Quantum Complexity Theory

(Preliminary Abstract)

Ethan Bernstein[*]
Umesh Vazirani[†]
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720

1 Introduction

Just as the theory of computability had 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). For example, computers that can operate on arbitrary length words in unit time, or that can exactly compute with infinite precision real numbers are unreasonable models - since it seems clear that they cannot be physically implemented. It had been argued that the Turing Machine model (or 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 $NP \neq BPP$, this would seem to put a wide range of important computational problems (the NP-hard problems) well beyond the capability of computers.

However, the Turing Machine is an inadequate model for 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? The first indication that such a device might potentially be more powerful than a probabilistic Turing Machine appeared in a paper by Feynman [Fe82] about a decade ago. In that paper, Feynman pointed out a very curious problem: it appears to be impossible to simulate a general quantum physical system on a probabilistic TM without an exponential slowdown. The difficulty with the simulation has nothing to do with the problem of simulating a continuous system with a discrete one - we may assume that the quantum physical system to be simulated is discrete, some kind of a quantum cellular automaton. 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 [De85]. This model may be thought of as a quantum physical analogue of a probabilistic TM - it has an infinite tape and a finite state control, and the actions of the machine are local and completely specified by this finite state control. In full generality, on any given input a quantum TM produces a random sample from a probability distribution.

Are general quantum Turing Machines physically realizable? (See [De89] for a dicussion of this issue.) We must not have to devise a new physical implementation for each specification of a finite state control. Therefore, the first step toward resolving this question involves answering whether there is a universal quantum Turing Machine. Deutsch [De85] describes a universal simulator for quantum Turing Machines; this simulator is not satisfactory from a complexity theoretic point of view because the simulation overhead is exponential in the running time of the simulated Turing Machine in the worst case.

In this paper, we prove the existence of a universal quantum Turing Machine whose simulation overhead is polynomially bounded. In full generality, on any given input a quantum TM produces a random sample from a

[*]Supported by an NSF Graduate Fellowship.

[†]Supported by grants NSF CCR-8896202 and NSF IRI-9120074.

25th ACM STOC '93-5/93/CA,USA
© 1993 ACM 0-89791-591-7/93/0005/0011...$1.50

probability distribution. We say that quantum TM T' simulates T with accuracy ϵ, if on every input x, T' outputs a sample from a distribution which is within total variation distance ϵ of the corresponding distribution for T. We prove that there is a universal quantum TM, which takes as input the description of a quantum TM T, time t, and input x, and simulates $T(x)$ for time t with accuracy ϵ. The slowdown is polynomial in t and $1/\epsilon$.

How powerful are polynomial time quantum TMs? The dynamics of a quantum TM are time reversible, and therefore it is necessary to appeal to results of Bennett [Be73] to show that polynomial time quantum TMs can simulate polynomial time deterministic TMs, and more generally, polynomial time probabilistic TMs. A sequence of papers [De85,DJ91,Jo91,BB92a,BB92b] study the power of quantum TMs that output the correct answer with certainty (probability 1): Deutsch and Jozsa [DJ91] showed the existence of a promise problem that can be solved exponentially faster on a quantum TM than on any classical deterministic TM. Berthiaume and Brassard [BB92a][BB92b] formulated this result in terms of oracle quantum TMs. They showed that there is an oracle relative to which there is a problem that can be solved in polynomial time on a quantum TM, whereas any classical probabilistic TM, which is restricted to *always* give the correct answer, requires exponential time. However, the crucial assumption in all the above results is that the classical TM is required to be error-free. For all the above problems, quantum TMs exhibit *no* advantage in running time over classical probabilistic TMs that are allowed a small probability of outputting the wrong answer on any input. It is well known that the error probability of such a probabilistic TM can be reduced to a negligible value (for example, much smaller than the chance of hardware failure) at a very modest increase in running time. For this reason the class BPP, of computational problems for which there exist bounded-error probabilistic polynomial time TMs, is regarded as the complexity class of efficiently solvable problems.

In this paper, we present the first evidence that quantum TMs might be more powerful than classical probabilistic TMs. We prove that there is an oracle relative to which there is a language that can be accepted in polynomial time by a quantum TM but cannot be accepted in $n^{o(\log n)}$ time by a bounded-error classical probabilistic TM. A more careful construction exhibits an oracle relative to which quantum polynomial time is not even contained in the class Arthur-Merlin where the verifier has $n^{o(\log n)}$ time (see [BM88] for a definition of this class).

Let BQP (*bounded-error quantum polynomial time*) be the class of languages that are accepted by a polynomial time quantum TM with error probability at most 1/3. It is not hard to show that $BQP \subseteq PSPACE$. Therefore, we cannot hope to prove that $BPP \subset BQP$ without resolving the longstanding open question $BPP \neq ?PSPACE$. In fact, Valiant [Va92] recently pointed out to us that the above result can be strengthened to say that $BQP \subseteq \#P$.

We feel that issues of quantum computation are relevant to physics as well as computer science. The study of the computational power of quantum TMs gives a method of demonstrating, in a quantifiable way, the inherent difference between the model proposed by quantum physics and *any* classical model. This may be viewed in the spirit of the famous Einstein-Podolsky-Rosen paradox and Bell's inequalities (discussed in [Fe82]) which demonstrate a difference in the statistical properties of the quantum model and any local hidden variable model. Also, showing that BQP is equivalent to BPP (or even just that BQP is contained in sub-exponential time as a preliminary step), would mean that the computer simulation of quantum physical systems could be efficiently carried out.

To introduce the model of a quantum TM, we need some new terminology. This is introduced in Section 2 in the context of classical probabilistic TMs. Then in Section 3, we formulate quantum TMs as a natural extension of this model. In Section 4, we give two useful alternative characterizations of a well-formed quantum TM. These are used in Section 5 in a series of constructions culminating in a universal quantum TM. Finally, in Section 6 we present our evidence that quantum TMs may be more powerful than classical probabilistic TMs. Please note that this is only a preliminary abstract; the final version of this paper will contain a number of issues that have been suppressed for this preliminary version.

2 A physics-like view of randomized computation.

Before we can formally define a QTM, we must introduce some new terminology. We will use the example of a classical probabilistic TM to introduce this terminology in a familiar setting. As a bonus, we will be able to precisely pin-point the point of departure in the definition of a QTM.

The new terminology is necessary because quantum mechanical systems cannot be observed without being altered. Let us consider how we can describe the computation of a PTM if we are not allowed to observe it while it is working. Therefore, we imagine that the PTM, T, is placed in a black box. The rules of the game are that we know the finite state control of T, as well as its initial configuration, and the interval of time between successive steps. What we must not observe are the outcomes of any coin-flips that T makes. If we mentally keep track of T's configuration, then we would know when it flipped its first coin, but we would not know which of the two possible successive configurations it entered. Instead, we will say that the machine is in a linear superposition of the two configurations, and assuming that the coin was fair, the coefficient for each configuration is $1/2$. We shall refer to the coefficient of a configuration as its *amplitude*. As the machine continues to compute and flip coins, it will, in general, be in some linear superposition of configurations.

Given the linear superposition at one time step, how do we compute the amplitude of a configuration in the superposition at the next time step? First we apply the finite state control's rules separately to each configuration in the superposition, and then we take a sum of these contributions weighted according to the superposition's amplitudes. Notice that this means the machine's finite state control is really specifying a linear mapping in the space of superpositions of configurations. This will remain true even if we give up the assumption that the machine must toss a fair two-sided coin, and instead allow the machine several arbitrary many-sided coins. Suppose we think of the transition function δ of a randomized TM on state set Q and alphabet Σ as

$$\delta \quad : \quad Q \times \Sigma \times \Sigma \times Q \times \{L, R\} \rightarrow [0, 1]$$

where $\delta(p, \sigma, \tau, q, d)$ gives the probability that if the machine is in state p reading a σ it will write a τ, enter state q, and move in direction d. Then the machine's transition function is a linear map, given by the matrix where the entry corresponding to the configuration $\mathbf{c_1}$ row and configuration $\mathbf{c_2}$ column is δ evaluated at the tuple which transforms $\mathbf{c_2}$ into $\mathbf{c_1}$ in a single step, or 0 if no such tuple exists. Since the number of possible configurations is unbounded, this matrix has infinite dimension. Nevertheless, it's finitely specified and satisfies a locality constraint. We shall refer to this matrix as the *time evolution operator* of the PTM. Depending on the particular machine, as the machine runs, its superposition may easily include an exponential number of distinct configurations. But, regardless of the machine, the linear superposition will always contain only non-negative amplitudes and the amplitudes will always sum to one.

Now, let's think about what happens when we open the box and observe the machine. We will find the machine in a specific configuration - not a linear superposition. The act of opening the box and observing the machine, forces it to choose a configuration out of the linear superposition - the rule is that each configuration is chosen to be the observed one with probability equal to its amplitude in the linear superposition. Next, consider the case where we are only interested in one particular bit of the configuration, and we take a quick peek inside the box to examine just that bit. Then we will see a one with probability equal to the sum of the amplitudes of configurations with a one in that position, and a zero with probability equal to the sum for configurations with a zero. We will then think of the machine as being in a linear superposition corresponding to the distribution of configurations conditioned according to the actual bit value we saw. This superposition could be computed by first erasing the amplitudes of all configurations inconsistent with the observed value, and then scaling up the remaining amplitudes to maintain unit sum. Therefore, we think of the act of observing a bit as operating on the machine to effect this restriction and renormalization.

Finally, note that although the superposition (distribution) of the machine after t steps may take exponential space to write down, we can sample from it online in the obvious way. Each time our simulation's configuration splits, we can choose a single next configuration randomly according to the split's weights. This on-line simulation may alternatively be achieved by observing the PTM after each step. It is easy to see that these multiple observations leave the final output distribution undisturbed.

3 Quantum computation

Our model of randomized computation is now surprisingly close to Deutsch's model of a quantum Turing Machines. The major change that is required is that in quantum physics, the amplitudes in a system's linear superposition and the entries in a system's time evolution matrix 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.

Making these changes to our model, we arrive at the following definitions (which are otherwise analogous to our model of PTMs).

Definition: A *quantum Turing Machine* (QTM) consists of a finite state set Q, a finite alphabet Σ, and a *quantum finite state control*

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

where $\delta(p, \sigma, \tau, q, d)$ gives the amplitude with which the machine in state p reading a σ will write a τ, enter state q, and move in direction d. As noted above, this transition function specifies a linear mapping M_δ (the time evolution operator) in the infinite dimensional space of superpositions of configurations. We will say that the machine's transition function is *well-formed* if the time evolution operator always preserves L_2 length. Well-formedness is an essential condition because we wish to associate squared magnitude of a configuration with its probability in a measurement.

Recall that a matrix M is called unitary if $M^\dagger M = MM^\dagger = I$ where $M^\dagger$ is the transpose conjugate of M. It's not hard to see that M preserves L_2 length if and only if $M^\dagger M = I$, and that if M is finite-dimensional this is in turn equivalent to M being unitary. However, for infinite dimensional matrices $M^\dagger M = I$ is not equivalent to $MM^\dagger = I$. Nevertheless, the time evolution operators of quantum TMs have special structure, and in fact these two conditions are equivalent for M_δ.

Theorem 1: A QTM is well-formed iff its time evolution operator is unitary.

Proof: Sketch. $M^\dagger M = I$ and M surjective together are enough to prove that M is unitary. So, suppose that the time evolution operator of a QTM is not surjective. Then, clearly there must be some particular configuration **c** which is not in the range of the operator. Moreover, any configuration which looks locally like **c** must also not be in the range. Now, select n contiguous cells of tape, and consider the the set S_n of configurations which have their tape head inside the n cells and whose tapes are blank outside the n cells. The corresponding columns of M_δ can only have non-zero entries in the corresponding rows and in the $2|Q||\Sigma|^n$ rows corresponding to configurations where the tape head has moved out of the n cells. However, at least $(n-2)|\Sigma|^{n-3}$ of the configurations in S_n look locally like **c**, and therefore also cannot be hit. Hence, for $n > 2|Q||\Sigma|^3 + 2$, not all of the columns of S_n can be mutually orthogonal, giving a contradiction. □

Definition: An *observation* of some bit of the superposition of a QTM returns a zero (one) with probability according to the sum of squared magnitudes of configurations in the superposition with a zero (one). Moreover, the superposition is immediately restricted to those configurations consistent with the observed value and renormalized to have unit L_2 length.

Quantum physics allows more general observations than we have included in our QTM. However, the effects of these more general observations can be efficiently simulated by the machines defined above. We do not have space to include a full discussion here. We will briefly note that with our current model of observations, we need only consider QTMs that are observed at only one time step. This is because when a bit would first be observed, we could copy the bit into a special area of the tape where it will never again be touched. The space of linear superpositions of configurations can then be thought of as being split into the subspace of configurations with a 0 and the subspace where configurations have a 1. Clearly, the portions of the linear superposition in these two subspaces will develop independently, since they can never interfere. But this means that if we later observe the saved bit, the linear superposition will immediately take on the superposition it would have developed had we seen the same value on an early observation of the bit. So, replacing each observation with a copy to this safe storage, we can allow the observer to sample from the same series of distributions just by observing a series of bits at a single time step.

In the following definition we will additionally assume a passive observer who merely reads a fixed bit at a fixed time. Again, although we will not give a discussion here, the power of a more general observer can be moved into the QTM being observed.

Definition: A language L is in the class EQP (*exact* or *error-free quantum polynomial time*) if there exists a QTM with a distinguished 'acceptance' tape cell, and a polynomial p, such that given any string x as input, observing the acceptance cell at time $p(x)$ correctly classifies x with respect to L. More generally, a language is in the class BQP if this classification can be accomplished with probability at least 2/3.

A QTM certainly cannot compute any non-recursive function. In fact, it's easy to see that by computing a QTMs linear superposition at each step, a PTM can simulate the QTM using exponential space and an exponential slowdown. By using depth first search, the straightforward simulation can be run using only polynomial space, so $BQP \subseteq PSPACE$. As Valiant [Va92] recently pointed out, this result can be strengthened to say that $BQP \subseteq \#P$.

However, this quantum model does lead to very curious effects. For example, suppose in our machine configurations $\mathbf{c_1}$ and $\mathbf{c_2}$ each lead to configuration **c** with amplitude $\sqrt{p}$. Then if we start the machine in either configuration and observe one step later, we'll see **c** with probability p. However, if we start the machine in some superposition of just $\mathbf{c_1}$ and $\mathbf{c_2}$ we won't necessarily see **c** with probability p. To see this, suppose our machine is started in the superposition $\alpha_1\mathbf{c_1} + \alpha_2\mathbf{c_2}$. If we observe the machine at time 0, we'll see $\mathbf{c_1}$ with probability $|\alpha_1|^2$ and $\mathbf{c_2}$ with probability $|\alpha_2|^2$, and then in either case at time 1, we'll see **c** with conditional probability p. But, if instead, we only observe at time 1, we'll see **c** with probability $|(\sqrt{p}\alpha_1)+(\sqrt{p}\alpha_2)|^2 = p|\alpha_1+\alpha_2|^2$ rather than $p(|\alpha_1|^2+|\alpha_2|^2)$. This means the simultaneous computations from $\mathbf{c_1}$ and from $\mathbf{c_2}$ interfere with each other. In fact, if $\alpha_1 = -\alpha_2$, then the contributions cancel out, and there's no chance we'll see **c** at all. Therefore, in this model observing the machine really can alter the machine's computation. For this reason, the straightforward simulation discussed above for

PTMs will no longer work. If we simulate a repeatedly observed machine we may obtain a distribution very far from that of the corresponding unobserved machine.

The above argument was certainly not rigorous, since we haven't argued that this hypothetical transition function can really occur in a well-formed quantum machine. Making such an argument requires checking that an infinite dimensional matrix is unitary. What we really need is some local characterization of the quantum finite state controls satisfying this requirement.

4 Alternative views of the well-formedness constraint

In this section, we will describe two additional characterizations of the well-formedness constraint on quantum finite state controls. First, we will show that the constraint is equivalent to requiring that the finite state control be reversible. Then, we will give a set of locally checkable constraints equivalent to the global one.

We will find it useful to briefly review the notion of a classical reversible TM, first examined as part of an effort to determine whether computation inherently dissipates energy.

Definition: A TM T is *reversible* if reversing its arrows gives a (deterministic) mapping of configurations T' that undoes the computation of T: For any pair of configurations $\mathbf{c}_1, \mathbf{c}_2$ $\mathbf{c}_1 \vdash_T \mathbf{c}_2$ iff $\mathbf{c}_2 \vdash_{T'} \mathbf{c}_1$.

Note that the arrow reversal of a reversible TM T is not necessarily a TM, since the reverse of a transition that both writes and moves must write in a non-local cell. Also, suppose we have a TM T which computes a function f, i.e. it outputs $f(x)$ on input x. Then if f is not injective there can clearly be no reversible TM that computes f (i.e. has $f(x)$ on its output tape and all other work tapes erased). However, Bennett [Be73] showed that for any TM T computing a function f, there is a reversible TM T such that T computes $x \cdot f(x)$ on input x; the running time of T is within a constant factor of the running time of T.

We can consider analogously the reversibility of QTMs: we want an arrow reversal of a QTM to reverse the machine's computation. However, we must decide what should happen to the arrows' amplitudes as we reverse their directions. We know that a quantum finite state control δ is well-formed iff $M_\delta\ M_\delta^\dagger$ is the identity matrix. Therefore $M_\delta^\dagger$ reverses δ. Now, looking at $M_\delta^\dagger$ we see that δ is well-formed iff it is reversible in the following way:

- The configuration map δ', specified by reversing the arrows and conjugating the amplitudes of δ, undoes the computation of δ: For any superpositions, $v_1 \vdash_\delta v_2$ iff $v_2 \vdash_{\delta'} v_1$.

It's easy to see that this is a generalization of classical reversibility, and thus, as Deutsch [De85] pointed out, the classical reversible TMs are a subclass of QTMs. So, using Bennett's result, any efficient classical computations can also be carried out efficiently on a QTM. In particular, P is contained in EQP and BPP is contained in BQP.

Next, we give a local characterization of the well-formedness constraint, directly in terms of the finite state control of the QTM.

Theorem 2: A quantum finite state control δ is well-formed iff it satisfies the following

- The amplitudes leaving any state-symbol pair have unit total squared magnitude:
$$\forall\ p, \sigma\ \in\ Q \times \Sigma \quad \sum_{\tau, q, d} |\delta(p, \sigma, \tau, q, d)|^2 = 1$$
- The superpositions of written character, new state, and direction leaving any two different state-symbol pairs are orthogonal:
$$\forall\ (p_1, \sigma_1) \neq (p_2, \sigma_2)\ \in\ Q \times \Sigma$$
$$\sum_{\tau, q, d} \delta(p_1, \sigma_1, \tau, q, d)\delta^*(p_2, \sigma_2, \tau, q, d)\ =\ 0$$
- If we fix all of δ's parameter except the new state, then δ specifies a linear superposition of new states consistent with this fixed data. The space of linear superpositions of states consists of two mutually orthogonal subspaces, one for each direction of movement, such that any superposition obtained in this way lies in the appropriate subspace.
$$\exists\ Q_L, Q_R\ \subseteq\ \mathbf{C}^{|Q|} \text{ mutually orthogonal such that}$$
$$\forall\ p, \sigma, \tau, d\ \in\ Q \times \Sigma \times \Sigma \times \{L, R\}$$
$$\sum_{q \in Q} \delta(p, \sigma, \tau, q, d)\ \mathbf{q}\ \in\ Q_d$$

Proof: Sketch. We know δ is well-defined iff $M_\delta^\dagger M_\delta$ gives the identity matrix, or equivalently iff the columns of M_δ 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 guaranteed to be orthogonal, and we need only consider pairs of configurations for which this is not the case. The second condition is necessary and sufficient for orthogonality of columns with the same head position, and the third for orthogonality of columns with tape head offset by exactly two cells. □

We can now argue that the interference which we described in the previous section can happen in a legally defined QTM. For example, consider the simple machine with $Q = \{q\}, \Sigma = \{0, 1\}$, and $\delta(q, b_1, b_2, q, R) = -\frac{1}{\sqrt{2}}$ if $b_1 = b_2 = 1$, $\frac{1}{\sqrt{2}}$ otherwise, and $\delta = 0$ everywhere else. It can be easily verified that this δ satisfies the local constraints and therefore is well-formed. Also if configurations $\mathbf{c_1}$ and $\mathbf{c_2}$ differ only in the bit under the tape head, then they each lead to the corresponding 0 configuration with amplitude $\frac{1}{\sqrt{2}}$. Therefore, this machine will exhibit the interference pattern discussed above.

5 The universal QTM

In this section, we exhibit a series of reductions culminating in a universal QTM. In the previous sections, we have highlighted two unique characteristics of QTMs: QTMs are always reversible, and they exhibit interference patterns. We must therefore simulate a QTM reversibly, and in such a way that the particular interference patterns of the machine are preserved. The simultaneous satisfaction of these two requirements makes building a universal machine for QTMs much more difficult than the straightforward constructions seen for many other classes of TMs. Our solution will consist of two stages. First, we will simplify the set of QTMs we must simulate by showing that the distribution computed by any QTM is also computed by some QTM from a simple subclass. Then, we will discuss reversible simulation and show how a single QTM can simulate any machine from this subclass.

First, we must say what it means for one QTM to simulate another.

Definition: In full generality, on any input x, a QTM T produces a sample from a probability distribution on configurations. We will say that machine T' *simulates* T if on input x it allows an observer to sample this distribution. Alternatively, we will say that T' simulates T with *accuracy* ϵ if it allows an observer to sample a distribution which is within ϵ total variation distance of this distribution.

Our first simplification will be to observe that we need work only with QTMs with real-valued transitions. The trick relies critically on the fact that we are working with TMs, and hence our configurations have a single state. In particular, we do not know if a similar result will hold for a parallel model of quantum computation like a quantum cellular automaton.

Lemma 3: Any QTM T is simulated, with constant slowdown, by some QTM T' satisfying

- **P1** T''s finite state control uses only real amplitudes.

Proof: Sketch. For each state q from T, give T' two states q_r and q_i. We'll rely on the natural correspondence between complex superpositions of configurations of T and real superpositions of configurations of T'

$$(a + b\sqrt{-1})\, \mathbf{q} \quad - \quad a\, \mathbf{q_r} + b\, \mathbf{q_i}$$

We let δ' be the finite state control compatible with δ under this correspondence:

$$\begin{aligned}
\delta'(p_r, \sigma, \tau, q_r, d) &= Re(\delta(p, \sigma, \tau, q, d)) \\
\delta'(p_r, \sigma, \tau, q_i, d) &= Im(\delta(p, \sigma, \tau, q, d)) \\
\delta'(p_i, \sigma, \tau, q_r, d) &= -Im(\delta(p, \sigma, \tau, q, d)) \\
\delta'(p_i, \sigma, \tau, q_i, d) &= Re(\delta(p, \sigma, \tau, q, d))
\end{aligned}$$

It can be easily verified, that T' is a well-formed QTM simulating T. □

Next, we restrict the complexity of the quantum finite state control by requiring that it enter any particular state from only one direction.

Lemma 4: Any QTM T is simulated, with constant slowdown, by some QTM T' satisfying **P1** and

- **P2** T' only enters any particular state while moving in one direction: 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 $d_1 = d_2$.

Proof: Sketch. Given T with finite state control δ, then, in the language of our local reversibility constraints, we choose orthonormal bases $\{v_{L,1}, \dots, v_{L,m}\}$ and $\{v_{R,1}, \dots, v_{R,n}\}$ for Q_L and Q_R. Then, we augment Q with a state for each $v_{d,i}$. Now, if T's finite state control is translated so that it maps from Q to the $v_{d,i}$, then it will obey **P2**. So T' can implement this map, and then using two more time steps and another set of auxiliary states, change basis back from the $v_{d,i}$ to Q. This achieves the required simulation with a slowdown by a factor of 3. □

The class of machines satisfying **P2** are much easier to analyze than general QTMs. The time evolution matrix of a general QTM is unitary, but this matrix is infinite dimensional, prohibiting the use of this unitarity except in a global argument. However, the finite state control of one of our simpler machines also specifies a finite dimensional unitary matrix. Indeed, we can write the entries of a δ satisfying **P2** in a matrix, which we will call L_δ, indexing the columns by the current state and symbol and the rows by the new state and symbol (the direction in which the machine moves is implied by the new state). Then the first and second local well-formedness constraints guarantee that L_δ must be unitary.

This local unitary characterization will allow us to consider only machines whose configurations have indegree and outdegree at most 2.

Definition: We will say that a square matrix M *decomposes* into square matrices $M_1, \ldots, M_n$ if M consists of independent copies of the M_i's. We will call M *k-decomposable* where k is the maximum dimension of any of these matrices.

Lemma 5: Any QTM T is simulated, with constant slowdown, by some QTM T' satisfying **P1-P2** and

- **P3** $M_{\delta'}$ is 2-decomposable.

Proof: Sketch. First, an easy argument by induction can be used to show that any d-dimensional real unitary matrix can be written as a product of $k = poly(d)$ 2-decomposable d-dimensional real unitary matrices. We apply this construction to L_δ. We augment T''s state set and alphabet to include $k+1$ copies of the old versions. Then, we let the ith matrix from the product determine the transitions from the ith copy to the $i+1$st copy, as the machine steps back and forth. Finally, the $k+1$st copy map deterministically back to the first copy, while the tape head moves in the proper direction according to the state. The new machine will simulate the old machine with slowdown by a constant factor. □

For our universal machine to simulate a QTM T we certainly must provide an encoding of T as input. However, the set of possible input strings is countable whereas T's amplitudes can be chosen arbitrarily from the uncountable set of reals. Therefore, we cannot hope for a single universal machine to exactly simulate all QTMs. So, our final simplification will be to restrict to a countable set of QTMs which approximate the whole set.

Lemma 6: There exists a 2×2 matrix R such that any QTM T is simulated up to time t, with accuracy ϵ and slowdown polynomial in t and $1/\epsilon$, by some QTM T' satisfying **P1-P3** and

- **P4** $M_{\delta'}$ can be decomposed into the set $\{[-1], [1], R\}$

Proof: Sketch. Consider a machine T satisfying **P1-P3**. We can simulate T if we can simulate each of the unitary 2×2 matrices of M_δ in such a way that the operation on any vector is within $O(t/\epsilon)$. This will be possible if we let R be a rotation in the plane such that for any other rotation R', there exists a $k \leq O(poly(t/\epsilon))$ such that R^k is within $O(\epsilon/t)$ of R'. Now, rotation by the angle

$$2\pi \sum_{i=1}^{\infty} 2^{-2^i}$$

gives one such R. □

Now, we will show that the countable subclass of machines satisfying **P1-P4** contains a universal machine. First, we must discuss the reversible simulation of TMs. In fact, the construction of a classical reversible TM that can simulate another classical reversible TM provided as input is not immediate.

Consider the straightforward simulation method for TMs: We insist that the target machine's finite state control be provided in table form as input. We then use chunks of cells, or 'supercells', of our tape to simulate cells of the target machine. Each supercell holds the corresponding symbol from the target machine's tape, and the supercell under the simulated tape head also contains the target machine's state. Simulating a single time step of the target machine just requires a table lookup and a bit of copying. Unfortunately, copying is an irreversible process. However, copying can be achieved reversibly in the case that the destination is known to be empty, for example by adding the desired entry bitwise mod 2 to the current contents of the destination. Similarly a supercell can be erased by bitwise addition using a copy of its contents. This suggests the following method. First we insert a flag into the table at the p, σ entry while erasing p, σ from the tape. Then we copy the new information from the table into the now empty locations. Finally, we use a 'reverse table' to allow us to use the new information written on the tape to erase the flag from the table.

Building a universal QTM introduces a new difficulty. Clearly, we'll again need to use many steps of our machine to simulate one atomic step of the desired machine. Now, however, the step we'll need to simulate will not be a deterministic mapping, but instead will involve quantum interference. The problem is that if we spread the interference over several steps of our simulation, the intermediate interference patterns may not be reversible. For example, consider the straightforward extension of the above simulation to QTMs. Since we need to consider only time evolution matrices which can be decomposed into $\{[1], [-1], R\}$, we can encode each finite state control into a table as before. But now when we look up a state-symbol pair in the simulated machine's tables we may find not one set of instructions for updating the tape, but a listing of two possible updates. We can have our machine split appropriately into two paths of computation and then try to follow the two sets of instructions in superposition. However, reversibly erasing the old state and symbol (or the flag in the table) is now problematic because the new tape contents are not sufficient to recover the old information.

Our solution will be to simulate the interference atomically. In other words, all but one of the steps in our simulation will be reversible deterministic steps. In the one special step, we will simulate all of the splitting and interference of the desired machine's step. To do this we will need to somehow bring the information which determines the desired splitting (which in general will be spread across the bits of the arbitrarily large supercell) into one state and symbol of our universal machine. The following is a brief sketch of how this can be accomplished. Since L_6 is 2-decomposable, we can group the entries of the first table into pairs where both list the same two instructions. The two entries, instead of listing these instructions, will each contain a pointer to a single entry of a second table which contains a single copy of the two instructions. As before, we first insert a flag at the p, σ entry of the first table while erasing p, σ from the tape. Next, we insert a flag at the appropriate entry in the second table, while erasing the first flag and adding a single bit to our machine's state according to which of the two possible first table entries we came from. This bit can then be transformed according to the special transformation R, and the two possible updates can be followed in superposition. But now, the new tape contents are sufficient to deterministically determine which entry of the second table was used, and hence to erase the flag. This technique allows us to prove the following lemma.

Lemma 7: There exists an encoding of the machines satisfying **P1-P4** and a QTM U such that given as input the encoding of some machine T, U simulates T with constant slowdown.

Lemmas 6 and 7 together imply our theorem.

Theorem 8: There exists a universal QTM U which when input a description of any QTM T, and any $t > 0$, simulates T for t steps with accuracy ϵ, and with slowdown polynomial in t and $1/\epsilon$.

6 Computing with a QTM

Can a QTM compute faster than a classical TM? In this section we will give some evidence suggesting a positive answer.

First let us introduce the Fourier power spectrum of a function mapping n-bit strings to real values. The usual way of thinking about such a function is by listing its 2^n values. Each point of the vector space $\mathbb{R}^n$ is thereby associated with a function on n-bit strings, and we express our function in terms of the orthonormal basis $b_0, \ldots, b_{2^n-1}$, where $b_i(x) = 1$ if $x = i$ and 0 otherwise. However, another useful basis for this space can be constructed using parity functions. For each subset of $[1, n]$, identified by an n-bit integer i, we define the normalized parity function par_i by $par_i(x) = 1$ if an even number of the bits of x in the positions selected by i are 1, and -1 otherwise (we'll denote this parity of x with respect to i by $x \oplus i$). Clearly, the 2^n parity functions also form an orthonormal basis of $\mathbb{R}^n$. So, we could also describe a function f by listing its unique representation as a linear combination of the parity bases $f = \sum_i \lambda_i par_i$. Since representing f by $\{\lambda_i\}$ corresponds to doing discrete spectral analysis on the n-hypercube, we call $\{\lambda_i\}$ the *Fourier power spectrum* of the function f. Clearly, we have $\sum_i \lambda_i^2 = \sum_x f(x)^2$, and if we restrict our attention to boolean functions with range $\{-1, 1\}$, then we have $\sum_i \lambda_i^2 = 2^n$. So, we define the *Fourier sampling problem* by providing as input a classical program to compute f, and requiring as output each i with probability $\frac{1}{2^n}\lambda_i^2$.

We will show that a QTM described by Deutsch and Jozsa [DJ91] exactly solves the Fourier sampling problem in polynomial time. This result does not prove that QTMs are more powerful than PTMs since a PTM can approximate the same distribution in polynomial time. Specifically, we can show that a PTM can, with high probability, sample from a distribution within ϵ total variation distance of the desired Fourier distribution. However, we will show that the QTM solution can be extended to solve a difficult problem.

First, we must describe Deutsch and Jozsa's QTM. In general, a QTM effects a unitary transformation. The particular one we will find useful is the Fourier transformation. Suppose we associate each standard basis function b_i with having i written on the QTM's tape, and that instead of using the standard representation of our function $f = \sum_i f(i) b_i$, we represent f with a superposition of these tapes, $f = \sum_i f(i)\ \mathbf{i}$. Then the Fourier transformation given by the matrix F, where $F_{i,j} = \frac{1}{2^{n/2}} i \oplus j$, converts this representation to the parity basis. To see this, note that applying F gives

$$\frac{1}{2^{n/2}} \sum_i \left(\sum_j f(j)\ (i \oplus j) \right) \mathbf{i}$$
$$= \frac{1}{2^{n/2}} \sum_i \lambda_i\ \mathbf{i}$$

This is exactly what we want, since observing this superposition will give i with probability $\frac{1}{2^n}\lambda_i^2$.

The transformation F also allows the QTM to construct the original superposition representing f, $\sum_i f(i)\ \mathbf{i}$. If we start with the tape 0^n and apply F, the result is the superposition where all n-bit tapes have equal magnitude and identical phase: $\frac{1}{2^{n/2}} \sum_i \mathbf{i}$. Then, the QTM can (again appealing to Bennett's result) follow the program for f, computing each $f(i)$ in the piece of the superposition with i on the tape. Next, the machine can apply a phase shift of 1 or -1 to each piece according to the value $f(i)$. We have not yet achieved the desired superposition because the pieces of the superposition differ not only in the string i, but also in the value $f(i)$ and in various information left over from running the program for f. However, these other differences can be removed by reversing the computation of f, giving us the desired superposition.

Finally, we describe how the QTM can effect the transformation F. It is easy to see that the QTM can accomplish this by applying the unitary transformation

$$\mathbf{b} \longrightarrow \frac{1}{\sqrt{2}}\,\mathbf{0} + (-1)^b\frac{1}{\sqrt{2}}\,\mathbf{1}$$

to each of the n bits one at a time.

In summary, the QTM solves the sampling problem in $O(n)$ time. First, it applies F to a piece of n tape cells initially all blank. Next, it computes $f(i)$ in parallel for all i in superposition. Then, it applies phase shift $f(i)$, reverses the computation of $f(i)$, and once again applies F.

Remarkably, this algorithm only requires calling the algorithm for f twice. We can see the power of this ability more clearly by turning this sampling problem into a deterministic problem. Promising that the function f is one of the parity functions, we define the following special case of the sampling problem

Definition: PARITY

PROMISE: $\exists$ n-bit k such that $\forall$ n-bit i, $f(i) = i \oplus k$.
ANSWER: k

The QTM we have already described amazingly solves for the n-bits of information to answer PARITY using only two computations of f. Solving PARITY in the straightforward manner on a classical TM requires computing n values of f. This suggests that a recursive construction will separate the two classes of machines.

To ready this problem for recursion, we first need to turn PARITY 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'll make the problem recursive. For each problem instance of size n, we'll 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. The QTM will be able to solve an instance of size n recursively in time $T(n)$ where

$$T(n) \leq O(n) + 2T(n/2) \quad \Longrightarrow \quad T(n) \leq O(n \log n)$$

However, the straightforward recursive solution on a PTM will require time $T(n)$ where

$$T(n) \geq \Omega(n)T(n/2) \quad \Longrightarrow \quad T(n) \geq n^{\Omega(\log n)}$$

A problem at depth $d \in [0, \log n + 1]$ is described by the strings $i_1, i_2, \ldots, i_d$ of length $n, n/2, \ldots, 2^{\log n - d+1}$ which specify its location in the recursive tree. Once again we have two functions f and g. Function f will be used to bottom out the recursion, and therefore when provided with a series of strings of length $n, n/2, \ldots, 1$, it gives the answer to the identified problem at depth $\log n + 1$. Function g takes input strings of length $n, n/2, \ldots, 2^{\log n - d+1}$ bits specifying a problem at level $d \leq \log n$ in the tree, plus a string of length $2^{\log n - d+1}$ specifying one of the parity patterns of length $2^{\log n - d+1}$. Its answer map the problem's $2^{\log n - d+1}$-bit answer back to a single bit. Formally we define a problem at depth d, identified by the strings $i_1, \ldots, i_d$, by

Definition: REC $(d, i_1, \ldots, i_d)$

PROMISE: If $d \leq \log n$, then $\exists$ $2^{\log n - d+1}$-bit k
such that $\forall$ $2^{\log n - d+1}$-bit i_{d+1},
REC $(d+1, i_1, \ldots, i_{d+1}) = i_{d+1} \oplus k$.
ANSWER: If $d \leq \log n$, $g(i_1, \ldots, i_d, k)$,
otherwise $f(i_1, \ldots, i_d)$.

Finally, to prevent the PTM from directly analyzing the programs for f and g to determine the answer, we instead present the machine with an oracle which can be queried for values of f and g. It is straightforward to alter the above QTM, using recursion and replacing function evaluation with oracle lookup, to construct a QTM to solve REC in time polynomial in n. However, when the oracle is chosen randomly, then, even in the best case, $n^{\Omega(\log n)}$ questions must be asked to determine the answer to REC with bounded probability of error. We omit the following lemma's proof due to the space constraints of this abstract.

Lemma 9: There exists a $c > 0$ such that for sufficiently large n the following is true: Fix arbitrarily the answers to any $n^{c \log n}$ queries that can be asked of the oracle, and then choose an oracle uniformly at random among all those which satisfy the promises and also agree with the fixed answers. Then answer to REC will be 1 with probability no further than $1/6$ away from $1/2$.

This lemma shows that a PTM (or even an Arthur-Merlin verifier) with time $n^{o(\log n)}$ cannot solve REC for a randomly chosen oracle with bounded error probability. So, if we choose a different random oracle independently for each n, and consider the language $L = \{1^n : \text{REC} = 1\}$, then any particular PTM (AM verifier) will fail to accept L with probability 1. Since, PTMs (AM verifiers) are countable, we will cause all machines to fail with probability 1.

Theorem 10: There exists an oracle relative to which the class EQP is not contained in two-sided error $n^{o(\log n)}$ time.

Theorem 11: There exists an oracle relative to which the class EQP is not contained in the class Arthur-

Merlin with verifier time $n^{o(\log n)}$.

7 Discussion

The main open issue that remains is characterizing the exact power of BQP. In particular, we would like to know whether $BQP \neq ? BPP$. As we have pointed out earlier, proving $BQP \neq BPP$ would also resolve a long-standing open question in complexity theory. However, it may be possible to obtain such a result under some standard complexity assumptions such as $P \neq NP$ or the existence of one-way functions. In general, a result showing the increased power of quantum computation might provide another method of demonstrating the inherent difference between quantum physics and any classical model. On the other hand, showing $BQP = BPP$ would show that quantum physical systems can be efficiently simulated on a classical computer.

In the quantum Turing Machines we have considered so far, the transitions must always move the tape head, either left or right. For classical machines, it is trivial to use such a machine to simulate a machine which is also allowed to leave its tape head unmoved. However, in QTMs, allowing stationary head transitions introduces more complex interference patterns. In particular, the third of our local well-formedness constraints does not generalize to three subspaces Q_L, Q_R, Q_-. The problem is that two configurations with tape heads one cell apart can interfere with the head in either of the two cells. The more complex local reversibility constraints for such machines will not allow a change of basis into states which specify the direction of previous movement, as in Lemma 4. Therefore our universality construction would not apply to machines in this more general formulation. We are currently searching for another simulation method that will allow us to decompose the time evolution matrices of these machines, and therefore extend out universal QTM construction.

Acknowledgements

We wish to thank Noam Nissan, Leonard Schulman, Richard Jozsa, Gilles Brassard, and Charles Bennett for helpful discussions, and to thank Richard Jozsa for noticing an error in an earlier version of this paper.

References

[BM88] L.Babai and S.Moran. "Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes." *Journal of Computer and System Sciences*, Vol. 36, 1988, pp. 254-276.

[Be73] C.Bennett. "Logical reversibility of computation." *IBM J. Res. Develop.*, Vol. 17, 1973, pp. 525-532.

[BB92a] A.Berthiaume and G.Brassard. "The quantum challenge to structural complexity theory." *Proceedings of 7th IEEE Conference on Structure in Complexity Theory*, 1992.

[BB92b] A.Berthiaume and G.Brassard. "Oracle quantum computing." *Proceedings of the Physics of Computation*, Dallas, 1992.

[De85] D.Deutsch. "Quantum theory, the Church-Turing principle and the universal quantum computer." *Proc. R. Soc. Lond.*, Vol. A400, 1985, pp. 97-117.

[De89] D.Deutsch. "Quantum computational networks." *Proc. R. Soc. Lond.*, Vol. A425, 1989, pp. 73-90.

[DJ91] D.Deutsch and R.Jozsa. "Rapid solution of problems by quantum computation." *Proc. R. Soc. Lond.*, Vol. A439, 1992, pp. 553-558.

[Fe82] R.Feynman. "Simulating physics with computers." *International Journal of Theoretical Physics*, Vol. 21, nos. 6/7, 1982, pp. 467-488.

[Jo91] R.Jozsa. "Characterizing classes of functions computable by quantum parallelism." *Proc. R. Soc. Lond.*, Vol. A435, 1991, pp. 563-574.

[Va92] L.Valiant. Personal communication, 1992.