the only possible disturbance involves the position of the tape head.

Lemma 4.2.5 *If M_1 and M_2 are normal form QTMs (reversible TMs) with the same alphabet, and q is a state of M_1, then there is a normal form QTM M which acts as M_1 except that each time it would enter state q, it instead runs machine M_2.*

Proof. Let M_1 and M_2 be as stated with initial and final states $q_{1,0}, q_{2,0}, q_{1,f}, q_{2,f}$, and with q a state of M_1.

Then we can construct the desired machine M as follows. First, take the union of M_1 and M_2 according to Lemma 4.2.4 on page 18 and make the start state $q_{1,0}$ if $q \neq q_{1,0}$ and $q_{2,0}$ otherwise, and make the final state $q_{1,f}$ if $q \neq q_{1,f}$ and $q_{2,f}$ otherwise. Then, swap the incoming transitions of q and $q_{2,0}$ and the outgoing transitions of q and $q_{2,f}$ according to Lemma 4.2.3 on page 18 to get the well-formed machine M.

Since M_1 is in normal form, the final state of M leads back to its initial state no matter whether q is the initial state of M_1, the final state of M_1, or neither. □

Next, we show how to take two machines and form a third by "dovetailing" one onto the end of the other. Notice, that when we dovetail QTMs, the second QTM will be started with the final superposition of the first machine as its input superposition. If the second machine has differing running times for various strings in this superposition, then the dovetailed machine might not halt even though the two original machines were well-behaved. Therefore, a QTM built by dovetailing two well-behaved QTMs may not itself be well-behaved.

Lemma 4.2.6 (Dovetailing Lemma) *If M_1 and M_2 are well-behaved, normal form QTMs (reversible TMs) with the same alphabet, then there is a normal form QTM (reversible TM) M which carries out the computation of M_1 followed by the computation of M_2.*

Proof. Let M_1 and M_2 be well-behaved, normal form QTMs (reversible TMs) with the same alphabet and with start states and final states $q_{1,0}, q_{2,0}, q_{1,f}, q_{2,f}$.

To construct M, we simply insert M_2 for the final state of M_1 using Lemma 4.2.5 on page 18.

To complete the proof we need to show that M carries out the computation of M_1 followed by that of M_2.

To see this, first recall that since M_1 and M_2 are in normal form, the only transitions into $q_{1,0}$ and $q_{2,0}$ are from $q_{1,f}$ and $q_{2,f}$ respectively. This means that no transitions in M_1 have been changed except for those into or out of state $q_{1,f}$. Therefore, since M_1 is well-behaved and does not prematurely enter state $q_{1,f}$, the machine M, when started in state $q_{1,0}$, will compute exactly as M_1 until M_1 would halt. At that point M will instead reach a superposition with all configurations in state $q_{2,0}$. Then, since no transitions in M_2 have been changed except for those into or out of $q_{2,f}$, M will proceed exactly as if M_2 had been started in the superposition of outputs computed by M_1. □

Now, we show how to build a conditional branch around two existing QTMs or reversible TMs. The branching machine will run one of the two machines on its first track input, depending on its second track input. Since a TM can have only one final state, we must rejoin the two branches at the end. We can join reversibly if we write back out the bit that determined which machine was used. The construction will simply build a reversible TM that accomplishes the desired branching and rejoining, and then insert the two machines for states in this branching machine.

Lemma 4.2.7 (Branching Lemma) *If M_1 and M_2 are well-behaved, normal form QTMs (reversible TMs) with the same alphabet, then there is a well-behaved, normal form QTM (reversible TM) M such that if the second track is empty, M runs M_1 on its first track and leaves its second track empty, and if the second track has a 1 in the start cell (and all other cells blank), M runs M_2 on its first track and leaves the 1 where its tape head ends up. In either case, M takes exactly four time steps more than the appropriate M_i.*

Proof. Let M_1 and M_2 be well-behaved, normal form QTMs (reversible TMs) with the same alphabet.

Then we can construct the desired QTM as follows. We will show how to build a stationary, normal form reversible TM BR which always takes four time steps and leaves its input unchanged, always has a superposition consisting of a single configuration, and has two states q_1 and q_2 with the following properties. If BR is run with a 1 in the start cell, and blanks elsewhere, then BR visits q_1 once with a blank tape and with its tape head in the start cell and doesn't visit q_2 at all, and similarly if BR is run with a blank tape then BR visits q_2 once with a blank tape and with its tape head in the start cell and doesn't visit q_1 at all. Then if we extend M_1 and M_2 to have a second track with the alphabet of BR, extend BR to have a first track with the common alphabet of M_1 and M_2, and insert M_1 for state q_1 and M_2 for state q_2 in BR, we will have the desired QTM M.

We complete the construction by exhibiting the reversible TM BR. The machine enters state q_1' or q_2' depending on whether the start cell contains a 1 and steps left, and enters the corresponding q_1 or q_2 while stepping back right. Then it enters state q_3 while stepping left and state q_f while stepping back right.

So, we let BR have alphabet $\{\#, 1\}$, state set $\{q_0, q_1, q_1', q_2, q_2', q_3, q_f\}$ and transition function defined by the following table

	#	1
q_0	$\#, q_2', L$	$\#, q_1', L$
q_1'	$\#, q_1, R$	
q_2'	$\#, q_2, R$	
q_1	$1, q_3, L$	
q_2	$\#, q_3, L$	
q_3	$\#, q_f, R$	
q_f	$\#, q_0, R$	$1, q_0, R$

It can be verified that the transition function of BR is one-to-one and that it can enter each state while moving in only one direction. Therefore, appealing to Theorem B.0.15 on page 56 it can be completed to give a reversible TM. □

Finally, we show how to take a unidirectional QTM or reversible TM and build its reverse. Two computational paths of a QTM will interfere only if they reach configurations that do not differ in any way. This means that we must be careful when building a QTM to erase any information that might differ along paths which we want to interfere. We will therefore sometimes use this lemma when constructing a QTM to allow us to completely erase an intermediate computation.

Since the time evolution of a well-formed QTM is unitary, we could reverse a QTM by applying the unitary inverse of its time evolution. However, this transformation is not the time evolution of a QTM since it acts on a configuration by changing tape symbols to the left and right of the tape head. However, since each state in a unidirectional QTM can be entered while moving in only one direction, we can reverse the head movements one step ahead of the reversal of the tape and state. Then, the reversal will have its tape head in the proper cell each time a symbol must be changed.

Definition 4.2.8 *If QTMs M_1 and M_2 have the same alphabet, then we say that M_2* reverses *the computation of M_1 if identifying the final state of M_1 with the initial state of M_2 and the initial state of M_1 with the final state of M_2 gives the following. For any input x on which M_1 halts, if c_x and ϕ_x are the initial configuration and final superposition of M_1 on input x, then M_2 halts on initial superposition ϕ_x with final superposition consisting entirely of configuration c_x.*

Lemma 4.2.9 (Reversal Lemma) *If M is a normal form, reversible TM or unidirectional QTM then there is a normal form, reversible TM or QTM M' that reverses the computation of M while taking two extra time steps.*

Proof. We will prove the lemma for normal form, unidirectional QTMs, but the same argument proves the lemma for normal form reversible TMs.

Let $M = (\Sigma, Q, \delta)$ be a normal form, unidirectional QTM with initial and final states q_0 and q_f, and for each $q \in Q$ let d_q be the direction which M must move while entering state q. Then, we define a bijection π on the set of configurations of M as follows. For configuration c in state q, let $\pi(c)$ be the configuration derived from c by moving the tape head one square in direction $\bar{d}_q$ (the opposite of direction d_q). Since the set of superpositions of the form $|c\rangle$ give an orthonormal basis for the space of superpositions of M, we can extend π to act as a linear operator on this space of superpositions by defining $\pi|c\rangle = |\pi c\rangle$. It is easy to see that π is a unitary transformation on the space of superpositions on M.

Our new machine M' will have the same alphabet as M, and state set given by Q together with new initial and final states q_0' and q_f'. The following three statements suffice to prove that M' reverses the computation of M while taking two extra time steps.

1. If c is a final configuration of M, and c' is the configuration c with state q_f replaced by state q_0', then a single step of M' takes superposition $|c'\rangle$ to superposition $\pi(|c\rangle)$.
2. If a single step of M takes superposition $|\phi_1\rangle$ to superposition $|\phi_2\rangle$, where $|\phi_2\rangle$ has no support on a configuration in state q_0, then a single step of M' takes superposition $\pi(|\phi_2\rangle)$ to superposition $\pi(|\phi_1\rangle)$.
3. If c is an initial configuration of M, and c' is the configuration c with state q_0 replaced by state q_f', then a single step of M' takes superposition $\pi(|c\rangle)$ to superposition $|c'\rangle$.

To see this, let x be an input on which M halts, and let $|\phi_1\rangle, \ldots, |\phi_n\rangle$ be the sequence of superpositions of M on input x, so that $|\phi_1\rangle = |c_x\rangle$ where c_x is the initial superposition of M on x, and $|\phi_n\rangle$ has support only on final configurations of M. Then, since the time evolution operator of M is linear, Statement 1 tells us that if we form the initial configuration $|\phi_n'\rangle$ of M' by replacing each state q_f in the $|\phi_n\rangle$ with state q_0', then M' takes $|\phi_n'\rangle$ to $\pi(|\phi_n\rangle)$ in a single step. Since M is in normal form, none of $|\phi_2\rangle, \ldots, |\phi_n\rangle$ have support on any superposition in state q_0. Therefore, Statement 2 tells us that the next n steps of M' lead to superpositions $\pi(|\phi_{n-1}\rangle), \ldots, \pi(|\phi_1\rangle)$. Finally, Statement 3 tells us that a single step of M' maps superposition $\pi(|c_x\rangle)$ to superposition $|c_x'\rangle$.

We define the transition function δ' to give a well-formed M' satisfying these three statements with the following transition rules.

1. $\delta'(q_0', \sigma) = |\sigma\rangle|q_f\rangle|\bar{d}_{q_f}\rangle$
2. For each $q \in Q - q_0$ and each $\tau \in \Sigma$
$$\delta'(q, \tau) = \sum_{p,\sigma} \delta(p, \sigma, \tau, q, d_q)^* |\sigma\rangle|p\rangle|\bar{d}_p\rangle$$
3. $\delta'(q_0, \sigma) = |\sigma\rangle|q_f'\rangle|d_{q_0}\rangle$
4. $\delta'(q_f', \sigma) = |\sigma\rangle|q_0'\rangle|R\rangle$

The first and third rules can easily be seen to ensure Statements 1 and 3. The second rule can be seen to ensure Statement 2 as follows: Since M is in normal form, it maps a superposition $|\phi_1\rangle$ to a superposition $|\phi_2\rangle$ with no support on any configuration in state q_0 if and only if $|\phi_1\rangle$ has no support on any configurations in state q_f. Therefore, the time evolution of M defines a unitary transformation from the space $\mathcal{S}_1$ of superpositions of configurations in states from the set $Q_1 = Q - q_f$ to the space of superpositions $\mathcal{S}_2$ of configurations in states from the set $Q_2 = Q - q_0$. This fact also tells us that the second rule above defines a linear transformation from space $\mathcal{S}_2$ back to space $\mathcal{S}_1$. Moreover, if M takes configuration c_1 with a state from Q_1 leads with amplitude α to configuration c_2 with a state from Q_2, then M' takes configuration $\pi(c_2)$ to configuration $\pi(c_1)$ with amplitude α^*. Therefore, the time evolution of M' on space $\mathcal{S}_2$ is the composition of π and its inverse around the conjugate transpose of the time evolution of M. Since this conjugate transpose must also be unitary, the second rule above actually gives a unitary transformation from the space $\mathcal{S}_2$ to the space $\mathcal{S}_1$ which satisfies Statement 2 above.

Since M' is clearly in normal form, we complete the proof by showing that M' is well-formed. To see this, just note that each of the four rules defines a unitary transformation to one of a set of four mutually orthogonal subspaces of the spaces of superpositions of M'. □

The Synchronization Theorem will allow us to take an existing QTM and put it inside a loop so that the machine can be run any specified number of times.

Building a reversible machine that loops indefinitely is trivial. However, if we want to loop some finite number of times, we need to carefully construct a reversible entrance and exit. As with the Branching Lemma above, the construction will proceed by building a reversible TM that accomplishes the desired looping, and then inserting the QTM for a particular state in this looping machine. However, there are several difficulties. First, in this construction, as opposed to the branching construction, the reversible TM leaves an intermediate computation written on its tape while the QTM runs. This means that inserting a non-stationary QTM would destroy the proper functioning of the reversible TM. Second, even if we insert a stationary QTM, the second (and any subsequent) time the QTM is run, it may be started in superposition on inputs of different lengths, and hence may not halt. There is therefore no general statement we are able to make about the behavior of the machine once the insertion is carried out. Instead, we describe here the reversible looping TM constructed in Appendix C on page 61 and argue about specific QTMs resulting from this machine when they are constructed.

Lemma 4.2.10 (Looping Lemma) *There is a stationary, normal form, reversible TM M and a constant c with the following properties. On input any positive integer k written in binary, M runs for time $O(k \log^c k)$ and halts with its tape unchanged. Moreover, M has a special state q^* such that on input k, M visits state q^* exactly k times, each time with its tape head back in the start cell.*

5 Changing the basis of a QTM's computation

In this section we introduce a fundamentally quantum mechanical feature of quantum computation, namely changing the computational basis during the evolution of a QTM. In particular, we will find it useful to change to a new orthonormal basis for the transition function in the middle of the computation – each state in the new state set is a linear combination of states from the original machine.

This will allow us to simulate a general QTM with one that is unidirectional. It will also allow us to prove that any partially defined quantum transition function which preserves length can be extended to give a well-formed QTM.

We start by showing how to change the basis of the states of a QTM. Then we give a set of conditions for a quantum transition function that are necessary and sufficient for a QTM to be well-formed. The last of these conditions, called separability, will allow us to construct a basis of the states of a QTM which will allow us to prove the Unidirection and Completion Lemmas below.

5.1 The change of basis

If we take a well-formed QTM and choose a new orthonormal basis for the space of superpositions of its states, then translating its transition function into this new basis will give another well-formed QTM that evolves just as the first under the change of basis. Note, that in this construction, the new QTM has the same time evolution operator as the original machine. However, the states of the new machine differ from those of the old. This change of basis will allow us to prove Lemmas 5.3.2 and 5.3.4 below.

Lemma 5.1.1 *Given a QTM $M = (\Sigma, Q, \delta)$ and a set of vectors B from $\tilde{C}^Q$ which forms an orthonormal basis for $\mathbf{C}^Q$, there is a QTM $M' = (\Sigma, B, \delta')$ which evolves exactly as M under a change of basis from Q to B.*

Proof. Let $M = (\Sigma, Q, \delta)$ be a QTM and B an orthonormal basis for $\mathbf{C}^Q$.

Since B is an orthonormal basis, this establishes a unitary transformation from the space of superpositions of states in Q to the space of superpositions of states in B. Specifically, for each $p \in Q$ we have the mapping

$$|p\rangle \rightarrow \sum_{v \in B} \langle p||v\rangle \; |v\rangle$$

Similarly, we have a unitary transformation from the space of superpositions of configurations with states in Q to the space of configurations with states in B. In this second transformation, a configuration with state p is mapped to the superposition of configurations where the corresponding configuration with state v appears with amplitude $\langle p||v\rangle$.

Let us see what the time evolution of M should look like under this change of basis. In M a configuration in state p reading a σ evolves in a single time step to the superposition of configurations corresponding to the superposition $\delta(p, \sigma)$.

$$\delta(p, \sigma) = \sum_{\tau, q, d} \delta(p, \sigma, \tau, q, d) \; |\tau\rangle|q\rangle|d\rangle$$

With the change of basis, the superposition on the right hand side will instead be

$$\sum_{\tau, v, d} \left(\sum_q \langle q|v\rangle \; \delta(p, \sigma, \tau, q, d) \right) |\tau\rangle|v\rangle|d\rangle$$

Now, since the state symbol pair $|v\rangle|\sigma\rangle$ in M' corresponds to the superposition

$$\sum_p \langle v|p\rangle \; |p\rangle|\sigma\rangle$$

in M, we should have in M'

$$\delta'(v, \sigma) = \sum_p \langle v|p\rangle \left(\sum_{\tau, v', d} \left(\sum_q \langle q|v'\rangle \; \delta(p, \sigma, \tau, q, d) \right) \right) |\tau\rangle|v'\rangle|d\rangle$$

Therefore, M' will behave exactly as M under the change of basis if we define δ' by saying that for each $v, \sigma \in B \times \Sigma$

$$\delta'(v, \sigma) = \sum_{\tau, v', d} \left(\sum_{p, q} \langle v|p\rangle \; \langle q|v'\rangle \; \delta(p, \sigma, \tau, q, d) \right) |\tau\rangle|v'\rangle|d\rangle$$

Since the vectors in B are contained in $\tilde{C}^Q$, each amplitude of δ' is contained in $\tilde{C}$.

Finally, note that the time evolution of M' must preserve Euclidean length since it is exactly the time evolution of the well-formed M under the change of basis. □

5.2 Local conditions for QTM well-formedness

In our discussion of reversible TMs in Appendix B we find properties of a deterministic transition function which are necessary and sufficient for it to be reversible. Similarly, our next theorem gives three properties of a quantum transition function which together are necessary and sufficient for it to be well-formed. The first property insures that the time evolution operator preserves length when applied to any particular configuration. Adding the second insures that the time evolution preserves the length of superpositions of configurations with their tape head in the same cell. The third property, which concerns "restricted" update superpositions, handles superpositions of configurations with tape heads in differing locations. This third property will be of critical use in the constructions of Lemmas 5.3.2 and 5.3.4 below.

Definition 5.2.1 *We will also refer to the superposition of states*

$$\sum_{q \in Q} \delta(p, \sigma, \tau, q, d)|q\rangle$$

resulting from restricting $\delta(p,\sigma)$ to those pieces writing symbol τ and moving in direction d as a direction d-going restricted superposition, *denoted by $\delta(p, \sigma|\tau, d)$.*

Theorem 5.2.2 *A QTM $M = (\Sigma, Q, \delta)$ is well-formed iff the following conditions hold*

Unit length

$$\forall \; p, \sigma \;\in\; Q \times \Sigma \quad \|\delta(p,\sigma)\| = 1$$

Orthogonality

$$\forall \; (p_1, \sigma_1) \neq (p_2, \sigma_2) \;\in\; Q \times \Sigma \quad \delta(p_1, \sigma_1) \cdot \delta(p_2, \sigma_2) = 0$$

Separability

$$\forall \; (p_1, \sigma_1, \tau_1), (p_2, \sigma_2, \tau_2) \;\in\; Q \times \Sigma \times \Sigma \quad \delta(p_1, \sigma_1|\tau_1, L) \cdot \delta(p_2, \sigma_2|\tau_2, R) = 0$$

Proof. Let U be the time evolution of a proposed QTM $M = (\Sigma, Q, \delta)$. We know M is well-formed iff U^* exists and U^*U gives the identity, or equivalently iff the columns of U have unit length and are mutually orthogonal. Clearly, the first condition specifies exactly that each column has unit length. In general, configurations whose tapes differ in a cell not under either of their heads, or whose tape heads are not either in the same cell or exactly two cells apart, cannot yield the same configuration in a single step. Therefore such pairs of columns are guaranteed to be orthogonal, and we need only consider pairs of configurations for which this is not the case. The second condition specifies the orthogonality of pairs of columns for configurations that differ only in that one is in state p_1 reading σ_1 while the other is in state p_2 reading σ_2.

Finally, we must consider pairs of configurations with their tape heads two cells apart. Such pairs can only interfere in a single step if they differ at most in their states and in the symbol written in these cells. The third condition specifies the orthogonality of pairs of columns for configurations which are identical except that the second has its tape head two cells to the left, is in state p_2 instead of p_1, has a σ_2 instead of a τ_2 in the cell under its tape head, and has a τ_1 instead of a σ_1 two cells to the left. $\square$

Now consider again unidirectional QTMs, those in which each state can be entered while moving in only one direction. When we considered this property for deterministic TMs, it meant that when looking at a deterministic transition function δ, we could ignore the direction update and think of δ as giving a bijection from the current state and tape symbol to the new symbol and state. Here, if δ is a unidirectional quantum transition function, then it certainly satisfies the separability condition since no left-going and right-going restricted superpositions have a state in common. Moreover, update triples will always share the same direction if they share the same state. Therefore, a unidirectional δ is well-formed iff, ignoring the direction update, δ gives a unitary transformation from superpositions of current state and tape symbol to superpositions of new symbol and state.

5.3 Unidirection and Completion Lemmas

The separability condition of Theorem 5.2.2 allows us to simulate any QTM with a unidirectional QTM by applying a change of basis. The same change of basis will also allow us to complete any partially specified quantum transition function which preserves length.

It is straightforward to simulate a deterministic TM with one which is unidirectional. Simply split each state q into two states q_r and q_l, both of which are given the same transitions that q had, and then edit the transition function so that transitions moving right into q enter q_r and transitions moving left into q enter q_l. The resulting machine is clearly not reversible since the transition function operates the same on each pair of states q_r, q_l.

To simplify the unidirection construction, we first show how to interleave a series of quantum transition functions.

Lemma 5.3.1 *Given k state sets $Q_0, \ldots, Q_{k-1}, Q_k = Q_0$ and k transition functions each mapping from one state set to the next*

$$\delta_i : Q_i \times \Sigma \to \tilde{\mathbf{C}}^{\Sigma \times Q_{i+1} \times \{L,R\}}$$

such that each δ_i preserves length, there is a well-formed QTM M with state set $\bigcup_i (Q_i, i)$ which alternates stepping according to each of the k transition functions.

Proof. Suppose we have k state sets transition functions as stated above. Then we let M be the QTM with the same alphabet, with state set given by the union of the individual state sets $\bigcup_i (Q_i, i)$, and with transition function according to the δ_i

$$\delta((p, i), \sigma) = \sum_{\tau, q, d} \delta_i(p, \sigma, \tau, q, d) |\tau\rangle |q, i+1\rangle |d\rangle$$

Clearly, the machine M alternates stepping according to $\delta_0, \ldots, \delta_{k-1}$. It is also easy to see that the time evolution of M preserves length; If $|\phi\rangle$ is a superposition of configurations with squared length α_i in the subspace with configurations with states from Q_i, then δ maps ϕ to a superposition with squared length α_i in the subspace with configurations with states from Q_{i+1}. $\square$

Lemma 5.3.2 (Unidirection Lemma) *Any QTM M is simulated, with slowdown by a factor of 5, by a unidirectional QTM M'. Furthermore, if M is well-behaved and in normal form, then so is M'.*

Proof. The key idea is that the separability condition of a well-formed QTM allows a change of basis to a state set in which each state can be entered from only one direction.

The separability condition says that

$$\forall \ (p_1, \sigma_1, \tau_1), (p_2, \sigma_2, \tau_2) \ \in \ Q \times \Sigma \times \Sigma$$

$$\delta(p_1, \sigma_1 | \tau_1, L) \cdot \delta(p_2, \sigma_2 | \tau_2, R) = 0$$

This means that we can split $\mathbf{C}^Q$ into mutually orthogonal subspaces $\mathbf{C}_L$ and $\mathbf{C}_R$ such that $span(\mathbf{C}_L, \mathbf{C}_R) = \mathbf{C}^Q$ and

$$\forall \ (p_1, \sigma_1, \tau_1) \in \ Q \times \Sigma \times \Sigma$$

$$\delta(p_1, \sigma_1 | \tau_1, d) \in \mathbf{C}_d$$

Now, as shown in Lemma 5.1.1 above, under a change of basis from state set Q to state set B the new transition function is defined by

$$\delta'(v, \sigma, \tau, v', d) = \sum_{p,q} \langle v|p\rangle\langle q|v'\rangle\delta(p, \sigma, \tau, q, d)$$

So, choose orthonormal bases B_L and B_R for the spaces $\mathbf{C}_L$ and $\mathbf{C}_R$ and let $M' = (\Sigma, B_L \cup B_R, \delta')$ be the QTM constructed according to Lemma 5.1.1 which evolves exactly as M under a change of basis from state set Q to state set $B = B_L \cup B_R$. Then any state in M' can be entered in only one direction. To see this, first note that since $\delta(p, \sigma|\tau, d) \in B_d$ and $v = \sum_q \langle v|q\rangle|q\rangle$, the separability condition implies that for $v \in B_{\overline{d}}$

$$\sum_q \delta(p, \sigma, \tau, q, d)\langle v|q\rangle^* = 0$$

Therefore, for any $v, \sigma, \tau, v' \in B \times \Sigma \times \Sigma \times B_{\overline{d}}$

$$\delta'(v, \sigma, \tau, v', d) = \sum_{p,q} \langle v|p\rangle\langle q|v'\rangle\delta(p, \sigma, \tau, q, d)$$

$$= \sum_p \langle v|p\rangle \sum_q \langle q|v'\rangle\delta(p, \sigma, \tau, q, d) = 0$$

Therefore any state in B can be entered while traveling in only one direction.

Unfortunately, this new QTM M' might not be able to simulate M. The problem is that the start state and final state of M might correspond under the change of basis isomorphism to superpositions of states in M', meaning that we would be unable to define the necessary start and final states for M'. To fix this problem, we use five time steps to simulate each step of M and interleave the five transition functions using Lemma 5.3.1 on page pagerefinterleaving.lemma.

1. Step right leaving the tape and state unchanged.

$$\delta_0(p, \sigma) = |\sigma\rangle|p\rangle|R\rangle$$

2. Change basis from Q to B while stepping left.

$$\delta_1(p, \sigma) = \sum_{b \in B} \langle p|b\rangle|\sigma\rangle|b\rangle|L\rangle$$

3. M' carries out a step of the computation of M. So, δ_2 is just the quantum transition function δ' from QTM M' constructed above.

4. Change basis back from B to Q while stepping left.

$$\delta_3(b, \sigma) = \sum_{p \in Q} \langle b|p\rangle|\sigma\rangle|p\rangle|L\rangle$$

5. Step right leaving the tape and state unchanged.

$$\delta_4(p, \sigma) = |\sigma\rangle|p\rangle|R\rangle$$

If we construct QTM M' with state set $(Q \times \{0,1,4\} \cup (B \times \{2,3\})$ using Lemma 5.3.1 on page 25 and let M' have start and final states $(q_0, 0)$ and $(q_f, 0)$ then M' simulates M with slowdown by a factor of 5.

Next, we must show that each of the five transition functions obeys the well-formedness conditions and hence according to Lemma 5.3.1 that the interleaved machine is well-formed.

The transition function $\delta_2 = \delta'$ certainly obeys the well-formedness conditions since M' is a well-formed QTM. Also, δ_0 and δ_4 obey the three well-formedness conditions since they are deterministic and reversible. Finally, the transition functions δ_1 and δ_3 satisfy the unit length and orthogonality conditions since they implement a change of basis, and they obey the separability condition since they only move in one direction.

Finally, we must show that if M is well-behaved and in normal form then we can make M' well-behaved and in normal form.

So, suppose M is well-behaved and in normal form. Then there is a T such that at time T the superposition includes only configurations in state q_f with the tape head back in the start cell, and at any time less than T the superposition contains no configuration in state q_f. But this means that when M' is run on input x, the superposition at time $5T$ includes only configurations in state $(q_f, 0)$ with the tape head back in the start cell, and the superposition at any time less than $5T$ contains no configuration in state $(q_f, 0)$. Therefore M' is also well-behaved.

Then for any input x, there is a T such that M enters a series of $T - 1$ superpositions of configurations all with states in $Q - \{q_0, q_f\}$ and then enters a superposition of configurations all in state q_f with the tape head back in the start cell. Therefore, on input x M' enters a series of $5T - 1$ superpositions of configurations all with states in $Q' - \{(q_f, 0), (q_0, 4)\}$ and then enters a superposition of configurations all in state $(q_f, 0)$ with the tape head back in the start cell. Therefore, M' is well-behaved. Also, swapping the outgoing transitions of $(q_f, 0)$ and $(q_0, 4)$, which puts M' in normal form, will not change the computation of M' on any input x. □

When we construct a QTM, we will often only be concerned with a subset of its transitions. Luckily, any partially defined transition function that preserves length can be extended to give a well-formed QTM.

Definition 5.3.3 *A QTM M whose quantum transition function δ' is only defined for a subset $S \subseteq Q \times \Sigma$ is called a* partial *QTM. If the defined entries of δ' satisfy the three conditions of Theorem 5.2.2 on page 24 then M is called a* well-formed partial *QTM.*

Lemma 5.3.4 (Completion Lemma) *Suppose M is a well-formed partial QTM with quantum transition function δ. Then there is a well-formed QTM M' with the same state set and alphabet whose quantum transition function δ' agrees with δ wherever the latter is defined.*

Proof. We noted above that a unidirectional quantum transition function is well-formed iff, ignoring the direction update, it gives a unitary transformation from superpositions of current state and tape symbol to superpositions of new symbol and state. So if our partial QTM is unidirectional, then we can easily fill in the undefined entries of δ by extending the set of update superpositions of δ to an orthonormal basis for the space of superpositions of new symbol and state.

For a general δ, we can use the technique of Lemma 5.1.1 on page 22 to change the basis of δ away from Q so that each state can be entered while moving in only one direction, extend the transition function, and then rotate back to the basis Q.

We can formalize this as follows:

Let $M = (\Sigma, Q, \delta)$ be a well-formed partial QTM with δ defined on the subset $S \subseteq Q \times \Sigma$. Denote by $\overline{S}$ the complement of S in $Q \times \Sigma$.

As in the proof of the Unidirection Lemma above, the separability condition allows us to partition $\mathbf{C}^Q$ into mutually orthogonal subspaces $\mathbf{C}_L$ and $\mathbf{C}_R$ such that $span(\mathbf{C}_L, \mathbf{C}_R) = \mathbf{C}^Q$ and

$$\forall\ (p_1, \sigma_1, \tau_1) \in\ S \times \Sigma$$

$$\delta(p_1, \sigma_1 | \tau_1, d) \in \mathbf{C}_d$$

Then, we choose orthonormal bases B_L and B_R for $\mathbf{C}_L$ and $\mathbf{C}_R$, and consider the unitary transformation from superpositions of configurations with states in Q to the space of configurations with states in $B = B_L \cup B_R$, where any configuration with state p is mapped to the superposition of configurations where the corresponding configuration with state v appears with amplitude $\langle p|v\rangle$.

If we call δ' the partial function δ followed by this unitary change of basis, then we have

$$\delta'(p, \sigma) = \sum_{\tau, v, d} \left(\sum_q \langle q|v\rangle \delta(p, \sigma, \tau, q, d) \right) |\tau\rangle |v\rangle |d\rangle$$

Since δ preserves length and δ' is δ followed by a unitary transformation, δ' also preserves length.

But now δ' can enter any state in B while moving in only one direction. To see this, first note that since $\delta(p, \sigma; \tau, d) \in B_d$ and $v = \sum_q \langle v|q\rangle |q\rangle$, the separability condition implies that for $v \in B_{\overline{d}}$

$$\sum_q \delta(p, \sigma, \tau, q, d) \langle v|q\rangle^* = 0$$

Therefore, for any $p, \sigma, \tau, v \in S \times \Sigma \times B_{\overline{d}}$

$$\delta'(p, \sigma, \tau, v, d) = \sum_q \langle q|v\rangle \delta(p, \sigma, \tau, q,) = 0$$

Therefore any state in B can be entered while traveling in only one direction.

Then, since the direction is implied by the new state, we can think of δ' as mapping the current state and symbol to a superposition of new symbol and state. Since δ' preserves length, the set $\delta'(S)$ is a set of orthonormal vectors, and we can expand this set to an orthonormal basis of the space of superpositions of new symbol and state. Adding the appropriate direction updates and assigning these new vectors arbitrarily to $\delta'(\bar{S})$, we have a completely defined δ' that preserves length. Therefore, assigning $\delta(\bar{S})$ as $\delta'(\bar{S})$ followed by the inverse of the basis transformation gives a completion for δ that preserves length. □

6 An efficient QTM implementing any given unitary transformation

Suppose that the tape head of a QTM is confined to a region consisting of k contiguous tape cells (the tape is blank outside this region). Then the time evolution of the QTM can be described by a d dimensional unitary transformation, where $d = k\text{card}(Q)\text{card}(\Sigma)^k$. In this section we show conversely that there is a QTM that given any d dimensional unitary transformation as input, carries out that transformation (on a region of its tape). To make this claim precise we must say how the d dimensional unitary transformation is specified. We assume that an approximation to the unitary transformation is specified by a $d \times d$ complex matrix whose entries are approximations to the entries of the actual unitary matrix corresponding to the desired transformation. We show in Theorem 6.4.1 on page 37 that there is a QTM which on input ϵ and a d dimensional transformation which is within distance $\frac{\epsilon}{2(10\sqrt{d})^d}$ of a unitary transformation carries out a transformation which is an ϵ approximation to the desired unitary transformation. Moreover, the running time of the QTM is bounded by a polynomial in d and $1/\epsilon$.

In a single step, a QTM can map a single configuration into a superposition of a bounded number of configurations. Therefore, in order to carry out an (approximation to an) arbitrary unitary transformation on a QTM, we show how to approximate it by a product of simple unitary transformations - each such simple transformation acts as the identity in all but two dimensions. We then show that there is a particular simple unitary transformation, such that any given simple transformation can be expressed as a product of permutation matrices and powers of this fixed simple matrix. Finally, we put it all together, and show how to design a single QTM that carries out an arbitrary unitary transformation - this QTM is deterministic except for a single kind of quantum coin-flip.

The decomposition of an arbitrary unitary transformation into a product of simple unitary transformations is similar to work carried out by Deutsch [21]. Deutsch's work, although phrased in terms of quantum computation networks, can be viewed as showing that a d dimensional unitary transformation can be decomposed into a product of transformations where each applies a particular unitary transformation to 3 dimensions and acts as the identity elsewhere. We must consider here several issues of efficiency not addressed by Deutsch. First, we are concerned that the decomposition contain a number of transformations which is polynomial in the dimension of the unitary transformation and in the desired accuracy. Second, we desire that the decomposition can itself be efficiently computed given the desired unitary transformation as input. For more recent work on the efficient simulation of a unitary transformation by a quantum computation network see [5] and the references therein.

6.1 Measuring errors in approximated transformations

In this section, we will deal with operators (linear transformations) on finite dimensional Hilbert spaces. It is often convenient to fix an orthonormal basis for the Hilbert space and describe the operator by a finite matrix with respect to the chosen basis. Let $e_1, \ldots, e_d$ be an orthonormal basis for Hilbert space $\mathcal{H} = \mathbf{C}^d$. Then we can represent an operator U on $\mathcal{H}$ by a $d \times d$ complex matrix M, whose i, j^{th} entry $m_{i,j}$ is (Ue_j, e_i). The i^{th} row of the matrix M is given by ${e_i}^T M$, and we will denote it by M_i. We denote by ${M_i}^*$ the conjugate transpose of M_i. The j^{th} column of M is given by Me_j. U^*, the adjoint of U, is represented by the $d \times d$ matrix M^*. M is unitary iff $MM^* = M^*M = I$. It follows that if M is unitary then the rows (and columns) of M are orthonormal.

Recall that for a bounded linear operator U on a Hilbert space $\mathcal{H}$, the norm of U is defined as

$$|U| \; = \; \sup_{||x||=1} |Ux|$$

If we represent U by the matrix M, then we can define the norm of the matrix M to be same as the norm of U. Thus, since we're working in a finite dimensional space,

$$||M|| \; = \; \max_{||v||=1} ||Mv||$$

.

Fact 6.1.1 *If U is unitary then $||U|| = ||U^*|| = 1$.*

Proof. $\forall x \in \mathcal{H}$, $\|Ux\|^2 = (Ux, Ux) = (x, U^*Ux) = (x, x) = \|x\|^2$. Therefore $\|U\| = 1$, and similar reasoning shows $\|U^*\| = 1$. $\square$

We will find it useful to keep track of how far our approximations are from being unitary. We will use the following simple measure of a transformation's distance from being unitary.

Definition 6.1.2 *A bounded linear operator U is called* ϵ-close to unitary *if there is a unitary operator $\tilde{U}$ such that $\|U - \tilde{U}\| \leq \epsilon$. If we represent U by the matrix M, then we can equivalently say that M is* ϵ-close to unitary *if there is a unitary matrix $\tilde{M}$ such that $\|M - \tilde{M}\| \leq \epsilon$.*

Notice that, appealing to Statement 3 in §2 if U is ϵ-close to unitary then $1-\epsilon \leq \|U\| \leq 1+\epsilon$. However, the converse is not true. For example the linear transformation $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ has norm 1 but is 1 away from unitary.

Next, we show that if M is close to unitary, then the rows of M are close to unit length and will be close to orthogonal.

Lemma 6.1.3 *If a d dimensional complex matrix M is ϵ-close to unitary, then:*

$$1 - \epsilon \;\leq\; \|M_i\| \;\leq\; 1 + \epsilon \tag{4}$$

$$\forall i \neq j \;\; \|M_i {M_j}^*\| \;\leq\; 2\epsilon + 3\epsilon^2 \tag{5}$$

Proof. Let $\tilde{M}$ be the unitary matrix such that $\|M - \tilde{M}\| \leq \epsilon$. Let $N = M - \tilde{M}$. Then we know that for each i, $\|\tilde{M}_i\| = 1$ and $\|N_i\| \leq \epsilon$.

So, for any i, we have $M_i = N_i + \tilde{M}_i$. Therefore $1 - \epsilon \leq \|M_i\| \leq 1 + \epsilon$.

Next, since $\tilde{M}$ is unitary, it follows that for $i \neq j$, $\tilde{M}_i \tilde{M}_j^* = 0$. Therefore, $(M_i - N_i)(M_j^* - N_j^*) = 0$. Expanding this as a sum of four terms, we get

$$M_i M_j^* = M_i N_j^* + N_i M_j^* - N_i N_j^*$$

Since $\|M\| \leq 1+\epsilon$ and $\|N\| \leq \epsilon$, the Schwarz inequality tells us that $\|M_i N_j^*\| \leq (1+\epsilon)\epsilon$, $\|N_i M_j^*\| \leq \epsilon(1+\epsilon)$, and $\|N_i N_j^*\| \leq \epsilon^2$. Using the triangle inequality we conclude that $\|M_i M_j^*\| \leq 2(1+\epsilon)\epsilon + \epsilon^2$. Therefore $\|M_i {M_j}^*\| \leq 2\epsilon + 3\epsilon^2$. $\square$

We will also use the following standard fact that a matrix with small entries must have small norm.

Lemma 6.1.4 *If M is a d-dimensional square complex matrix such that $|m_{i,j}| \leq \epsilon$ for all i, j, then $\|M\| \leq d\epsilon$.*

Proof. If each entry of M has magnitude at most ϵ, then clearly each row M_i of M must have norm at most $\sqrt{d}\epsilon$. So, if v is a d-dimensional column vector with $|v| = 1$, we must have

$$\|Mv\|^2 = \sum_i \|M_i v\|^2 \leq \sum_i d\epsilon^2 = d^2\epsilon^2$$

where the inequality follows from the Scwarz Inequality. Therefore $\|M\| \leq d\epsilon$. $\square$

6.2 Decomposing a unitary transformation

We now describe a class of exceedingly simple unitary transformations which we will be able to carry out using a single QTM. These "near-trivial" transformations either apply a phase shift in one dimension or apply a rotation between two dimensions, while acting as the identity otherwise.

Definition 6.2.1 *A $d \times d$ unitary matrix M is* near-trivial *if it satisfies one of the following two conditions*

1. *M is the identity except that one of its diagonal entries is $e^{i\theta}$ for some $\theta \in [0, 2\pi]$. i.e. $\exists j\ m_{j,j} = e^{i\theta}, \forall k \neq j\ m_{k,k} = 1$, and $\forall k \neq l\ m_{k,l} = 0$.*
2. *M is the identity except that the submatrix in one pair of distinct dimensions j and k is the rotation by some angle $\theta \in [0, 2\pi]$: $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. So, as a transformation M is near-trivial if there exists θ and $i \neq j$ such that $Me_i = (\cos\theta)e_i + (\sin\theta)e_j$, $Me_j = -(\sin\theta)e_i + (\cos\theta)e_j$, and $\forall k \neq i, j\ Me_k = e_k$,*

We call a transformation which satisfies the former a near-trivial phase shift, *and we call a transformation which satisfies the latter a* near-trivial rotation.

We will write a near-trivial matrix M in the following way. If M is a phase shift of $e^{i\theta}$ in dimension j then we will write down $[j, j, \theta]$ and if M is a rotation of angle θ between dimensions j and k we will write down $[j, k, \theta]$. This convention guarantees that the matrix that we are specifying is a near-trivial matrix and therefore a unitary matrix, *even if* for precision reasons we write down an approximation to the matrix that we really wish to specify. This feature will substantially simplify our error analyses.

Before we show how to use near-trivial transformations to carry out an arbitrary unitary transformation, we first show how to use them to map any particular vector to a desired target direction.

Lemma 6.2.2 *There is a deterministic algorithm which on input a vector $v \in \mathbf{C}^d$ and a bound $\epsilon > 0$, computes near trivial matrices $U_1, \ldots, U_{2d-1}$ such that*

$$\|U_1 \cdots U_{2d-1} v \ - \ \|v\| e_1\| \leq \epsilon$$

where e_1 is the unit vector in the first coordinate direction. The running time of the algorithm is bounded by a polynomial in d, $\log 1/\epsilon$ and the length of the input.

Proof. First, we use d phase shifts to map v into the space $\mathbb{R}^d$. We therefore want to apply to each dimension i with $v_i \neq 0$ the phase shift $\frac{v_i^*}{\|v_i\|}$. So, we let P_i be the near-trivial matrix which applies to dimension i the phase shift by angle ϕ_i where $\phi_i = 0$ if $v_i = 0$, and otherwise $\phi_i = 2\pi - \cos^{-1}\frac{Re(v_i)}{\|v_i\|}$ or $\cos^{-1}\frac{Re(v_i)}{\|v_i\|}$ depending whether $Im(v_i)$ is positive or negative. Then $P_1 \cdots P_d v$ is the vector with i^{th} coordinate $\|v_i\|$.

Next, we use $d-1$ rotations to move all of the weight of the vector into dimension 1. So, we let R_i be the near-trivial matrix which applies to dimensions i and $i+1$ the rotation by angle θ_i where

$$\theta_i = cos^{-1}\frac{\|v_i\|}{\sqrt{\sum_{j=i}^{d}\|v_j\|^2}}$$

if the sum in the denominator is not 0 and $\theta_i = 0$ otherwise. Then

$$R_1 \cdots R_{d-1} P_1 \cdots P_d v = \|v\| e_1$$

Now, instead of producing these values ϕ_i, θ_i exactly, we can in time polynomial in d and $\log 1/\epsilon$, and the length of the input compute values ϕ_i' and θ_i' which are within $\delta = \frac{\epsilon}{(2d-1)\|v\|}$ of the desired values. Call P_i' and R_i' the near-trivial matrices corresponding to P_i and R_i but using these approximations. Then, since the distance between points at angle θ and θ' on the unit circle in the real plane is at most $|\theta - \theta'|$,

$$\|R_i - R_i'\| \leq \delta.$$

Thinking of the same inequality on the unit circle in the complex plane, we have

$$\|P_i - P_i'\| \leq \delta.$$

Finally, since each matrix P_i, P_i', R_i, R_i' is unitary, Fact 2.0.1 on page 6 gives us

$$\|R_1' \cdots R_{d-1}' P_1' \cdots P_d' \ - \ R_1 \cdots R_{d-1} P_1 \cdots P_d\| \leq (2d-1)\delta$$

and therefore

$$\|R_1' \cdots R_{d-1}' P_1' \cdots P_d' v \ - \ \|v\| e_1\| \leq (2d-1)\delta \|v\| = \epsilon$$

□

We now show how the ability to map a particular vector to a desired target direction allows us to approximate an arbitrary unitary transformation.

Theorem 6.2.3 *There is a deterministic algorithm running in time polynomial in d and $\log 1/\epsilon$ and the length of the input which when given as input U, ϵ where $\epsilon > 0$ and U is a $d \times d$ complex matrix which is $\frac{\epsilon}{2(10\sqrt{d})^d}$-close to unitary, computes d-dimensional near-trivial matrices $U_1, \ldots, U_n$, with n polynomial in d such that $\|U \ - \ U_n \cdots U_1\| \leq \epsilon$.*

Proof. First we introduce notation to simplify the proof. Let U be a $d \times d$ complex matrix. Then we say U is *k-simple* if its first k rows and columns are the same as those of the d-dimensional identity. Notice that the product of two $d \times d$ k-simple matrices is also k-simple.

If U were d-simple, we would have $U = I$ and the desired computation would be trivial. In general, the U which our algorithm must approximate will not even be 1-simple. So, our algorithm will proceed through d phases, such that during the i^{th} phase the remaining problem is reduced to approximating a matrix which is $i+1$ simple.

Suppose we start to approximate a k-simple U with a series of near-trivial matrices with the product V. Then to approximate U, we would still need to produce a series of near-trivial matrices whose product W satisfies $W \approx UV^*$. To reduce the problem we must therefore compute near-trivial matrices whose product V is such that UV^* is close to being $k+1$-simple. We can accomplish this by using the algorithm of Lemma 6.2.2 above.

So, let U be given which is k-simple and is δ-close to unitary, and let Z be the lower right $d-k \times d-k$ submatrix of Z. We invoke the procedure of Lemma 6.2.2 on inputs Z_1^T (the vector corresponding to the first row of Z) and δ. The output is a sequence of $d-k$-dimensional near trivial matrices $V_1, \ldots, V_{2d-2k-1}$ such that their product $V = V_1 \times \ldots \times V_{2d-2k-1}$ has the property that $\|VZ_1^T - \|Z_1\| e_1\| \leq \delta$.

Now suppose that we extend V and the V_i back to d dimensional, k-simple matrices, and we let $W = UV^*$. Then clearly V is unitary and V and W are k-simple. In fact, since V is unitary and U is δ-close to unitary, W is also δ-close to unitary. Moreover, W is close to being $k+1$-simple as desired. We will show below that the $k+1^{st}$ row of W satisfies $\|W_{k+1} - e_{k+1}^T\| \leq 2\delta$, and that the entries of the $k+1^{st}$ column of W satisfy $\|w_{j,k+1}\| \leq 6\delta$ for $j \neq k+1$.

So, let X be the $d \times d$, $k+1$-simple matrix such that

$$x_{1,1} = 1;\ x_{j,1} = 0 \text{ for } j \neq 1;\ x_{1,j} = 0 \text{ for } j \neq 1;\ x_{j,l} = w_{j,l} \text{ for } j, l > l+1$$

It follows from our bounds on the norm of the first row of W and on the entries of the first column of W that $\|W - X\| \leq 2\delta + 6\sqrt{d}\delta$. Since W is δ-close to unitary, we can then conclude that X is $3\delta + 6\sqrt{d}\delta$-close to unitary.

Unfortunately, we cannot compute the entries of $W = UV^*$ exactly. Instead, appealing to Lemma 6.1.4 on page 30, we compute them to within δ/d to obtain a matrix $\hat{W}$ such that $\|\hat{W} - W\| \leq \delta$. Let's use the entries of $\hat{W}$ to define matrix $\hat{X}$ analogous to X. Using the triangle inequality, it is easy to see that $\|W - \hat{X}\| \leq 3\delta + 6\sqrt{d}\delta$ and $\hat{X}$ is $4\delta + 6\sqrt{d}\delta$-close to unitary. If we are willing to incur an error of $\|W - \hat{X}\| \leq 3\delta + 6\sqrt{d}\delta$, then we are left with the problem of approximating the $k+1$-simple $\hat{X}$ by a product of near-trivial matrices. Therefore, we have reduced the problem of approximating the k-simple matrix U by near-trivial matrices to the problem of approximating the $k+1$-simple matrix $\hat{X}$ by near-trivial matrices while incurring two sources of error:

1. An error of $\|W - \hat{X}\| \leq 3\delta + 6\sqrt{d}\delta$, since we are approximating $\hat{X}$ instead of W.
2. The new matrix $\hat{X}$ is only $4\delta + 6\sqrt{d}\delta$-close to unitary.

Let $\delta' = 10\sqrt{d}\delta$. Clearly δ' is an upper-bound on both the sources of error cited above. Therefore, the total error in the approximation is just $\sum_{j=1}^{d}(10\sqrt{d})^j\delta \leq 2(10\sqrt{d})^d\delta$. The last inequality follows since $10\sqrt{d} \geq 2$ and therefore the sum can be bounded by a geometric series. Therefore, the total error in the approximation is bounded by ϵ, since by assumption U is δ-close to unitary for $\delta = \frac{\epsilon}{2(10\sqrt{d})^d}$.

It is easy to see that this algorithm runs in time polynomial in d and $\log 1/\epsilon$. Our algorithm consists of d iterations of first calling the algorithm from Lemma 6.2.2 on page 31 to compute V and then computing the matrix $\hat{X}$. Since the each iteration takes time polynomial in d and $\log \frac{(10\sqrt{d})^d}{\epsilon}$, these d calls take a total time polynomial in d and $\log 1/\epsilon$.

Finally, we show as required that the $k+1^{st}$ row of W satisfies $\|W_{k+1} - e_{k+1}^T\| \leq 2\delta$, and that the entries of the $k+1^{st}$ column of W satisfy $\|w_{j,k+1}\| \leq 6\delta$ for $j \neq k+1$. To see this, first recall that the lower dimension V satisfies $\|VZ_1^T - \|Z_1\|e_1\| \leq \delta$ where Z_1 is the first row of the lower right $k \times k$ submatrix of U. Therefore, the higher dimension V satisfies $\|VU_{k+1}^T - \|U_{k+1}\|e_{k+1}\| \leq \delta$. Then, since $1 - \delta \leq \|U_{k+1}\| \leq 1 + \delta$, it follows that $\|VU_{k+1}^T - e_{k+1}\| \leq 2\delta$. Therefore, the $k+1^{st}$ row of W satisfies $\|W_{k+1} - e_{k+1}^T\| \leq 2\delta$.

Next, we will show that this implies that the entries of the $k+1^{st}$ column of W satisfy $\|w_{j,k+1}\| \leq 6\delta$ for $j \neq k+1$. To see this, first notice that since V is unitary and U is delta close to unitary, W is also δ-close to unitary. This means that, by Statement 5 of Lemma 6.1.3 on page 30 $\left|W_{k+1}W_j^*\right| \leq 2\delta + 3\delta^2$. Now let us use the condition $\|W_{k+1} - e_{k+1}^T\| \leq 2\delta$. This implies that $|w_{k+1,k+1}| \geq 1 - 2\delta$. Also, let us denote by $\hat{W}_j$ the $d-1$ dimensional row vector arrived at by dropping $w_{j,k+1}$ from W_j. Then the condition that W_{k+1} is close to e_{k+1}^T also implies that $\|\hat{W_{k+1}}\| \leq 2\delta$. Also, the fact that W is δ-close to unitary implies that $\|\hat{W}_j\| \leq 1 + \delta$. Putting all this together, we have $2\delta + 3\delta^2 \geq \left|W_{k+1}W_j^*\right| = \left|w_{k+1,k+1}w_{j,k+1}^* + \hat{W_{k+1}}\hat{W_j}^*\right| \geq \left|w_{k+1,k+1}w_{j,k+1}^*\right| - \left|\hat{W_{k+1}}\hat{W_j}^*\right|$. Therefore $\left|w_{k+1,k+1}w_{j,k+1}^*\right| \leq 2\delta + 3\delta^2 + \left|\hat{W_{k+1}}\hat{W_j}^*\right| \leq 2\delta + 3\delta^2 + 2\delta(1+\delta) \leq 4\delta + 5\delta^2$. Therefore $|w_{j,k+1}| \leq \frac{4\delta + 5\delta^2}{1 - 2\delta}$. Finally, since we may assume that $\delta \leq 1/10$, we have $|w_{j,k+1}| \leq 6\delta$.

□

6.3 Carrying out near-trivial transformations

In this section, we show how to construct a single QTM that can carry out, at least approximately, any specified near-trivial transformation. Since a near-trivial transformation can apply an arbitrary rotation, either between two dimensions or in the phase of a single dimension, we must first show how a fixed rotation can be used to efficiently approximate an arbitrary rotation. Note that a single copy of this fixed rotation gives the only "non-classical" amplitudes (those other than $0, 1$) in the transition function of the universal QTM constructed below. See Adleman, et. al. [1] and Solovay and Yao [40] for the constructions of universal QTMs whose amplitudes are restricted to a small set of rationals.

Lemma 6.3.1 *Let* $\mathcal{R} = 2\pi \sum_{i=1}^{\infty} 2^{-2^i}$

Then there is a deterministic algorithm taking time polynomial in $\log 1/\epsilon$ *and the length of the input which on input* θ, ϵ *with* $\theta \in [0, 2\pi]$ *and* $\epsilon > 0$*, produces integer output* k *bounded by a polynomial in* $1/\epsilon$ *such that*

$$|k\mathcal{R} - \theta| \ mod\ 2\pi \ \leq \ \epsilon$$

Proof. First, we describe a procedure for computing such a k.

Start by calculating n, a power of 2, such that $\epsilon > \frac{2\pi}{2^{n-1}}$. Next, approximate $\frac{\theta}{2\pi}$ as a fraction with denominator 2^n. In other words, find an integer $m \in [1, 2^n]$ such that

$$\left|\frac{\theta}{2\pi} - \frac{m}{2^n}\right| \leq \frac{1}{2^n}$$

Then we can let $k = m2^n$ because

$$\begin{aligned} m2^n\mathcal{R} \bmod 2\pi &= \left(2\pi m \sum_{i=1}^{\infty} 2^{n-2^i}\right) \bmod 2\pi \\ &= \left(2\pi m \sum_{i=\log n+1}^{\infty} 2^{n-2^i}\right) \bmod 2\pi \\ &= \left(\frac{2\pi m}{2^n} + 2\pi m \sum_{i=\log n+2}^{\infty} 2^{n-2^i}\right) \bmod 2\pi \end{aligned}$$

and since

$$\begin{aligned} m \sum_{i=\log n+2}^{\infty} 2^{n-2^i} &\leq m2^{n-4n+1} \\ &\leq 2^{n-3n+1} \\ &\leq 2^{-2n+1} \end{aligned}$$

we have

$$\begin{aligned}
|m2^n\mathcal{R} - \theta| \bmod 2\pi &\le \left|m2^n\mathcal{R} - \frac{2\pi m}{2^n}\right| \bmod 2\pi \;+\; \left|\frac{2\pi m}{2^n} - \theta\right| \\
&\le \frac{2\pi}{2^{2n-1}} + \frac{2\pi}{2^n} \\
&< \frac{2\pi}{2^{n-1}} \\
&< \epsilon
\end{aligned}$$

□

At this point it should be clear that a single QTM can carry out any sequence of near-trivial transformations to any desired accuracy ϵ. We formalize this notion below, by showing that there is a QTM that accepts as input the descriptions of a sequence of near-trivial transformations, and an error bound ϵ, and applies an ϵ approximation of the product of these transformations on any given superposition. The formalization is quite tedious, and the reader is encouraged to skip the rest of the subsection if the above statement is convincing.

Below, we give a formal definition of what it means for a QTM to carry out a transformation.

Definition 6.3.2 *Let $\Sigma \cup \#$ be the alphabet of the first track of QTM M. Let $\mathcal{V}$ be the complex vector space of superpositions of k length strings over Σ. Let U be a linear transformation on $\mathcal{V}$, and let x_U be a string that encodes U (perhaps approximately). We say that x_U* causes *M to carry out the k cell transformation U with accuracy ϵ in time T if for every $|\phi\rangle \in \mathcal{V}$, on input $|\phi\rangle|x_U\rangle|\epsilon\rangle$, M halts in exactly T steps with its tape head back in the start cell and with final superposition $(U'|\phi\rangle)|x\rangle$, where U' is a unitary transformation on $\mathcal{V}$ such that $\|U - U'\| \le \epsilon$. Moreover, for a family A of transformations, we say that M carries out these transformations in polynomial time if T is bounded by a polynomial in $1/\epsilon$ and the length of the input.*

In the case that A contains transformations that are not unitary, we say that M carries out *the set of transformations A with* closeness factor *c if for any $\epsilon > 0$ and any $U \in A$ which is $c\epsilon$-close to unitary, there is a unitary transformation U' with $\|U' - U\| \le \epsilon$ such that $|x_U\rangle|\epsilon\rangle$ causes M to carry out the transformation U' in time which is polynomial in $1/\epsilon$ and the length of its input.*

Recall that a near-trivial transformation written as x, y, θ calls for a rotation between dimensions x and y of angle θ if $x \ne y$ and a phase shift of $e^{i\theta}$ to dimension x otherwise. So, we want to build a stationary, normal form QTM that takes as input $w; x, y, \theta; \epsilon$ and transforms w according to the near-trivial transformation described by x, y, θ with accuracy ϵ. We also need this QTM's running time to depend only on the length of w but not its value. If this is the case then the machine will also halt on an initial superposition of the form $|\phi\rangle|x, y, \theta\rangle|\epsilon\rangle$ where $|\phi\rangle$ is a superposition of equal-length strings w.

Lemma 6.3.3 *There is a stationary, normal form QTM M with first track alphabet $\{\#, 0, 1\}$ that carries out the set of near-trivial transformations on its first track in polynomial time.*

Proof. Using the encoding x, y, θ for near-trivial transformations described above in §6.2, we will show how to construct QTMs M_1 and M_2 with first track alphabet $\{\#, 0, 1\}$ such that M_1 carries out the set of near-trivial rotations on its first track in polynomial time, and M_2 carries out the set of near-trivial phase shifts on its first track in polynomial time. Using the Branching Lemma (page pagerefbranching.lemma on these two machines, we can construct a QTM that behaves correctly provided there is an extra track containing a 1 when $x = y$ and we have a phase shift to perform, and containing a 0 when $x \neq y$ and we have a rotation to perform. We can then construct the desired QTM M by dovetailing before and after with machines that compute and erase this extra bit based on x, y. The Synchronization Theorem lets us construct two such stationary, normal form QTMs whose running times depend only on the lengths of x and y. Therefore, this additional computation will not disturb the synchronization of the computation.

Next, we show how to construct the QTM M_1 to carry out near-trivial rotations.

It is easy to construct a QTM which on input b, applies a rotation by angle θ between $|0\rangle$ and $|1\rangle$, while leaving b alone if $b = \#$. But Lemma 6.3.1 above tells us that we can achieve any rotation by applying the single rotation $\mathcal{R}$ at most a polynomial number of times. Therefore the following five step process will allow us to apply a near-trivial rotation. We must be careful that when we apply the rotation, the two computational paths with $b \in \{0, 1\}$ differ only in b since otherwise they will not interfere.

1. Calculate k such that $k\ \mathcal{R} \bmod 2\pi \in [\theta - \epsilon, \theta + \epsilon]$.
2. Transform w, x, y into b, x, y, z where $b = 0$ if $w = x$, $b = 1$ if $w = y$ and $w \neq x$, and $b = \#$ otherwise, and where $z = w$ if $b = \#$ and z is the empty string otherwise.
3. Run the rotation applying machine k times on the first bit of z.
4. Reverse Step 2 transforming $\#, x, y, w$ with $w \neq x, y$ into w, x, y, transforming $0, x, y$ into x, x, y, and transforming $1, x, y$ with $x \neq y$ into y, x, y.
5. Reverse Step 1 erasing k.

We build the desired QTM M by constructing a QTM for each of these five steps and then dovetailing them together.

First, notice that the length of the desired output of Steps 1,2,4, and 5 can be computed just from the length of the input. Therefore, using Lemmas 6.3.1 from page 34 and the Synchronization Theorem from page 17, we can build polynomial time, stationary, normal form QTMs for Steps 1,2,4, and 5 which run in time which depends on the lengths of w, x, y, but not their particular values.

To complete the construction we must build a machine for the rotation with these same properties. The stationary, normal form QTM R with alphabet $\{\#, 0, 1\}$, state set $\{q_0, q_1, q_f\}$ and transition function defined by

	$\#$	0	1
q_0	$\|\#\rangle\|q_1\rangle\|L\rangle$	$\cos\mathcal{R}\|0\rangle\|q_1\rangle\|L\rangle + \sin\mathcal{R}\|1\rangle\|q_1\rangle\|L\rangle$	$-\sin\mathcal{R}\|0\rangle\|q_1\rangle\|L\rangle + \cos\mathcal{R}\|1\rangle\|q_1\rangle\|L\rangle$
q_1	$\|\#\rangle\|q_f\rangle\|R\rangle$		
q_f	$\|\#\rangle\|q_0\rangle\|R\rangle$	$\|0\rangle\|q_0\rangle\|R\rangle$	$\|1\rangle\|q_0\rangle\|R\rangle$

runs for constant time and applies rotation $\mathcal{R}$ between start cell contents $|0\rangle$ and $|1\rangle$ while leaving other inputs unchanged. Inserting R for the special state in the reversible TM from the Looping Lemma, we can construct a normal form QTM which applies rotation $k\mathcal{R}$ between inputs $|0, k\rangle$ and $|1, k\rangle$ while leaving input $|\#w, k\rangle$ unchanged. Since the machine we loop on is stationary and takes constant time regardless of its input, the resulting looping machine is stationary and takes time depending only on k.

Finally, with appropriate use of Lemmas 4.2.1 on page 17 and 4.2.2 on page 17 we can dovetail these five stationary, normal form QTMs to achieve a stationary, normal form QTM M that implements the desired computation. Since, the phase application machine run in time which is independent of w, x, and y, and the other four run in time which depends only on the length of w, x, and y, the running time of M depends on the length of w, x, and y but not on their particular values. Therefore it halts with the proper output not only if run on an input with a single string w, but also if run on an initial superposition with different strings w but with the same near-trivial transformation and ϵ.

The QTM M_2 to carry out near-trivial phase shifts is the same as M_1 except that we replace the transition function of the simple QTM which applies a phase shift rotation by angle $\mathcal{R}$ as follows giving a stationary QTM which applies phase shift $e^{i\mathcal{R}}$ if b is a 0 and phase shift 1 otherwise.

	#	0	1
q_0	$\|\#\rangle\|q_1\rangle\|L\rangle$	$e^{i\mathcal{R}}\|0\rangle\|q_1\rangle\|L\rangle$	$\|1\rangle\|q_1\rangle\|L\rangle$
q_1	$\|\#\rangle\|q_f\rangle\|R\rangle$		
q_f	$\|\#\rangle\|q_0\rangle\|R\rangle$	$\|0\rangle\|q_0\rangle\|R\rangle$	$\|1\rangle\|q_0\rangle\|R\rangle$

The proof that this gives the desired M_2 is identical to the proof for M_1 and is omitted. □

6.4 Carrying out a unitary transformation on a QTM

Now that we can approximate a unitary transformation by a product of near-trivial transformations, and we have a QTM to carry out the latter, we can build a QTM to apply an approximation of a given unitary transformation.

Theorem 6.4.1 (Unitary Transformation Theorem) *There is a stationary, normal form QTM M with first track alphabet $\{\#, 0, 1\}$ that carries out the set of all transformations on its first track in polynomial time with required closeness factor $\frac{1}{2(10\sqrt{d})^d}$ for transformations of dimension d.*

Proof. Given an $\epsilon > 0$ and a transformation U of dimension $d = 2^k$ which is $\frac{\epsilon}{2(10\sqrt{d})^d}$-close to unitary, we can carry out U to within ϵ on the first k cells of the first track using the following steps.

1. Calculate and write on clean tracks $\frac{\epsilon}{2n}$, and a list of near-trivial $U_1, \ldots, U_n$ such that $\|U - U_n \cdots U_1\| \leq \epsilon/2$, and such that n is polynomial in 2^k.
2. Apply the list of transformations $U_1, \ldots, U_n$, each to within $\frac{\epsilon}{2n}$.
3. Erase $U_1, \ldots, U_n$, and $\frac{\epsilon}{2n}$.

We can construct a QTM to accomplish these steps as follows. First, using Lemma 6.2.3 on page 32 and the Synchronization Theorem on page 17, we can build polynomial time, stationary, normal form QTMs for Steps 1 and 3 which run in time which depends only on U and ϵ. Finally, we can build a stationary, normal form QTM to accomplish Step 2 in time which is polynomial in 2^k and $1/\epsilon$ as follows. We have a stationary, normal form QTM constructed in Lemma 6.3.3 on page 35 to apply any specified near-trivial transformation to within a given bound ϵ. We dovetail this with a machine, constructed using the Synchronization Theorem on page 17, that rotates U_1 around to the end of the list of transformations. Since the resulting QTM is stationary, and takes time which depends only on ϵ and the U_i, we can insert it for the special state in the machine of the Looping Lemma to give the desired QTM for Step 2.

With appropriate use of Lemmas 4.2.1 on page 17 and 4.2.2 on page 17, dovetailing the QTMs for these three steps gives the desired M'. Since the running times of the three QTMs are independent of the contents of the first track, so is the running time of M'. Finally, notice that when we run M' we apply to the first k cells of the first track, we apply a unitary transformation U' with

$$\|U' - U\| \;\leq\; \|U' - U_n \cdots U_1\| + \|U_n \cdots U_1 - U\| \;\leq\; n\frac{\epsilon}{2n} + \frac{\epsilon}{2} \;\leq\; \epsilon$$

as desired. □

7 Constructing a universal QTM

A universal QTM must inevitably decompose one step of the simulated machine using many simple steps. A step of the simulated QTM is a mapping from the computational basis to a new orthonormal basis. A single step of the universal QTM can only map some of the computational basis vectors to their desired destinations. In general this partial transformation will not be unitary, because the destination vectors will not be orthogonal to the computational bases which have not yet been operated on. The key construction that enables us to achieve a unitary decomposition is the Unidirection Lemma. Applying this lemma, we get a QTM whose mapping of the computational basis has the following property; there is a decomposition of the space into subspaces of constant dimension, such that each subspace gets mapped onto another.

More specifically, we saw in the previous section that we can construct a QTM that carries out to a close approximation any specified unitary transformation. On the other hand, we have also noted that the transition function of a unidirectional QTM specifies a unitary transformation from superpositions of current state and tape symbol to superpositions of new symbol and state. This means a unidirectional QTM can be simulated by repeatedly applying this fixed-dimensional unitary transformation followed by the reversible deterministic transformation that moves the simulated tape head according to the new state. So, our universal machine will first convert its input to a unidirectional QTM using the construction of the Unidirection Lemma, and then simulate this new QTM by repeatedly applying this unitary transformation and reversible deterministic transformation.

Since we wish to construct a single machine that can simulate every QTM, we must build it in such a way that every QTM can be provided as input. Much of the definition of a QTM is easily encoded: We can write down the size of the alphabet Σ, with the first symbol assumed to be the blank symbol, and we can write down the size of the state set Q, with the first state assumed to be the start state. To complete the specification, we need to describe the transition function δ by giving the $2\,\mathrm{card}(\Sigma)^2\,\mathrm{card}(Q)^2$ amplitudes of the form $\delta(i_1, i_2, i_3, i_4, d)$. If we had restricted the definition of a QTM to include only machines with rational transition amplitudes, then we could write down each amplitude explicitly as the ratio of two integers. However, we have instead restricted the definition of a QTM to include only those machines with amplitudes in $\tilde{\mathbf{C}}$, which means that for each amplitude there is a deterministic algorithm which computes the amplitude's real and imaginary parts to within 2^{-n} in time polynomial in n. We will therefore specify δ by giving a deterministic algorithm that computes each transition amplitude to within 2^{-n} in time polynomial in n.

Since the universal QTM we will construct returns its tape head to the start cell after simulating each step of the desired machine, it will incur a slowdown which is (at least) linear in T. We conjecture that with more care a universal QTM can be constructed whose slowdown is only polylogarithmic in T.

Theorem 7.0.2 *There is a normal form QTM $\mathcal{M}$ such that for any well-formed QTM M, any $\epsilon > 0$, and any T, $\mathcal{M}$ can simulate M with accuracy ϵ for T steps with slowdown polynomial in T and $1/\epsilon$.*

Proof. As described above, our approach will be first to use the construction of the Unidirection Lemma to build a unidirectional QTM M' which simulates M with slowdown by a factor of 5, and then to simulate M'. We will first describe the simulation of M', and then return to describe the easily computable preprocessing that needs to be carried out.

So, suppose $M = (\Sigma, Q, \delta)$ is a unidirectional QTM that we wish to simulate on our universal QTM.

We start by reviewing the standard technique of representing the configuration of a target TM on the tape of a universal TM. We will use one track of the tape of our universal QTM to simulate the current configuration of M. Since the alphabet and state set of M could have any fixed size, we will use a series of $\log\,\mathrm{card}(Q \times \Sigma)$ cells of our tape, referred to as a "supercell", to simulate each cell of M. Each supercell holds a pair of integers p, σ where $\sigma \in [1, \mathrm{card}(\Sigma)]$ represents the contents of the corresponding cell of M, and $p \in [0, \mathrm{card}(Q)]$ represents the state of M, if its tape head is scanning the corresponding cell and $p = 0$ otherwise. Since the tape head of M can only move distance T away from the start cell in time T, we only need supercells for the $2T + 1$ cells at the center of M's tape (and we place markers to denote the ends).

Now we know that if we ignore the update direction, then δ gives a unitary transformation U of dimension $d = \mathrm{card}(Q \times \Sigma)$ from superpositions of current state and tape symbol to superpositions of new state and symbol. So, we can properly update the superposition on the simulation tracks if we first apply U to the current state and symbol of M, and then move the new state specification left or right one supercell according to the direction in which that state of M can be entered.

We will therefore build a QTM $STEP$ that carries out one step of the simulation as follows. In addition to the simulation track, this machine will is provided as input a desired accuracy γ, a specification of U (which is guaranteed to be $\frac{\gamma}{2(10\sqrt{d})^d}$-close to the unitary), and a string $s \in \{0,1\}^{\mathrm{card}(Q)}$ which gives the direction in which each state of M can be entered. The machine $STEP$ operates as follows.

1. Transfer the current state and symbol $p; \sigma$ to empty work space near the start cell, leaving a special marker in their place.
2. Apply U to $p; \sigma$ to within γ, transforming $p; \sigma$ into a superposition of new state and symbol $q; \tau$.
3. Reverse Step 1, transferring q, τ back to the marked, empty supercell (and emptying the work space).
4. Transfer the state specification q one supercell to the right or left depending whether the q^{th} bit of s is a 0 or 1.

Using the Synchronization Theorem on page 17, we can construct stationary, normal form QTMs for Steps 1, 3, and 4 which take time which is polynomial in T and (for a fixed M) depends only on T. Step 2 can be carried out in time polynomial in $\mathrm{card}(\Sigma), \mathrm{card}(Q)$, and γ with the unitary transformation applying QTM constructed in the Unitary Transformation Theorem. With appropriate use of Lemmas 4.2.1 on page 17 and 4.2.2 on page 17, dovetailing these four normal form QTMs gives us the desired normal form QTM $STEP$.

Since each of the four QTMs takes time which depends (for a fixed M) only on T and ϵ so does $STEP$. Therefore, if we insert $STEP$ for the special state in the reversible TM constructed in the Looping Lemma, and provide additional input T, the resulting QTM $STEP'$ will halt after time polynomial in T and $1/\epsilon$ after simulating T steps of M with accuracy $T\epsilon$.

Finally, we construct the desired universal QTM $\mathcal{M}$ by dovetailing $STEP'$ after a QTM which carries out the necessary preprocessing. In general, the universal machine must simulate QTMs which are not unidirectional. So, the preprocessing for desired QTM M, desired input x, and desired simulation accuracy

ϵ consists of first carrying out the construction of Unidirection Lemma to build a unidirectional QTM M' which simulates M with slowdown by a factor of 5^3. The following inputs are then computed for $STEP'$.

1. The proper $2T+1$ supercell representation of the initial configuration of M' with input x.
2. The d-dimensional transformation U for M' with each entry written to accuracy $\frac{\epsilon}{40T(10\sqrt{d})^{d+2}}$
3. The string of directions s for M'.
4. The desired number of simulation steps $5T$, and the desired accuracy $\gamma = \epsilon/40T$.

It can be verified that each of these inputs to $\mathcal{M}$ can be computed in deterministic time which is polynomial in T, $1/\epsilon$, and the length of the input. If the transformation U is computed to the specified accuracy, the transformation actually provided to $STEP$ will be within $\frac{\epsilon}{40T(10\sqrt{d})^d}$ of the desired unitary U, and so will be $\frac{\epsilon}{40T(10\sqrt{d}^d)}$-close to unitary as required for the operation of $STEP$. So, each time $STEP$ applies runs with accuracy $\epsilon/40T$, it will have applied a unitary transformation which is within $\epsilon/20T$ of U. Therefore, after $5T$ runs of $STEP$, we will have applied a unitary transformation which is within $\epsilon/4$ of the $5T$ step transformation of M'. This means that observing the simulation track of $\mathcal{M}$ after it has completed will give a sample from a distribution which is within total variation distance ϵ of the distribution sampled from by observing M on input x at time T. □

8 The computational power of QTMs

In this section, we explore the computational power of QTMs from a complexity theoretic point of view. It is natural to define quantum analogues of classical complexity classes [13]. In classical complexity theory, **BPP** is regarded as the class of all languages that are efficiently computable on a classical computer. The quantum analog of BPP — BQP (bounded-error quantum polynomial time) — should similarly be regarded as the class of all languages that are efficiently computable on a QTM.

8.1 Accepting languages with QTMs

Definition 8.1.1 *Let M be a stationary, normal form, multi-track QTM M whose last track has alphabet $\{\#, 0, 1\}$. If we run M with string x on the first track and the empty string elsewhere, wait until M halts* [4]*, and then observe the last track of the start cell, we will see a* 1 *with some probability p. We will say that M* accepts *x with probability p and* rejects *x with probability $1-p$.*

Consider any language $\mathcal{L} \subseteq (\Sigma - \#)^*$.

We say that QTM M exactly accepts *the $\mathcal{L}$ if M accepts every string $x \in \mathcal{L}$ with probability* 1 *and rejects every string $x \in (\Sigma - \#)^* - \mathcal{L}$ with probability* 1.

We define the class **EQP** (exact *or* error-free quantum polynomial time) *as the set of languages which are exactly accepted by some polynomial time QTM. More generally, we define the class* **EQTime** $(T(n))$

[3] Note that the transition function of M' is again specified with a deterministic algorithm, which depends on the algorithm for the transition function of M

[4] This can be accomplished by performing a measurement to check whether the machine is in the final state q_f. Making this partial measurement does not have any other effect on the computation of the QTM

as the set of languages which are exactly accepted by some QTM whose running time on any input of length n is bounded by $T(n)$.

A QTM accepts *the language $\mathcal{L} \subseteq (\Sigma - \#)^*$* with probability *$p$ if M accepts with probability at least p every string $x \in \mathcal{L}$ and rejects with probability at least p every string $x \in (\Sigma - \#)^* - \mathcal{L}$. We define the class* **BQP** *(*bounded-error quantum polynomial time*) as the set of languages which are accepted with probability 2/3 by some polynomial time QTM. More generally, we define the class* **BQTime** $(T(n))$ *as the set of languages which are accepted with probability 2/3 by some QTM whose running time on any input of length n is bounded by $T(n)$.*

The limitation of considering only stationary, normal form QTMs in these definitions is easily seen not to limit the computational power by appealing to the stationary, normal form universal QTM constructed in Theorem 7.0.2 on page 38.

8.2 Upper and lower bounds on the power of QTMs

Clearly **EQP** $\subseteq$ **BQP**. Since reversible TMs are a special case of QTMs, Bennett's results imply that **P** $\subseteq$ **EQP** and **BPP** $\subseteq$ **BQP**. We include these two simple proofs for completeness.

Theorem 8.2.1 P $\subseteq$ EQP

Proof. Let $\mathcal{L}$ be a language in **P**. Then there is some polynomial time deterministic algorithm that on input x produces output 1 if $x \in \mathcal{L}$ and 0 otherwise. Appealing to the Synchronization Theorem on page pagerefdet.synchronized, there is therefore a stationary, normal form QTM running in polynomial time which on input x produces output $x;1$ if $x \in \mathcal{L}$ and $x;0$ otherwise. This is an **EQP** machine accepting $\mathcal{L}$. $\Box$

Theorem 8.2.2 BPP $\subseteq$ BQP

Proof. Let $\mathcal{L}$ be a language in **BPP**. Then there must be a polynomial $p(n)$ and a polynomial time deterministic TM M with 0, 1 output which satisfy the following. For any string x of length n, if we call S_x the set of $2^{p(n)}$ bits computed by M on the inputs $x;y$ with $y \in \{0,1\}^{p(n)}$, then the proportion of 1's in S_x is at least 2/3 when $x \in \mathcal{L}$ and at most 1/3 otherwise. We can use a QTM to decide whether a string x in the language $\mathcal{L}$ by first breaking into a superposition split equally among all $|x\rangle|y\rangle$ and then running this deterministic algorithm.

First, we dovetail a stationary, normal form QTM which takes input x to output $x;0^{p(n)}$ with a stationary, normal form QTM constructed as in Theorem 8.4.1 below which applies a Fourier transform to the contents of its second track. This gives us a stationary, normal form QTM which on input x produces the superposition $\sum_{y\in\{0,1\}^{p(n)}} \frac{1}{2^{p(n)/2}}|x\rangle|y\rangle$. Dovetailing this with a synchronized, normal form version of M built according to the Synchronization Theorem on page pagerefdet.synchronized gives a polynomial time QTM which on input x produces a final superposition

$$\sum_{y\in\{0,1\}^{p(n)}} \frac{1}{2^{p(n)/2}}|x\rangle|y\rangle|M(x;y)\rangle$$

Since the proportion of 1's in S_x is at least 2/3 if $x \in \mathcal{L}$ and at most 1/3 otherwise, observing the bit on the third track will give the proper classification for string x with probability at least 2/3. Therefore, this is a **BQP** machine accepting the language $\mathcal{L}$. □

Clearly **BQP** is in exponential time. The next result gives the first non-trivial upper bound on **BQP**.

Theorem 8.2.3 BQP $\subseteq$ PSPACE

Proof. Let $M = (\Sigma, Q, \delta)$ be a **BQP** machine with running time $p(n)$.

According to Theorem 3.4.3 on page 13, any QTM M' which is

$$\frac{\epsilon}{24\,\mathrm{card}(\Sigma)\ \mathrm{card}(Q)p(n)}\text{-close}$$

to M will simulate M for $p(n)$ steps with accuracy ϵ. If we simulate M with accuracy $1/12$, then the success probability will still be at least $7/12$. Therefore, we need only work with the QTM M' where each transition amplitude from M is computed to its first $\log\left(288\,\mathrm{card}(\Sigma)\ \mathrm{card}(Q)p(n)\right)$ bits.

Now the amplitude of any particular configuration at time T is the sum of the amplitudes of each possible computational path of M' of length T from the start configuration to the desired configuration. The amplitude of each such path can be computed exactly in polynomial time. So, if we maintain a stack of at most $p(n)$ intermediate configurations we can carry out a depth-first search of the computational tree to calculate the amplitude of any particular configuration using only polynomial space (but exponential time).

Finally, we can determine whether a string x of length n is accepted by M by computing the sum of squared magnitudes at time $p(n)$ of all configurations that M' can reach that have a 1 in the start cell and comparing this sum to $7/12$. Clearly the only reachable "accepting" configurations of M' are those with a 1 in the start cell and blanks in all but the $2p(n)$ cells within distance $p(n)$ of the start cell. So, using only polynomial space, we can step through all of these configurations computing a running sum of the their squared magnitudes. □

Following Valiant's suggestion [43], the upper bound can be further improved to $\mathbf{P}^{\sharp\mathbf{P}}$. This proof can be simplified by using a theorem from [9] which shows how any **BQP** machine can be turned into a "clean" version M that on input x produces a final superposition with almost all of its weight on $x; M(x)$ where $M(x)$ is a 1 if M accepts x and a 0 otherwise. This means we need only estimate the amplitude of this one configuration in the final superposition of M.

Now, the amplitude of a single configuration can be broken down not only into the sum of the amplitudes of all of the computational paths that reach it, but also into the sum of positive real contributions, the sum of negative real contributions, the sum of positive imaginary contributions, and the sum of negative imaginary contributions. We will show that each of these four pieces can be computed using a $\sharp\mathbf{P}$ machine.

Recall that $\sharp\mathbf{P}$ is the set of functions f mapping strings to integers for which there exists a polynomial $p(n)$ and a language $\mathcal{L} \in \mathbf{P}$ such that for any string x, the value $f(x)$ is the number of strings y of length $p(|x|)$ for which xy is contained in $\mathcal{L}$.

Theorem 8.2.4 *If the language $\mathcal{L}$ is contained in the class* **BQTime** $(T(n))$ *with $T(n) > n$, with $T(n)$ time-constructible, then for any $\epsilon > 0$, there is a QTM M' which accepts $\mathcal{L}$ with probability $1 - \epsilon$ and has the following property. When run on input x of length n, M' runs for time bounded by $cT(n)$, where c is a polynomial in $\log 1/\epsilon$, and produces a final superposition in which $|x\rangle|\mathcal{L}(x)\rangle$, with $\mathcal{L}(x) = 1$ if $x \in \mathcal{L}$ and 0 otherwise, has squared magnitude at least $1 - \epsilon$.*

Theorem 8.2.5 BQP $\subseteq \mathbf{P}^{\sharp\mathbf{P}}$

Proof. Let $M = (\Sigma, Q, \delta)$ be a **BQP** machine with observation time $p(n)$. Appealing to Theorem 8.2.4 above, we can without loss of generality that M is a 'clean **BQP** machine'. i.e. on any input x, at time $p(|x|)$, the squared magnitude of the final configuration with output $x; M(x)$ is at least $2/3$.

Now as above, we will appeal to Theorem 3.4.3 on page 13 which tells us that if we work with the QTM M' where each amplitude of M is computed to its first $b = \log(288 \operatorname{card}(\Sigma) \operatorname{card}(Q)p(n))$ bits, and we run M' on input x, then the squared magnitude at time $p(|x|)$ of the final configuration with output $x; M(x)$ will be at least $7/12$.

We will carry out the proof by showing how to use an oracle for the class $\sharp\mathbf{P}$ to efficiently compute with error magnitude less than $1/36$ the amplitude of the final configuration $x; 1$ of M' at time T. Since the true amplitude has magnitude at most 1, the squared magnitude of this approximated amplitude must be within $1/12$ of the squared magnitude of the true amplitude. To see this, just note that if α is the true amplitude, and $\|\alpha' - \alpha\| < 1/36$, then

$$\left| \|\alpha\|^2 - \|\alpha'\|^2 \right| \leq \|\alpha' - \alpha\|^2 + 2\|\alpha\|\|\alpha' - \alpha\| \leq 1/36(2 + 1/36) < 1/12$$

Since the success probability of M' is at least $7/12$, comparing the squared magnitude of this approximated amplitude to $1/2$ lets us correctly classify the string x.

We will now show how to approximate the amplitude described above. First, notice that the amplitude of a configuration at time T is the sum of the amplitudes of each computational path of length T from the start configuration to the desired configuration. The amplitude of any particular path is the product of the amplitudes in the transition function of M' used in each step along the path. Since each amplitude consists of a real part and an imaginary part, we can think of this product as consisting of the sum of 2^T terms each of which is either purely real or purely imaginary. So, the amplitude of the desired configuration at time T is the sum of these 2^T terms over each path. We will break this sum into four pieces, the sum of the positive real terms, the sum of the negative real terms, the sum of the positive imaginary terms, and the sum of the negative imaginary terms, and we will compute each to within error magnitude less than $1/144$ with the aid of a $\sharp\mathbf{P}$ algorithm. Taking the difference of the first two and the last two will then give us the amplitude of the desired configuration to with error of magnitude at most $1/36$ as desired.

We can compute the sum of the positive real contributions for all paths as follows. Suppose for some fixed constant c which is polynomial in T we are given the following three inputs, all of length polynomial in T: a specification of a T step computational path p of M', a specification t of one of the 2^T terms, and an integer w between 0 and 2^{cT}. Then it is easy to see that we could decide in deterministic time polynomial in T whether p is really a path from the start configuration of M on x to the desired final configuration, whether the term t is real and positive, and whether the term t^{th} term of the amplitude for path p is greater than $w/2^{cT}$. If we fix a path p and term t satisfying these constraints, then the number of w for which this algorithm accepts, divided by 2^{cT}, is within $1/2^{cT}$ of the value of the t^{th} for path p. So, if we fix only a path p satisfying these constraints, then the number of t, w for which the algorithm accepts, divided by 2^{cT}, is within $1/2^{(c-1)T}$ of the sum of the positive real terms for path p. Therefore, the number of p, t, w for which the algorithm accepts, divided by 2^{cT}, is within $N/2^{(c-1)T}$ of the sum of all of the positive real terms of all of the T step paths of M' from the start configuration to the desired configuration. Since there are at most $2\operatorname{card}(\Sigma)\operatorname{card}(Q)$ possible successors for any configuration in a legal path of M, choosing $c > 1 + \frac{\log 144}{T} + \frac{2\operatorname{card}(\Sigma)\operatorname{card}(Q)\log T}{T}$ gives $N/2^{(c-1)T} < 1/144$ as desired. Similar reasoning gives $\mathbf{P}^{\sharp\mathbf{P}}$ algorithms for approximating each of the remaining three sums of terms. $\square$

8.3 Oracle QTMs

In this subsection and the next, we will assume without loss of generality that the TM alphabet for each track is $\{0, 1, \#\}$. Initially all tracks are blank except that the input track contains the actual input surrounded by blanks. We will use Σ to denote $\{0, 1\}$.

In the classical setting, an oracle may be described informally as a device for evaluating some Boolean function $f : \Sigma^* \rightarrow \Sigma$, on arbitrary arguments, at unit cost per evaluation. This allows to formulate questions such as "if f were efficiently computable by a TM, which other functions (or languages) could be efficiently computed by TMs?". In this section we define oracle QTMs so that the equivalent question can be asked in the quantum setting.

An oracle QTM has a special *query track* on which the machine will place its questions for the oracle. Oracle QTMs have two distinguished internal states: a pre-query state q_q and a post-query state q_a. A query is executed whenever the machine enters the pre-query state with a single (non-empty) block of non-blank cells on the query track [5]. Assume that the non-blank region on the query tape is in state $|x \cdot b\rangle$ when the pre-query state is entered, where $x \in \Sigma^*$, $b \in \Sigma$, and "$\cdot$" denotes concatenation. Let f be the Boolean function computed by the oracle. The result of the oracle call is that the state of the query tape becomes $|x \cdot b \oplus f(x)\rangle$, where "$\oplus$" denotes the exclusive-or (addition modulo 2), after which the machine's internal control passes to the post-query state. Except for the query tape and internal control, other parts of the oracle QTM do not change during the query. If the target bit $|b\rangle$ was supplied in initial state $|0\rangle$, then its final state will be $|f(x)\rangle$, just as in a classical oracle machine. Conversely, if the target bit is already in state $|f(x)\rangle$, calling the oracle will reset it to $|0\rangle$, a process known as "uncomputing", which is essential for proper interference to take place.

The power of quantum computers comes from their ability to follow a coherent superposition of computation paths. Similarly oracle QTMs derive great power from the ability to perform superpositions of queries. For example, an oracle for Boolean function f might be called when the query tape is in state $|\psi, 0\rangle = \sum_x \alpha_x |x, 0\rangle$, where α_x are complex coefficients, corresponding to an arbitrary superposition of queries with a constant $|0\rangle$ in the target bit. In this case, after the query, the query string will be left in the entangled state $\sum_x \alpha_x |x, f(x)\rangle$.

That the above definition of oracle QTMs yields unitary evolutions is self-evident if we restrict ourselves to machines that are well-formed in other respects, in particular evolving unitarily as they enter the pre-query state and leave the post-query state.

Let us define $\mathbf{BQTime}\,(T(n))^O$ as the sets of languages accepted with probability at least $2/3$ by some oracle QTM M^O whose running time is bounded by $T(n)$. This bound on the running time applies to each individual input, not just on the average. Notice that whether or not M^O is a $\mathbf{BQP}$-machine might depend upon the oracle O—thus M^O might be a $\mathbf{BQP}$-machine while $M^{O'}$ might not be one.

We have carefully defined oracle QTMs so that the same technique used to reverse a QTM in the Reversal Lemma can also be used to reverse an oracle QTM.

Lemma 8.3.1 *If M is a normal form, unidirectional oracle QTM then there is a normal form oracle QTM M' such that for any oracle O, M'^O reverses the computation of M^O while taking two extra time steps.*

Proof. Let $M = (\Sigma, Q, \delta)$ be a normal form, unidirectional oracle QTM with initial and final states q_0 and q_f and with query states $q_q \neq q_f$ and q_a. We construct M' from M exactly as in the proof of the

[5] Since a QTM must be careful to avoid leaving around intermediate computations on its tape, requiring that the query track contain only the query string adds no further difficulty to the construction of oracle QTMs

Reversal Lemma. We further give M' the same query states q_q, q_a as M, but with roles reversed. Since M is in normal form, and $q_q \neq q_f$, we must have $q_a \neq q_0$. Recall that the transition function of M' is defined so that for $q \neq q_0$

$$\delta'(q, \tau) = \sum_{p,\sigma} \delta(p, \sigma, \tau, q, d_q)^* |\sigma\rangle |p\rangle |\hat{d}_p\rangle$$

Therefore, since state q_q always leads to state q_a in M, state q_a always leads to state q_q in M'. Therefore M' is an oracle QTM.

Next, note that since the tape head position is irrelevant for the functioning of the oracle, the operation of the oracle in M'^O reverses the operation of the oracle in M^O. Finally, the same argument used in the Reversal Lemma can be used again here to prove that M'^O is well-formed and that M'^O reverses the computation of M^O while taking two extra time steps. □

The above definition of a quantum oracle for an arbitrary Boolean function will suffice for the purposes of the present paper, but the ability of quantum computers to perform general unitary transformations suggests a broader definition, which may be useful in other contexts. For example, oracles that perform more general, non-Boolean unitary operations have been considered in computational learning theory [15], and used to obtain large separations [32] between quantum and classical relativized complexity classes. See [9] for a discussion of more general definitions of oracle quantum computing.

8.4 Fourier Sampling and the power of QTMs

In this section we give evidence that QTMs are more powerful than bounded-error probabilistic TMs. We define the recursive Fourier sampling problem which on input the program for a boolean function takes on value 0 or 1. We show that the recursive Fourier sampling problem is in **BQP**. On the other hand, we prove that if the boolean function is specified by an oracle, then the recursive Fourier sampling problem is not in **BQTime** $(n^{o(\log n)})$. This result provided the first evidence that QTMs are more powerful than classical probabilistic TMs with bounded error probability [11].

One could ask what the relevance of these oracle results is, in view of the non-relativizing results on probabilistically checkable proofs [36, 3]. Moreover, Arora et al. [2] make the case that the fact that the **P** versus **NP** question relativizes does not imply that the question "cannot be resolved by current techniques in complexity theory". On the other hand, in our oracle results (and also in the subsequent results of [39]), the key property of oracles that is exploited is their back-box nature, and for the reasons sketched below, these results did indeed provide strong evidence that **BQP** $\neq$ **BPP** (of course, the later results of [36] gave even stronger evidence). This is because if one assumes that **P** $\neq$ **NP** and the existence of one-way functions (both these hypotheses are unproven, but widely believed by complexity theorists), it is also reasonable to assume that, in general, it is impossible to (efficiently) figure out the function computed by a program by just looking at its text (i.e. without explicitly running it on various inputs). Such a program would have to be treated as a black-box. Of course, such assumptions cannot be made if the question at hand is whether $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

Fourier Sampling: Consider the vector space of complex valued functions $f : Z_2^n \to \mathbf{C}$. There is a natural inner product on this space given by

$$(f, g) = \sum_{x \in Z_2^n} f(x) g(x)^*$$

The standard orthonormal basis for the vector space is the set of delta functions $\delta_y : Z_2^n \to \mathbf{C}$, given by $\delta_y(y) = 1$ and $\delta_y(x) = 0$ for $x \neq y$. Expressing a function in this basis is equivalent to specifying its values at each point in the domain. The characters of the group Z_2^n yield a different orthonormal basis consisting of the parity basis functions $\chi_s : Z_2^n \to \mathbf{C}$, given by $\chi_s(x) = \frac{-1^{s \cdot x}}{2^{n/2}}$, where $s \cdot x = \sum_{i=1}^n s_i x_i$. Given any function $f : Z_2^n \to \mathbf{C}$, we may write it in the new parity basis as $f = \sum_s \hat{f}(s)\chi_s$, where $\hat{f} : Z_2^n \to \mathbf{C}$ is given by $\hat{f}(s) = (f, \chi_s)$. The function $\hat{f}$ is called the discrete Fourier transform or Hadamard transform of f. Here the latter name refers to the fact that the linear transformation that maps a function f to its Fourier transform $\hat{f}$ is the Hadamard matrix $\mathcal{H}_n$. Clearly this transformation is unitary, since it is just effecting a change of basis from one orthonormal basis (the delta function basis) to another (the parity function basis). It follows that $\sum_x |f(x)|^2 = \sum_s |\hat{f}(s)|^2$.

Moreover, the discrete Fourier transform on Z_2^n can be written as the Kronecker product of the transform on each of n copies of Z_2 — the transform in that case is given by the matrix $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$. This fact can be used to write a simple and efficient QTM that effects the discrete Fourier transformation in the following sense: suppose the QTM has tape alphabet $\{0, 1, \#\}$, and the initially only n tape cells contain non-blank symbols. Then we can express the initial superposition as $\sum_{x \in \{0,1\}^n} f(x)|x\rangle$, where $f(x)$ is the amplitude of the configuration with x in the non-blank cells. Then there is a QTM that effects the Fourier transformation by applying the above transform on each of the n bits in the n non-blank cells. This takes $O(n)$ steps, and results in the final configuration $\sum_{s \in \{0,1\}^n} \hat{f}(s)|s\rangle$ (Theorem 8.4.1 below). Notice that this fourier transformation is taking place over a vector space of dimension 2^n. On the other hand, the result of the fourier transform — $\hat{f}$ — resides in the amplitudes of the quantum superposition, and is thus not directly accessible. We can, however, perform a measurement on the n non-blank tape cells, to obtain a sample from the probability distribution P such that $P[s] = |\hat{f}(s)|^2$. We shall refer to this operation as Fourier sampling. As we shall see, this is a very powerful operation. The reason is that each value $\hat{f}(s)$ depends upon all the (exponentially many values) of f, and it does not seem possible to anticipate which values of s will have large probability (constructive interference) without actually carrying out an exponential search.

Given a boolean function $g : Z_2^n \to \{1, -1\}$, we may define a function $f : Z_2^n \to \mathbf{C}$ of norm 1 by letting $f(x) = g(x)/2^{n/2}$. Following Deutsch and Josza [22] if g is a polynomial time computable function, there is a QTM that produces the superposition $\sum_{x \in \{0,1\}^n} g(x)|x\rangle$ in time polynomial in n. Combining this with the Fourier sampling operation above, we get a polynomial time QTM that samples from a certain distribution related to the given polynomial time computable function g (Theorem 8.4.2 below). We shall call this composite operation Fourier sampling with respect to g.

Theorem 8.4.1 *There is a normal form QTM which when run on an initial superposition of n-bit strings $|\phi\rangle$, halts in time $2n + 4$ with its tape head back in the start cell, and produces final superposition $\mathcal{H}_n|\phi\rangle$.*

Proof. We can construct the desired machine using the alphabet $\{\#, 0, 1\}$ and the set of states $\{q_0, q_a, q_b, q_c, q_f\}$. The machine will operate as follows when started with a string of length n as input. In state q_0, the machine steps right and left and enters state q_b. In state q_b, the machine steps right along the input string until it reaches the $\#$ at the end and enters state q_c stepping back left to the string's last symbol. During this rightward scan, the machine applies the transformation $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$ to the bit in each cell. In state q_c, the machine steps left until it reaches the $\#$ to the left of the start cell, at which point steps back right and halts. The following gives the quantum transition function of the desired machine.

	#	0	1
q_0		$\|0\rangle\|q_a\rangle\|R\rangle$	$\|1\rangle\|q_a\rangle\|R\rangle$
q_a	$\|\#\rangle\|q_b\rangle\|L\rangle$		
q_b	$\|\#\rangle\|q_c\rangle\|L\rangle$	$\frac{1}{\sqrt{2}}\|0\rangle\|q_b\rangle\|R\rangle + \frac{1}{\sqrt{2}}\|1\rangle\|q_b\rangle\|R\rangle$	$\frac{1}{\sqrt{2}}\|0\rangle\|q_b\rangle\|R\rangle - \frac{1}{\sqrt{2}}\|1\rangle\|q_b\rangle\|R\rangle$
q_c	$\|\#\rangle\|q_f\rangle\|L\rangle$	$\|0\rangle\|q_c\rangle\|L\rangle$	$\|1\rangle\|q_c\rangle\|L\rangle$
q_f	$\|\#\rangle\|q_0\rangle\|R\rangle$	$\|0\rangle\|q_0\rangle\|R\rangle$	$\|1\rangle\|q_0\rangle\|R\rangle$

It is can be easily verified that the Completion Lemma can be used to extend this machine to a well-formed QTM, and that this machine has the desired behavior described above. □

Theorem 8.4.2 *For every polynomial time computable function $g : \{0,1\}^n \to \{-1,1\}$, there is a polynomial time, stationary QTM where observing the final superposition on input 0^n gives each n-bit string s with probability $|\hat{f}(s)|^2$, where $f(x) = g(x)/2^{n/2}$.*

Proof. We build a QTM to carry out the following steps.

1. Apply a Fourier transform to 0^n to produce $\sum_{i \in \{0,1\}^n} \frac{1}{2^{n/2}} |i\rangle$.
2. Compute f to produce $\sum_{i \in \{0,1\}^n} \frac{1}{2^{n/2}} |i\rangle|f(i)\rangle$.
3. Apply a phase-applying machine to produce $\sum_{i \in \{0,1\}^n} \frac{1}{2^{n/2}} f(i)|i\rangle|f(i)\rangle$.
4. Apply the reverse of the machine in Step 2 to produce $\sum_{i \in \{0,1\}^n} \frac{1}{2^{n/2}} f(i)|i\rangle$.
5. Apply another Fourier transform to produce $\sum_{i \in \{0,1\}^n} \frac{1}{2^{n/2}} \hat{f}(s)|s\rangle$ as desired.

We have already constructed above the Fourier transform machine for Steps 1 and 5. Since f can be computed in deterministic polynomial time, the Synchronization Theorem on page 17 lets us construct polynomial time QTMs for Steps 2 and 4 which take input $|i\rangle$ to output $|i\rangle|f(i)\rangle$ and vice versa, with running time independent of the particular value of i.

Finally, we can apply phase according to $f(i)$ in Step 3 by extending to two tracks the stationary, normal form QTM with alphabet $\{\#, -1, 1\}$ defined by

	#	-1	1
q_0	$\|\#\rangle\|q_1\rangle\|R\rangle$	-$\|$-1$\rangle\|q_1\rangle\|R\rangle$	$\|1\rangle\|q_1\rangle\|R\rangle$
q_1	$\|\#\rangle\|q_f\rangle\|L\rangle$	$\|$-1$\rangle\|q_f\rangle\|L\rangle$	$\|1\rangle\|q_f\rangle\|L\rangle$
q_f	$\|\#\rangle\|q_0\rangle\|R\rangle$	$\|$-1$\rangle\|q_0\rangle\|R\rangle$	$\|1\rangle\|q_0\rangle\|R\rangle$

Dovetailing these five stationary, normal form QTMs gives the desired machine. □

Remarkably, this quantum algorithm performs Fourier sampling with respect to f while calling the algorithm for f only twice. To see more clearly why this is remarkable, consider the special case of the sampling problem where the function f is one of the (unnormalized) parity functions. Suppose f corresponds to the parity function χ_k; i.e. $f(i) = (-1)^{i \cdot k}$, where $i, k \in \{0,1\}^n$. Then the result of fourier sampling with respect to f is always k. Let us call this promise problem the parity problem: on input a program that computes f where f is an (unnormalized) parity function, determine k. Notice that the

QTM for fourier sampling extracts n bits of information (the value of k) using just two invocations of the subroutine for computing boolean function f. It is not hard to show that if we f is specified by an oracle, then in a probabilistic setting, extracting n bits of information must require at least n invocations of the boolean function. We now show how to amplify this this advantage of quantum computers over probabilistic computers by defining a recursive version of the parity problem: the recursive Fourier sampling problem.

To ready this problem for recursion, we first need to turn the parity problem into a problem with range $\{-1, 1\}$. This is easily done by adding a second function g, and requiring the answer $g(k)$ rather than k itself.

Now, we will make the problem recursive. For each problem instance of size n, we will replace the 2^n values of f with 2^n independent recursive subproblems of size $n/2$, and so on, stopping the recursion with function calls at the bottom. Since a QTM needs only two calls to f to solve the parity problem, it will be able to solve an instance of the recursive problem of size n recursively in time $T(n)$ where

$$T(n) \;\leq\; poly(n) + 4T(n/2) \qquad \Longrightarrow \qquad T(n) \leq poly(n)$$

However, since a PTM needs n calls to f to solve the parity problem, the straightforward recursive solution to the recursive problem on a PTM will require time $T(n)$ where

$$T(n) \;\geq\; nT(n/2) \qquad \Longrightarrow \qquad T(n) \;\geq\; n^{\log n}$$

To allow us to prove a separation between quantum and classical machines, we will replace the functions with an oracle. For any "legal" oracle O, any oracle satisfying some special constraints to be described below, we will define the language $\mathcal{R}_O$. We will show that there is a QTM which given access to any legal oracle O, accepts $\mathcal{R}_O$ in time $O(n \log n)$ with success probability 1, but that there is a legal oracle O such that $\mathcal{R}_O$ is not contained in $\mathbf{BPTime}\,(n^{o(\log n)})^O$.

Our language will consist of strings from $\{0,1\}^*$ with length a power of 2. We will call all such strings *candidates*. The decision whether a candidate x is contained in the language will depend on a recursive tree. If candidate x has length n, then it will have $2^{n/2}$ children in the tree, and in general a node at level $l \geq 0$ (counting from the bottom with leaves at level -1 for convenience) will have 2^{2^l} children. So, the root of the tree for a candidate x of length n is identified by the string x, its $2^{n/2}$ children are each identified by a string $x\$x_1$ where $x_1 \in \{0,1\}^{n/2}$, and, in general, a descendent in the tree at level l is identified by a string of the form $x\$x_1\$x_2\$\ldots\$x_m$ with $|x| = n, |x_1| = n/2, \ldots, |x_m| = 2^{l+1}$. Notice that any string of this form for some n a power of 2 and l
$in[0, \log n - 1]$ defines a node in some tree. So, we will consider oracles O which map queries over the alphabet $\{0, 1, \$\}$ to answers $\{-1, +1\}$, and we will use each such oracle O to define a function V_O that gives each node a value $\{-1, +1\}$. The language $\mathcal{R}_O$ will contain exactly those candidates x for which $V_O(x) = 1$.

We will define V_O for leaves (level 0 nodes) based directly on the oracle O. So, if x is a leaf, then we let $V_O(x) = O(x)$.

We will define V_O for all other nodes by looking at the answer of O to a particular query which is chosen depending on the values of V_O for its children. Consider node x at level $l \geq 0$. We will insist that the oracle O be such that there is some $k_x \in \{0,1\}^{2^l}$ such that the children of x all have values determined by their parity with k_x

$$\forall y \in \{0,1\}^{2^l} \quad V_O(x\$y) = (-1)^{y \cdot k_x}$$

and we will then give $V_O(x)$ the value $O(x\$k_x)$. We will say that the oracle O is *legal*, if this process allows us to successfully define V_O for all nodes whose level is ≥ 0. Any query of the form $x\$k$ with x a node at level $l \geq 0$ and $k \in \{0,1\}^{2^l}$, or of the form x with x a leaf, is called a *query at node* x. A query which is located in the same recursive tree as node x, but not in the subtree rooted at x, is called *outside of* x. Notice that for any candidate x, the values of V_O at nodes in the tree rooted at x and the decision whether x is in $\mathcal{R}_O$ all depend only on the answers of queries located at nodes in the tree rooted at x.

Theorem 8.4.3 *There is an oracle QTM M such that for every legal oracle O, M^O runs in polynomial time and accepts the language $\mathcal{R}_O$ with probability 1.*

Proof. We have built the language $\mathcal{R}_O$ so that it can be accepted efficiently using a recursive quantum algorithm. To avoid working through the details required to implement a recursive algorithm reversibly, we will instead implement the algorithm with a machine that writes down an iteration that "unwraps" the desired recursion. Then the Looping Lemma will let us build a QTM to carry out this iteration.

First consider the recursive algorithm to compute V_O for a node x. If x is a leaf, then the value $V_O(x)$ can be found by querying the string $x\$$. If x is a node at level $l \geq 0$, then calculate V_O as follows.

1. Split into an equal superposition of the 2^{2^l} children of x.
2. Recursively compute V_O for these children in superposition.
3. Apply phase to each child given by that child's value V_O.
4. Reverse the computation of Step 2 to erase the value V_O.
5. Apply the Fourier transform converting the superposition of children of x into a superposition consisting entirely of the single string k_x.
6. Query $x\$k_x$ to find the value V_O for x.
7. Reverse Steps 1-5 to erase k_x (leaving only x and $V_O(x)$).

Notice that the number of steps required in the iteration obeys the recursion discussed above, and hence is polynomial in the length of x. So, we can use the Synchronization Theorem on page 17 to construct a polynomial time QTM which, for any particular x, writes a list of the steps which must be carried out. Therefore, we complete the proof by constructing a polynomial time QTM to carry out any such a list of step.

Since our algorithm requires us to recursively run both the algorithm and its reverse, we need to see how to handle each step and its reverse. Before we start, we will fill out the node x to a description of a leaf by adding strings of 0's. Then, Steps 1 and 5 at level $l \geq 0$ just require applying the Fourier transform QTM to the 2^l bit string at level l in the current node description. Since the Fourier transform is its own reverse, the same machine also reverses Steps 1 and 5. Step 3 is handled by the phase-applying machine already constructed above as part of the Fourier sampling QTM in Theorem 8.4.2. Again, the transformation in Step 3 is its own reverse, so the same machine can be used to reverse Step 3. Step 6 and its reverse can be handled by a reversible TM which copies the relevant part of the current node description, queries the oracle, and then returns the node description from the query tape. Notice that each of these machines takes time which depends only on the length of its input.

Since we have stationary, normal form QTMs to handle each step at level l and its reverse in time bounded by a polynomial in 2^l, we can use the Branching Lemma to construct a stationary, normal form QTM to carry out any specified step of the computation. Dovetailing with a machine which rotates the first step in the list to the end, and inserting the resulting machine into the reversible TM of the Looping Lemma gives the desired QTM. □

Computing the function V_O takes time $\Omega(n^{\log n})$ even for a probabilistic computer allowed a bounded probability of error. We can see this with the following intuition. First, consider asking some set of queries of a legal O that determine the value of V_O for a node x described by the string $x_1\$\ldots\$x_m$ at level l. There are two ways that the asked queries might fix the value at x. The first is that the queries outside of the subtree rooted at x might be enough to fix V_O for x. If this happens, we say that $V_O(x)$ is *fixed by constraint*. An example of this is that if we asked all of the queries in the subtrees rooted at all of the siblings of x, then we have fixed V_O for all of the siblings, thereby fixing the string k such that $V_O(x_1\$\ldots\$x_m\$y)$ equals $(-1)^{y\cdot k}$.

The only other way that the value of the node x might be fixed is the following. If the queries fix the value of V_O for some of the children of x then this will restrict the possible values for the string k_x such that $V_O(x\$y)$ always equals $(-1)^{y\oplus k_x}$, and such that $V_O(x) = O(x\$k_x)$. If the query $x\$k_x$ for each possible k_x has been asked and all have the same answers, then this fixes the value V_O at x. If the query $x\$k_x$ for the correct k_x has been asked, then we call x a *hit*.

Now notice that fixing the values of a set of children of a level l node x restricts the value of k_x to a set of $2^{2^l - c}$ possibilities where c is the maximum size of a linearly independent subset of the children whose values are fixed. This can be used to prove that it takes $n \cdot n/2 \cdots 1 = n^{\Omega(\log n)}$.

However, this intuition is not yet enough since we are interested in arguing against a probabilistic TM rather than a deterministic TM. So, we will argue not just that it takes $n^{\Omega(\log n)}$ queries to fix the value of a candidate of length n, but that if fewer than $n^{\Omega(\log n)}$ queries are fixed, then choosing a random legal oracle consistent with those queries gives to a candidate of length n the value 1 with probability extremely close to 1/2. This will give the desired result. To see this, call the set of queries actually asked and the answers given to those queries a *run* of the probabilistic TM. We will have shown that if we take any run on a candidate of length n with fewer than $n^{\Omega(\log n)}$ queries, then the probability that a random legal oracle agreeing with the run assigns to x the value 1 is extremely close to 1/2. This means that a probabilistic TM whose running time is $n^{o(\log n/2)}$ will fail with probability 1 to accept $\mathcal{R}_O$ for a random legal oracle O.

Definition 8.4.4 *A* run of size k *is defined as a pair* S, f *where* S *is a set of* k *query strings and* f *is map from* S *to* $\{0,1\}$ *such that there is at least one legal oracle agreeing with* f.

Let r *be a run, let* y *be a node at level* $l \geq 2$, *and let* O *be a legal oracle at and below* y *which agrees with* r. *Then* O *determines the string* k_y *for which* $V_O(y) = O(y\$k_y)$. *If* $y\$k_y$ *is a query in* r, *then we say* O makes y a hit for r. *Suppressing the dependency on* r *in the notation, we define* $P(y)$ *as the probability that* y *is a hit for* r *when we choose a legal oracle at and below* y *uniformly at random from the set of all such oracle at and below* y *which agree with* r. *Similarly, for* x *an ancestor of* y, *we define* $P_x(y)$ *as the probability that* y *is a hit when a legal oracle is chosen at and below* x *uniformly at random from the set of all oracle at and below* x *which agree with* r.

Lemma 8.4.5 $P_x(y) \leq 2P(y)$

Proof. Let S be the set of legal oracles at and below y which agree with r. We can write S as the disjoint union of S_h and S_n where the former is the set of those oracles in S that make y a hit for r. Further splitting according to the value $V_O(y)$, we can write S as the disjoint union of four sets, $S_{h+}, S_{h-}, S_{n+}, S_{n-}$. Using this notation, we have $P(y) = \frac{\mathrm{card}(S_h)}{\mathrm{card}(S_h)+\mathrm{card}(S_n)}$. It is easy to see that, since the oracles in S_n do not make y a hit for r, $\mathrm{card}(S_{n+}) = \mathrm{card}(S_{n-}) = \mathrm{card}(S_n)/2$.

Next consider the set T of all legal oracles defined at and below x, but outside y, which agree with r. Each oracle $O \in T$ determines by constraint the value $V_O(y)$, but leaves the string k_y completely undetermined. If we again write T as the disjoint union of T_+ and T_- according to the constrained value $V_O(y)$, we notice that the set of legal oracles at and below x is exactly $(T_+ \times S_+) \cup (T_- \times S_-)$. So, we have

$$\begin{aligned} P_x(y) &= \frac{\mathrm{card}(T_+)\mathrm{card}(S_{h+}) + \mathrm{card}(T_-)\mathrm{card}(S_{h-})}{\mathrm{card}(T_+)\mathrm{card}(S_+) + \mathrm{card}(T_-)\mathrm{card}(S_-)} \\ &= \frac{\mathrm{card}(T_+)\mathrm{card}(S_{h+}) + \mathrm{card}(T_-)\mathrm{card}(S_{h-})}{\mathrm{card}(T_+)\mathrm{card}(S_{h+}) + \mathrm{card}(T_-)\mathrm{card}(S_{h-}) + \mathrm{card}(T)\mathrm{card}(S_n)/2} \end{aligned}$$

Without loss of generality, let $\mathrm{card}(T_+) \geq \mathrm{card}(T_-)$. Then since $n/(n+c)$ with $c, n > 0$ increases with n, we have

$$\begin{aligned} P_x(y) &\leq \frac{\mathrm{card}(T_+)\mathrm{card}(S_h)}{\mathrm{card}(T_+)\mathrm{card}(S_h) + \mathrm{card}(T)\mathrm{card}(S_n)/2} \\ &\leq \frac{\mathrm{card}(T_+)\mathrm{card}(S_h)}{\mathrm{card}(T_+)\mathrm{card}(S_h) + \mathrm{card}(T_+)\mathrm{card}(S_n)/2} \\ &= \frac{2\mathrm{card}(S_h)}{2\mathrm{card}(S_h) + \mathrm{card}(S_n)} \leq 2P(y) \end{aligned}$$

□

For a positive integer n, we define $\gamma(n) = n(n/2)\cdots 1$. Notice that $\gamma(n) > n^{(\log n)/2}$.

Theorem 8.4.6 *Suppose r is a run, y is a node at level $l \geq 2$ with q queries from r at or below y, and x is an ancestor of y. Then $P_x(y) \leq q/\gamma(n/4)$ where $n = 2^l$.*

Proof. We prove the theorem by induction on l.

So, fix a run r and a node y at level 2 with q queries from r at or below y. If $q = 0$, y can never be a hit. So, certainly the probability that y is a hit is at most q as desired.

Next, we perform the inductive step. So, assume the theorem holds true for any r and y at level less than l with $l \geq 2$. Then, fix a run r and a node y at level l with q queries from r at or below y. Let $n = 2^l$. We will show that $P(y) \leq \frac{q}{2\gamma(n/4)}$, and then the theorem will follow from Lemma 8.4.5. So, for the remainder of the proof, all probabilities are taken over the choice of a legal oracle at and below y uniformly at random from the set of all those legal oracles at and below y which agree with r.

Now, suppose that q' of the q queries are actually at y. Clearly, if we condition on there being no hits among the children of y, then k_y will be chosen uniformly among all n-bit strings, and hence the probability y is a hit would be $q'/2^n$. If we instead condition on there being exactly c hits among the 2^n children of y, then the probability that y is a hit must be at most $q'/2^{n-c}$. Therefore the probability y is a hit is bounded above by the sum of $q'/2^{n/2}$ and the probability that at least $n/2$ of the children of y are hits.

Now consider any child z of y. Applying the inductive hypothesis with y and z taking the roles of x and y, we know that if r has q_z queries at and below z, then $P_y(z) \leq q_z/\gamma(n/8)$. Therefore the *expected* number of hits among the children of y is at most $(q - q')/\gamma(n/8)$. This means that the probability that at least $n/2$ of the children of y are hits is at most

$$\frac{(q-q')}{\gamma(n/8)n/2} = \frac{(q-q')}{2\gamma(n/4)}$$

Therefore,

$$P(y) \;\le\; \frac{q-q'}{2\gamma(n/4)} + \frac{q'}{2^{n/2}} \;\le\; \frac{q}{2\gamma(n/4)}$$

since $2\gamma(n/4) < 2^{n/2}$ for $n > 4$. □

Corollary 8.4.7 *For any $T(n)$ which is $n^{o(\log n)}$, relative to a random legal oracle O, with probability 1, $\mathcal{R}_O$ is not contained in* $\mathbf{BPTime}\,(T(n))$.

Proof. Fix $T(n)$ which is $n^{o(\log n)}$.

We will show that for any probabilistic TM M, when we pick a random legal oracle O, with probability 1, M^O either fails to run in time $cn^{o(\log n)}$ or it fails to accept $\mathcal{R}_O$ with error probability bounded by 1/3. Then since there are a countable number of probabilistic TMs and the intersection of a countable number of probability 1 events still has probability 1, we conclude that with probability 1, $\mathcal{R}_O$ is not contained in $\mathbf{BPTime}\,(n^{o(\log n)})$.

We prove the corollary by showing that, for large enough n, the probability that M^O runs in time greater than $T(n)$ or has error greater than 1/3 on input 0^n is at least 1/8 for every way of fixing the oracle answers for trees other than the tree rooted at 0^n. The probability is taken over the random choices of the oracle for the tree rooted at 0^n.

Fix arbitrarily a legal behavior for O on all trees other than the one rooted at 0^n. Then consider picking a legal behavior for O for the tree rooted at 0^n uniformly at random and run M^O on input 0^n. We can classify runs of M^O on input 0^n based on the run r that lists the queries the machines asks, and the answers it receives. If we take all probabilities over both the randomness in M and the choice of oracle O, then the probability M^O correctly classifies 0^n in time $T(n)$ is

$$\sum_r Pr[r]Pr[correct \mid r]$$

where $Pr[r]$ is the probability of run r, where $Pr[correct \mid r]$ is the probability the answer is correct given run r, and where r ranges over all runs with at most $T(n)$ queries. Theorem 8.4.6 tells us that if we condition on any run r with fewer than $\frac{1}{12}\gamma(n/4)$ queries, then the probability 0^n is a hit is less than 1/12. This means that the probability the algorithm correctly classifies 0^n, conditioned on any particular run r with fewer than $\frac{1}{12}\gamma(n/4)$ queries, is at most 7/12. Therefore, for n large enough that $T(n)$ is less than $\frac{1}{12}\gamma(n)$, the probability M^O correctly classifies 0^n in time $T(n)$ is at most 7/12. So, for sufficiently large n, when we choose O, then with probability at least 1/8 M^O either fails to run in time $T(n)$ or has success probability less than 2/3 on input 0^n. □

A A QTM is well-formed iff its time evolution is unitary

First, we note that the time evolution operator of a QTM always exists. Note that this is true even if the QTM is not well-formed.

Lemma A.0.8 *If M is a QTM with time evolution operator U, then U has an adjoint operator U^* in the inner-product space of superpositions of the machine M.*

Proof. Let M be a QTM with time evolution operator U. Then the adjoint of U is the operator U' is the operator whose matrix element in any pair of dimensions i, j is the complex conjugate of the matrix element of U in dimensions j, i. The operator U' defined in this fashion still maps all superpositions to other superpositions (= finite linear combinations of configurations), since any particular configuration of M can be reached with non-zero weight from only a finite number of other configurations. It is also easy to see that for any superpositions ϕ, ψ

$$\langle U'\phi|\psi\rangle = \langle\phi|U\psi\rangle$$

as desired. □

We include the proofs of the following standard facts for completeness. We will use them in the proof of the theorem below.

Fact A.0.9 *If U is a linear operator on an inner-product space V and U^* exists, then U preserves norm iff $U^*U = I$.*

Proof. For any $x \in V$, the square of the norm of Ux is $(Ux, Ux) = (x, U^*Ux)$. It clearly follows that if $U^*U = I$ then U preserves norm. For the converse, let $B = U^*U - I$. Since U preserves norm, for every $x \in V$, $(x, U^*Ux) = (x, x)$. Therefore for every $x \in V$, $(x, Bx) = 0$. It follows that $B = 0$ and therefore $U^*U = I$. □

This further implies the following useful fact:

Fact A.0.10 *Suppose U is an linear operator in an inner-product space V and U^* exists. Then*

$$\forall x \in V \quad \|Ux\| = \|x\| \qquad \leftrightarrow \qquad \forall x, y \in V \quad (Ux, Uy) = (x, y)$$

Proof. Since $\|Ux\| = \|x\| \leftrightarrow (Ux, Ux) = (x, x)$, one direction follows by substituting $x = y$. For the other direction, if U preserves norm then by fact A.0.9, $U^*U = I$. Therefore $(Ux, Uy) = (x, U^*Uy) = (x, y)$. □

We need to establish one one additional fact about norm preserving operators, before we can prove our theorem:

Fact A.0.11 *Let V be a countable inner-product space, and let $\{|i\rangle\}_{i\in I}$ be an orthonormal basis for V. If U is a norm preserving linear operator on V and U^* exists, then $\forall i \in I$ $\|U^*|i\rangle\| \leq 1$. Moreover, if $\forall i \in I$ $\|U^*|i\rangle\| = 1$ then U is unitary.*

Proof. Since U preserves norm, $\|UU^*|i\rangle\| = \|U^*|i\rangle\|$. But the projection of $UU^*|i\rangle$ on $|i\rangle$ has norm $|\langle i|UU^*|i\rangle| = \|U^*|i\rangle\|^2$. Therefore $\|U^*|i\rangle\| \geq \|U^*|i\rangle\|^2$, and therefore $\|U^*|i\rangle\| \leq 1$.

Moreover, if $\|U^*|i\rangle\| = 1$, then since U is norm preserving, $\|UU^*|i\rangle\| = 1$. On the other hand the projection of $UU^*|i\rangle$ on $|i\rangle$ has norm $\|U^*|i\rangle\|^2 = 1$. It follows that for $j \neq i$, the projection of $UU^*|i\rangle$ on $|j\rangle$ must have norm 0. Thus $|\langle j|UU^*|i\rangle| = 0$. It follows that $UU^* = I$. □

If V is finite dimensional, then $U^*U = I$ implies $UU^* = I$, and therefore an operator is norm preserving if and only if it is unitary. However, if $\mathcal{H}$ is infinite dimensional, then $U^*U = I$ does not imply $UU^* =$

I.[6]. Nevertheless, the time evolution operators of QTMs have a special structure, and in fact these two conditions are equivalent for the time evolution operator of a QTM.

Theorem A.0.12 *A QTM is well-formed iff its time evolution operator is unitary.*

Proof. Let U be the norm preserving time evolution operator of a well-formed QTM $M = (\Sigma, Q, \delta)$. Consider the standard orthonormal basis for the superpositions of M given by the set of vectors $|c\rangle$, where c ranges over all configurations of M (as always, $|c\rangle$ is the superposition with amplitude 1 for configuration c and 0 elsewhere). We may express the action of U with respect to this standard basis by a countable dimensional matrix whose c, c'^{th} entry $u_{c,c'} = \langle c'|U|c\rangle$. This matrix has some special properties. First, each row and column of the matrix has only a finite number of non-zero entries. Secondly, there are only finitely many different types of rows, where two rows are of the same type if their entries are just permutations of each other. We shall show that each row of the matrix has norm 1, and therefore by the Fact A.0.11 above U is unitary. To do so we will identify a set of n columns of the matrix (for arbitrarily large n) and restrict attention to the finite matrix consisting of all the chosen rows and all columns with non-zero entries in these rows. Let this matrix be the $m \times n$ matrix B. By construction, B satisfies two properties: 1) it is almost square; $m/n \leq 1 + \epsilon$ for arbitrarily small ϵ. 2) There is a constant a such that each distinct row type of the infinite matrix occurs at least m/a times among the rows of B.

Now the sum of the squared norms of the rows of B is equal to the sum of the squared norms of the columns. The latter quantity is just n (since the columns of the infinite matrix are orthonormal by Fact A.0.9 above). If we assume that some row of the infinite matrix has norm $1 - \delta$ for $\delta > 0$, then we can choose n sufficiently large and ϵ sufficiently small so that the sum of the squared norms of the rows is at most $m(1-1/a) + m/a(1-\delta) \leq m - m\delta/a \leq n + n\epsilon - n\delta/a < n$. This gives the required contradiction, and therefore all rows of the infinite matrix have norm 1 and therefore by the Fact A.0.11 above U is unitary.

To construct the finite matrix B let $k > 2$ and fix some contiguous set of k cells on the tape. Consider the set of all configurations S such that the tape head is located within these k cells and such that the tape is blank outside of these k cells. It is easy to see that the number of such configurations $n = k\ \mathrm{card}(\Sigma)^k\ \mathrm{card}(Q)$. The columns indexed by configurations in S are used to define the finite matrix B referred to above. The non-zero entries in these columns are restricted to rows indexed by configurations in S together with rows indexed by configurations where the tape head is in the cell immediately to the left or right of the k special cells, and such that the tape is blank outside of these $k+1$ cells. The number of these additional configurations over and above S is at most $2\ \mathrm{card}(\Sigma)^{k+1}\ \mathrm{card}(Q)$. Therefore $m = n(1 + 2/k\mathrm{card}(\Sigma))$. For any $\epsilon > 0$, we can choose k large enough such that $m \leq n(1+\epsilon)$.

Recall that a row of the infinite matrix (corresponding to the operator U) is indexed by configurations. We say that a configuration c is of type $q, \sigma_1, \sigma_2, \sigma_3$ if c is in state $q \in Q$ and the three adjacent tape cells centered about the tape head contain the three symbols $\sigma_1, \sigma_2, \sigma_3 \in \Sigma$. The entries of a row indexed by c must be a permutation of the entries of a row indexed by any configuration c' of the same type as c. This is because a transition of the QTM M depends only on the state of M and the symbol under the tape head, and since the tape head moves to an adjacent cell during a transition. Moreover, any configuration d that yields configuration c with non-zero amplitude as a result of a single step must have tape contents identical to those of c in all but these three cells. It follows that there are only a finite number of such configurations d, and the row indexed by c can have only a finite number of non-zero entries. A similar argument shows that each column has only a finite number of non-zero entries.

Now for given $q, \sigma_1, \sigma_2, \sigma_3$ consider the rows of the finite matrix B indexed by configurations of type $q, \sigma_1, \sigma_2, \sigma_3$, and such that the tape head is located at one of the $k-2$ non-border cells. Then

[6] Consider for example the space of finite complex linear combinations of positive integers and the linear operator which maps $|i\rangle$ to $|i+1\rangle$.

$|T| = (k-2)\mathrm{card}(\Sigma)^{k-3}$. Each row of B indexed by a configuration $c \in T$ has the property that if d is any configuration that yields c with non-zero amplitude in a single step, then $d \in S$. Therefore the row indexed by c in the finite matrix B has the same non-zero entries as the row indexed by c in the infinite matrix. Therefore it makes sense to say that row c of B is of the same type as row c of the infinite matrix. Finally, the rows of each type constitute a fraction at least $|T|/m$ of all rows of B. Substituting the bounds from above, we get that this fraction is at least $\frac{(k-2)\mathrm{card}(\Sigma)^{k-3}}{k\ \mathrm{card}(\Sigma)^k\ \mathrm{card}(Q)(1+2/k\mathrm{card}(\Sigma))}$. Since $k \geq 4$, this is at least $\frac{1}{a}$ for constant $a = 2\mathrm{card}(\Sigma)^3\mathrm{card}(Q)(1 + 1/2\mathrm{card}(\Sigma))$. This establishes all the properties of the matrix B used in the proof above. □

B Reversible TMs are as powerful as deterministic TMs

In this appendix, we prove the Synchronization Theorem of §4.1.

We begin with a few simple facts about reversible TMs. We give necessary and sufficient conditions for a deterministic TM to be reversible, and we show that, just as for QTMs, a partially defined reversible TM can always be completed to give a well-formed reversible TM. We also give, as an aside, an easy proof that reversible TMs can efficiently simulate reversible, generalized TMs.

Theorem B.0.13 *A TM or generalized TM M is reversible iff the following two conditions both hold*

1. *Each state of M can be entered while moving in only one direction. In other words, if $\delta(p_1, \sigma_1) = (\tau_1, q, d_1)$ and $\delta(p_2, \sigma_2) = (\tau_2, q, d_2)$ then $d_1 = d_2$.*
2. *The transition function δ is one-to-one when direction is ignored.*

Proof. First we show that these two conditions imply reversibility.

Suppose $M = (\Sigma, Q, \delta)$ is a TM or generalized TM satisfying these two conditions. Then, the following procedure lets us take any configuration of M and compute its predecessor if it has one. First, since each state can be entered while moving in only one direction, the state of the configuration tells us in which cell the tape head must have been in the previous configuration. Looking at this cell, we can see what tape symbol was written in the last step. Then, since δ is one-to-one we know the update rule, if any, that was used on the previous step, allowing us to reconstruct the previous configuration.

Next, we show that the first property is necessary for reversibility. So, for example, consider a TM or generalized TM $M = (\Sigma, Q, \delta)$ such that $\delta(p_1, \sigma_1) = (\tau_1, q, L)$ and $\delta(p_2, \sigma_2) = (\tau_2, q, R)$. Then, we can easily construct two configurations which lead to the same next configuration: Let c_1 be any configuration where the machine is in state p_1 reading a σ_1 and where the symbol two cells to the left of the tape head is a τ_2, and let c_2 be identical to c_1 except that the σ_1 and τ_2 are changed to τ_1 and σ_2, the machine is in state p_2, and the tape head is two cells further left. Therefore M is not reversible. Since similar arguments apply for each pair of distinct directions, the first condition in the theorem must be necessary for reversibility.

Finally, we show that the second condition is also necessary for reversibility. Suppose that $M = (\Sigma, Q, \delta)$ is a TM or generalized TM with $\delta(p_1, \sigma_1) = \delta(p_2, \sigma_2)$. Then, any pair of configurations which differ only in the state and symbol under the tape head, where one has (p_1, σ_1) and the other (p_2, σ_2), lead to the same next configuration, and again M is not reversible. □

Corollary B.0.14 *If M is a reversible TM then every configuration of M has exactly one predecessor.*

Proof. Let $M = (\Sigma, Q, \delta)$ be a reversible TM. By the definition of reversibility, each configuration of M has at most one predecessor.

So, let c be a configuration of M in state q. Theorem B.0.13 tells us that M can enter state q while moving its tape head in only one direction d_q. Since Theorem B.0.13 tells us that, ignoring direction, δ is one-to-one, taking the inverse of δ on the state q and the symbol in direction $\bar{d}_q$ tells us how to transform c into its predecessor. □

Corollary B.0.15 *If δ is a partial function from $Q \times \Sigma$ to $\Sigma \times Q \times \{L, R\}$ satisfying the two conditions of Theorem B.0.13 then δ can be extended to a total function that still satisfies Theorem B.0.13.*

Proof. Suppose δ is a partial function from $Q \times \Sigma$ to $\Sigma \times Q \times \{L, R\}$ that satisfies the properties of Theorem B.0.13. Then, for each $q \in Q$ let d_q be the one direction, if any, in which q can be entered, and let d_q be (arbitrarily) L otherwise. Then we can fill in undefined values of δ with as yet unused triples of the form (τ, q, d_q) so as to maintain the conditions of Theorem B.0.13. Since there are card(Σ) card(Q) such triples there will be exactly enough to fully define δ. □

Theorem B.0.16 *If M is a generalized reversible TM, then there is a reversible TM M' that simulates M with slowdown at most 2.*

Proof. The idea is to replace any transition that has the tape head stand still with two transitions. The first updates the tape and moves to the right, remembering which state it should enter. The second steps back to the left and enters the desired state.

So, if $M = (\Sigma, Q, \delta)$ is a generalized reversible TM then we let M' be identical to M except that for each state q with a transition of the form $\delta(p, \sigma) = (\tau, q, N)$ we add a new state q' and we also add a new transition rule $\delta(q', \sigma) = \sigma, q, L$ for each $\sigma \in \Sigma$. Finally, we replace each transition $\delta(p, \sigma) = (\tau, q, N)$ with the transition $\delta(p, \sigma) = (\tau, q', R)$. Clearly M' simulates M with slowdown by a factor of at most 2.

To complete the proof, we need to show that M' is also reversible.

So, consider a configuration c of M'. We need to show that c has at most one predecessor. If c is in state $q \in Q$ and M enters q moving left or right, then the transitions into q in M' are identical to those in M and therefore since M is reversible, c has at most one predecessor in M'. Similarly, if c is in one of the new states q' then the transitions into q' in M' are exactly the same as those into q in M, except that the tape head moves right instead of staying still. So, again the reversibility of M implies that c has at most one predecessor. Finally, suppose c is in state $q \in Q$ where M enters q while standing still. Then, Theorem B.0.13 tells us that all transitions in M that enter q have direction N. Therefore, all of them have been removed, and the only transitions entering q in M' are the new ones of the form $\delta(q', \sigma) = \sigma, q, L$. Again, this means that c can have only one predecessor. □

We will prove the Synchronization Theorem in the following way. Using ideas from the constructions of Bennett [7] and Morita et.al. [33], we will show that given any deterministic TM M there is a reversible multi-track TM which on input x produces output $x; M(x)$, and whose running time depends only on the sequence of head movements of M on input x. Then, since any deterministic computation can be simulated efficiently by an "oblivious" machine whose head movements depend only on the length of its input, we will be able to construct the desired "synchronized" reversible TM.

The idea of Bennett's simulation is to run the target machine, keeping a history to maintain reversibility. Then the output can be copied and the simulation run backwards so that the history is exactly erased while the input is recovered. Since the target tape head moves back and forth, while the history steadily grows, Bennett uses a multi-tape TM for the simulation.

The idea of Morita et.al.'s simulation is to use a simulation tape with several tracks, some of which are used to simulate the tape of the desired machine, and some of which are used to keep the history. Provided that the machine can move reversibly between the current head position of the target machine and the end of the history, it can carry out Bennett's simulation with a quadratic slowdown. Morita et.al. work with TMs with one-way infinite tapes so that the simulating machine can move between the simulation and the history by moving left to the end of the tape and then searching back towards the right. In our simulation, we write down history for every step the target machine takes, rather than just the non-reversible steps. This means that the end of the history will always be further right than the simulation, allowing us to work with two-way infinite tapes. Also, we use reversible TMs that can only move their head in the directions $\{L, R\}$ rather than the generalized reversible TMs used by Bennett and Morita et.al.

Definition B.0.17 *A deterministic TM is* oblivious *if its running time and the position of its tape head at each time step depend only on the length of its input.*

In carrying out our single tape Bennett constructions we will find it useful to first build simple reversible TMs to copy a string from one track to another and to exchange the strings on two tracks. We will copy strings delimited by blanks in the single tape Bennett construction, but will copy strings with other delimiters in a later section.

Lemma B.0.18 *For any alphabet Σ, there is a normal form, reversible TM M with alphabet $\Sigma \times \Sigma$ with the following property. When run on input $x; y$ M runs for $2\max(|x|, |y|) + 4$ steps, returns the tape head to the start cell, and outputs $y; x$.*

Proof. We let M have alphabet $\Sigma \times \Sigma$, state set $\{q_0, q_1, q_2, q_3, q_f\}$, and transition function defined by

	$(\#, \#)$	other (σ_1, σ_2)
q_0	$(\#, \#), q_1, L$	$(\sigma_1, \sigma_2), q_1, L$
q_1	$(\#, \#), q_2, R$	
q_2	$(\#, \#), q_3, L$	$(\sigma_2, \sigma_1), q_2, R$
q_3	$(\#, \#), q_f, R$	$(\sigma_1, \sigma_2), q_3, L$
q_f	$(\#, \#), q_0, R$	$(\sigma_1, \sigma_2), q_0, R$

Since each state in M can be entered in only one direction, and its transition function is one-to-one, M is reversible. Also, it can be verified that M performs the desired computation in the stated number of steps. $\square$

Lemma B.0.19 *For any alphabet Σ, there is a normal form, reversible TM M with alphabet $\Sigma \times \Sigma$ with the following property. When run on input x, M outputs $x; x$, and when run on input $x; x$ it outputs x. In either case, M runs for $2|x| + 4$ steps and leaves the tape head back in the start cell.*

Proof. We let M have alphabet $\Sigma \times \Sigma$, state set $\{q_0, q_1, q_2, q_3, q_f\}$, and transition function defined by the following where each transition is duplicated for each non-blank $\sigma \in \Sigma$

	$(\#,\#)$	$(\sigma,\#)$	(σ,σ)
q_0	$(\#,\#), q_1, L$	$(\sigma,\#), q_1, L$	$(\sigma,\sigma), q_1, L$
q_1	$(\#,\#), q_2, R$		
q_2	$(\#,\#), q_3, L$	$(\sigma,\sigma), q_2, R$	$(\sigma,\#), q_2, R$
q_3	$(\#,\#), q_f, R$	$(\sigma,\#), q_3, L$	$(\sigma,\sigma), q_3, L$
q_f	$(\#,\#), q_0, R$	$(\sigma,\#), q_0, R$	$(\sigma,\sigma), q_0, R$

Since each state in M can be entered in only one direction, and its transition function is one-to-one, M can be extended to a reversible TM. Also, it can be verified that M performs the desired computation in the stated number of steps. □

Theorem B.0.20 *Let M be an oblivious deterministic TM which on any input x produces output $M(x)$, with no embedded blanks, with its tape head back in the start cell, and with $M(x)$ beginning in the start cell. Then there is a reversible TM M' which on input x produces output $x; M(x)$ and on input $x; M(x)$, produces output x. In both cases, M' halts with its tape head back in the start cell, and takes time which depends only on the lengths of x and $M(x)$, and which is bounded by a quadratic polynomial in the running time of M on input x.*

Proof. Let $M = (\Sigma, Q, \delta)$ be an oblivious deterministic TM as stated in the theorem and let q_0, q_f be the initial and final states of M.

The simulation will run in three stages.

1. M' will simulate M maintaining a history to make the simulation reversible.
2. M' will copy its first track, as shown in Lemma B.0.19.
3. M' runs the reverse of the simulation of M erasing the history while restoring the input.

We will construct a normal form reversible TM for each of the three stages, and then dovetail them together.

Each of our machines will be a four-track TM with the same alphabet. The first track, with alphabet $\Sigma_1 = \Sigma$, will be used to simulate the tape of M. The second track, with alphabet $\Sigma_2 = \{\#, 1\}$, will be used to store a single 1 locating the tape head of M. The third track, with alphabet $\Sigma_3 = \{\#, \$\} \cup (Q \times \Sigma)$, will be used to write down a list of the transitions taken by M, starting with the marker \$. This \$ will help us find the start cell when we enter and leave the copying phase. The fourth track, with alphabet $\Sigma_4 = \Sigma$, will be used to write the output of M.

In describing the first machine, we will give a partial list of transitions obeying the conditions of Theorem B.0.13, and then appeal to Corollary B.0.15. Our machines will usually only be reading and writing one track at a time. So, for convenience we will list a transition with one or more occurrences of the symbols of the form v_i and v_i' to stand for all possible transitions with v_i replaced by a symbol from the appropriate Σ_i other than the special marker \$, and with v_i' replaced by a symbol from Σ_i other than \$ and $\#$.

The first phase of the simulation will be handled by the machine M_1 with state set given by the union of sets Q, $Q \times Q \times \Sigma \times [1, 4]$, $Q \times [5, 7]$, and $\{q_a, q_b\}$. Its start state will be q_a and the final state q_f.

The transitions of M_1 are defined as follows. First, we mark the position of the tape head of M in the start cell, mark the start of the history in cell 1, and enter state q_0.

$$\begin{array}{lclll} q_a, & (v_1, \#, \#, v_4) & \rightarrow & (v_1, 1, \#, v_4), & q_b, & R \\ q_b, & (v_1, \#, \#, v_4) & \rightarrow & (v_1, \#, \$, v_4), & q_0, & R \end{array}$$

Then, for each pair p, σ with $p \neq q_f$ and with transition $\delta(p, \sigma) = (\tau, q, d)$ in M we include transitions to go from state p to state q updating the simulated tape of M while adding (p, σ) to the end of the history. We first carry out the update of the simulated tape, remembering the transition taken, and then move to the end of the history to deposit the information on the transition. If we are in the middle of the history, we reach the end of the history by walking right until we hit a blank. However, if we are to the left of the history, then we must first walk right over blanks until we reach the start of the history.

$$\begin{array}{cccccc} p, & (\sigma, 1, v_3, v_4) & \rightarrow & (\tau, \#, v_3, v_4), & (q, p, \sigma, 1), & d \\ (q, p, \sigma, 1), & (v_1, \#, \$, v_4) & \rightarrow & (v_1, 1, \$, v_4), & (q, p, \sigma, 3), & R \\ (q, p, \sigma, 1), & (v_1, \#, v_3', v_4) & \rightarrow & (v_1, 1, v_3', v_4), & (q, p, \sigma, 3), & R \\ (q, p, \sigma, 1), & (v_1, \#, \#, v_4) & \rightarrow & (v_1, 1, \#, v_4), & (q, p, \sigma, 2), & R \\ (q, p, \sigma, 2), & (v_1, \#, \#, v_4) & \rightarrow & (v_1, \#, \#, v_4), & (q, p, \sigma, 2), & R \\ (q, p, \sigma, 2), & (v_1, \#, \$, v_4) & \rightarrow & (v_1, \#, \$, v_4), & (q, p, \sigma, 3), & R \\ (q, p, \sigma, 3), & (v_1, \#, v_3', v_4) & \rightarrow & (v_1, \#, v_3', v_4), & (q, p, \sigma, 3), & R \\ (q, p, \sigma, 3), & (v_1, \#, \#, v_4) & \rightarrow & (v_1, \#, (p, \sigma), v_4), & (q, 4), & R \end{array}$$

When the machine reaches state $(q, 4)$ it is standing on the first blank after the end of the history. So, for each state $q \in Q$, we include transitions to move from the end of the history back left to the head position of M. We enter our walk left state $(q, 5)$ while writing a $\#$ on the history tape. So, to maintain reversibility, we must enter a second left-walking state $(q, 6)$ to look for the tape head marker past the left end of the history. When we reach the tape head position, we step left and right entering state q.

$$\begin{array}{cccccc} (q, 4), & (v_1, \#, \#, v_4) & \rightarrow & (v_1, \#, \#, v_4), & (q, 5), & L \\ (q, 5), & (v_1, \#, v_3', v_4) & \rightarrow & (v_1, \#, v_3', v_4), & (q, 5), & L \\ (q, 5), & (v_1, \#, \$, v_4) & \rightarrow & (v_1, \#, \$, v_4), & (q, 6), & L \\ (q, 5), & (v_1, 1, v_3', v_4) & \rightarrow & (v_1, 1, v_3', v_4), & (q, 7), & L \\ (q, 5), & (v_1, 1, \$, v_4) & \rightarrow & (v_1, 1, \$, v_4), & (q, 7), & L \\ (q, 6), & (v_1, \#, \#, v_4) & \rightarrow & (v_1, \#, \#, v_4), & (q, 6), & L \\ (q, 6), & (v_1, 1, \#, v_4) & \rightarrow & (v_1, 1, \#, v_4), & (q, 7), & L \\ (q, 7), & (v_1, v_2, v_3, v_4) & \rightarrow & (v_1, v_2, v_3, v_4), & q, & R \end{array}$$

Finally, we put M_1 in normal form by including the transition

$$q_f, \quad \sigma \quad \rightarrow \quad \sigma, \quad q_a, \quad R$$

for each σ in the simulation alphabet.

It can be verified that each state can be entered in only one direction using the above transitions, and relying on the fact that M is in normal form, and we don't simulate it's transitions from q_f back to q_0, it can also be verified that the partial transition function described is one-to-one. Therefore, Corollary B.0.15 says that we can extend these transitions to give a reversible TM M_1. Notice also that the operation of M_1 is independent of the contents of the fourth track, and that it leaves these contents unaltered.

For the second and third machines, we simply use the copying machine constructed in Lemma B.0.19 above, and the reverse of M_1 constructed using Lemma 4.2.9 on page 20. Since M_1 operated independently of the fourth track, so will its reversal. Therefore, dovetailing these three TMs gives a reversible TM M', which on input x produces output $x;\epsilon;\epsilon;M(x)$ with its tape head back in the start cell, and on input $x;\epsilon;\epsilon;M(x)$ produces output x.

Notice that the time required by M_1 to simulate a step of M is bounded by a polynomial in the running time of M, and depends only on the current head position of M, the direction M moves in the step, and how many steps have already been carried out. Therefore, the running time of the first phase of M' depends only on the series of head movements of M. In fact, since M is oblivious, this running time depends only the length of x. The same is true of the third phase of M', since the reversal of M_1 takes exactly two extra time steps. Finally, the running time of the copying machine depends only on the length of $M(x)$. Therefore, the running time of M' on input x depends only on the lengths of x and $M(x)$ and is bounded by a quadratic polynomial in the running time of M on x. □

Theorem B.0.21 *Let M_1 be an oblivious deterministic TM which on any input x produces output $M_1(x)$, with no embedded blanks, with its tape head back in the start cell, with $M_1(x)$ beginning in the start cell, and such that the length of $M_1(x)$ depends only on the length of x. Let M_2 be an oblivious deterministic TM with the same alphabet as M_1 which on any input $M_1(x)$ produces output x, with its tape head back in the start cell, and with x beginning in the start cell. Then there is a reversible TM M' which on input x produces output $M_1(x)$. Moreover, M' on input x halts with its tape head back in the start cell, and takes time which depends only on the length of x and which is bounded by a polynomial in the running time of M_1 on input x and the running time of M_2 on $M_1(x)$.*

Proof. Let M_1 and M_2 be as stated in the theorem with the common alphabet Σ.

Then the idea to construct the desired M' is first to run M_1 to compute $x;M(x)$, then to run an exchange routine to produce $M(x);x$ and finally to run M_2 to erase the string x, where each of the three phases starts and ends with the tape head in the start cell. Using the construction in Theorem B.0.20, we build normal form, reversible TMs to accomplish the first and third phases in times which depend only on the lengths of x and $M_1(x)$ and bounded by a polynomial in the running times of M_1 on input x and M_2 on input $M_1(x)$. In Lemma B.0.18 on page 57 we have already constructed a reversible TM that performs the exchange in time depending only on the lengths of the two strings. Dovetailing these three machines gives the desired M. □

Since any function computable in deterministic polynomial time can be computed in polynomial time by an oblivious generalized deterministic TM, Theorems B.0.20 and B.0.21 together with Theorem 4.1.2 give us the following.

Theorem 4.1.3 (Synchronization Theorem) *If f is a function mapping strings to strings which can be computed in deterministic polynomial time and such that the length of $f(x)$ depends only on the length of x, then there is a polynomial time, stationary, normal form QTM which given input x, produces output $x;f(x)$, and whose running time depends only on the length of x.*

If f is a function from strings to strings that such that both f and f^{-1} can be computed in deterministic polynomial time, and such that the length of $f(x)$ depends only on the length of x, then there is a polynomial time, stationary, normal form QTM which given input x, produces output $f(x)$, and whose running time depends only on the length of x.

C A reversible looping TM

We prove here the Looping Lemma from §4.2.

Lemma 4.2.10 (Looping Lemma) *There is a stationary, normal form, reversible TM M and a constant c with the following properties. On input any positive integer k written in binary, M runs for time $O(k \log^c k)$ and halts with its tape unchanged. Moreover, M has a special state q^* such that on input k, M visits state q^* exactly k times, each time with its tape head back in the start cell.*

Proof. As mentioned above in §4.2, the difficulty is to construct a loop with a reversible entrance and exit. We accomplish this as follows. Using the Synchronization Theorem, we can build three-track stationary, normal form reversible TM $M_1 = (\Sigma, Q, \delta)$ running in time polynomial in $\log k$ that on input $b; x; k$ where $b \in \{0, 1\}$ outputs $b'; x + 1; k$ where b' is the opposite of b if $x = 0$ or $k - 1$ (but not both) and $b' = b$ otherwise. Calling the initial and final states of this machine q_0, q_f, we construct a reversible M_2 that loops on machine M_1 as follows. We will give M_2 new initial and final states q_a, q_z, and ensure that it has the following three properties.

1. Started in state q_a with a 0 on the first track, M_2 steps left and back right, changing the 0 to a 1, and entering state q_0.
2. When in state q_f with a 0 on the first track, M_2 steps left and back right into state q_0.
3. When in state q_f with a 1 on the first track, M_2 steps left and back right, changing the 1 to a 0, and halts.

So, on input $0; 0; k$ M_2 will step left and right into state q_0, changing the tape contents to $1; 0; k$. Then machine M_1 will run for the first time changing $1; 0; k$ to $0; 1; k$ and halting in state q_f. Whenever M_2 is in state q_f with a 0 on the first track, it reenters q_0 to run M_2 again. So, machine M_2 will run $k - 1$ more times until it finally produces $1; k; k$. At that point, M_2 changes the tape contents to $0; k; k$ and halts. So on input $0; 0; k$, M_2 visits state q_f exactly k times, each time with its tape head back in the start cell. This means we can construct the desired M by identifying q_f as q^*, and dovetailing M_2 before and after with reversible TMs, constructed using the Synchronization Theorem, to transform k to $0; 0; k$ and $0; k; k$ back to k.

We complete the proof by constructing a normal form, reversible M_2 that satisfies the three properties above. We give M_2 the same alphabet as M_1 and additional states q_a, q_b, q_y, q_z. The transition function for M_2 is the same as that of M_1 for states in $Q - q_f$ and otherwise depends only on the first track (leaving the others unchanged) and is given by the following table

	#	0	1
q_a		$(1, q_b, L)$	
q_b	$(\#, q_0, R)$		
q_f		$(0, q_b, L)$	$(0, q_y, L)$
q_y	$(\#, q_z, R)$		
q_z	$(\#, q_a, R)$	$(0, q_a, R)$	$(1, q_a, R)$

It is easy to see that M_2 is is normal form and satisfies the three properties stated above. Moreover, since M_1 is reversible and obeys the two conditions of Theorem B.0.13, it can be verified that the transition function of M_2 also obeys the two conditions of Theorem B.0.13. Therefore, according to Theorem B.0.15, the transition function of M_2 can be completed giving a reversible TM. □

Acknowledgements

This paper has greatly benefitted from the careful reading and many useful comments of Richard Jozsa and Bob Solovay. We wish to thank them as well as Gilles Brassard, Charles Bennett, Noam Nisan and Dan Simon.

References

[1] Adleman, L., DeMarrais, J., and Huang, M., *Quantum computability*, manuscript, 1994.

[2] Arora S., Impagliazzo, R., Vazirani, U., *On the Role of the Cook-Levin Theorem in Complexity Theory*, manuscript, 1993.

[3] Arora S., Lund C., Motwani R., Sudan M. and Szegedy M., *Proof verification and intractability of approximation problems*, Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992.

[4] Babai, L., and Moran, S., *Arthur–Merlin games: a randomized proof system, and a hierarchy of complexity classes*, Journal of Computer and System Sciences, 36(1988), pp. 254–276.

[5] Barenco, A., Bennett, C., Cleve, R., DiVincenzo, D., Margolus, N., Shor, P., Sleator, T., Smolin, J., and Weinfurter, H., *Elementary gates for quantum computation*, manuscript, 1995.

[6] Benioff, P., *Quantum Hamiltonian models of Turing machines* Journal of Statistical Physics, 29(1982), pp. 515–546.

[7] Bennett, C. H., *Logical reversibility of computation*, IBM J. Res. Develop., 17(1973), pp. 525–532.

[8] Bennett, C. H., *Time/space tradeoffs for reversible computation*, SIAM J. Comput., 18(1989), pp. 766–776.

[9] Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U., *Strengths and weaknesses of quantum computing*, SIAM J. Comput., to appear.

[10] Bernstein, E., *Quantum complexity theory*, PhD dissertation, U.C. Berkeley, to appear.

[11] Bernstein, E. and Vazirani, U., *Quantum complexity theory*, Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 11–20.

[12] Berthiaume, A., and Brassard, G., *The quantum challenge to structural complexity theory*, Proceedings of 7th IEEE Conference on Structure in Complexity Theory, 1992, pt. 132–137.

[13] Berthiaume, A., and Brassard, G., *Oracle quantum computing*, Journal of Modern Optics, 41(1994), pp. 2521–2535.

[14] Berthiaume, A., Deutsch, D., and Jozsa, R., *The stabilisation of quantum computation*, Proceedings of the Workshop on Physics and Computation, 1994, p. 60.

[15] Bshouty, N. and Jackson, J., *Learning DNF over uniform distribution using a quantum example oracle*, Proceedings of 8th Annual ACM Conference on Computational Learning Theory, 1995, pp. 118–127.

[16] Calderbank, A.R. and Shor, P., *Good quantum error correcting codes exist*, Phys. Rev. A, to appear.

[17] Chirac and Zoller, *Quantum computation using trapped cold ions*, manuscript, 1995.

[18] Chuang, I., LaFlamme, R., Shor, P., and Zurek, W., *Quantum computers, factoring and decoherence*, manuscript, 1995.

[19] Cohen-Tannoudji, C., Diu, B., LaLoe, F., *Quantum mechanics*, 1977, pp. 108–181.

[20] Deutsch, D., *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. R. Soc. Lond., A400(1985), pp. 97–117.

[21] Deutsch, D., *Quantum computational networks*, Proc. R. Soc. Lond., A425(1989), pp. 73–90.

[22] Deutsch, D. and Jozsa, R., *Rapid solution of problems by quantum computation*, Proc. R. Soc. Lond., A439(1992), pp. 553–558.

[23] DiVincenzo, D., *Two-bit gates are universal for quantum computation*, Phys. Rev. A, 51(1995), pp. 1015-1022.

[24] Dürr, C., Santha, M., Thanh, *A decision procedure for well-formed linear quantum cellular automata*, RAC workshop, Berkeley, 1995.

[25] Feynman, R., *Simulating physics with computers*, International Journal of Theoretical Physics, 21(1982), pp. 467–488.

[26] Feynman, R., *Quantum mechanical computers*, Foundations of Physics, 16(1986), pp. 507–531. (Originally appeared in Optics News, February 1985.)

[27] Fredkin, E., and Toffoli, T., *Conservative Logic*, Internat. J. Theoret. Phys., 21(1982), p. 219.

[28] Grover, L., *Searching for a needle in a haystack – a fast quantum mechanical algorithm*, manuscript, 1995.

[29] Landauer, R., *Is quantum mechanics useful?*, Proc. R. Soc. Lond., to appear.

[30] Lipton, R., Personal communication, 1994.

[31] Lloyd, S., *A potentially realizable quantum computer*, Science, 261(1993), pp. 1569–1571.

[32] Machta, J., *Phase information in quantum oracle computing*, Physics Department, University of Massachusetts at Amherst, manuscript, 1996.

[33] Morita, K., Shirasaki, A., and Gono, Y., *A 1-tape 2-symbol reversible Turing Machine*, IEEE Transactions of the IEICE, E72(1989), pp. 223–228.

[34] von Neumann J., *Various Techniques Used in Connection with Random Digits*, Notes by G. E. Forsythe, National Bureau of Standards, Applied Math Series, 12(1951), pp. 36–38. Reprinted in von Neumann's Collected Works, 5(1963), Pergamon Press, pp. 768–770.

[35] Palma, G., Suominen, K., and Ekert, A., *Quantum computers and dissipation*, manuscript, 1995.

[36] A. Shamir., *IP=PSPACE*, Proceedings of the 22nd ACM Symposium on the Theory of Computing, 1990, pp. 11–15.

[37] Shor, P., *Algorithms for quantum computation: Discrete log and factoring*, Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1994.

[38] Shor, P., *Fault-tolerant quantum computation*, Proceedings of the 37th Annual IEEE Symposium on the Foundations of Computer Science, 1996.

[39] Simon, D., *On the power of quantum computation*, Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1994.

[40] Solovay, R., and Yao, A., manuscript, 1996.

[41] Toffoli, T., *Bicontinuous extensions of invertible combinatorial functions*, Mathematical Systems Theory, 14(1981), pp. 13–23.

[42] Unruh, W., *Maintaining coherence in quantum computers*, Phys. Rev. A, 51(1995), p. 992.

[43] Valiant, L., Personal communication, 1992.

[44] Vazirani, U. and Vazirani, V., *Random polynomial time is equal to semi-random polynomial time*, Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science, 1985.

[45] Watrous, J., *On one dimensional quantum cellular automata*, Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science, 1995.

[46] Yao, A., *Quantum circuit complexity*, Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, 1993, pp. 352–361.

[47] Zuckerman, D., *Weak Random Sources*, PhD dissertation, U.C. Berkeley, 1990.