

Closed Timelike Curves Make Quantum and Classical Computing Equivalent

Scott Aaronson*
MIT

John Watrous†
University of Waterloo

Abstract

While closed timelike curves (CTCs) are not known to exist, studying their consequences has led to nontrivial insights in general relativity, quantum information, and other areas. In this paper we show that if CTCs existed, then quantum computers would be no more powerful than classical computers: both would have the (extremely large) power of the complexity class PSPACE, consisting of all problems solvable by a conventional computer using a polynomial amount of memory. This solves an open problem proposed by one of us in 2005, and gives an essentially complete understanding of computational complexity in the presence of CTCs. Following the work of Deutsch, we treat a CTC as simply a region of spacetime where a “causal consistency” condition is imposed, meaning that Nature has to produce a (probabilistic or quantum) fixed-point of some evolution operator. Our conclusion is then a consequence of the following theorem: given any quantum circuit (not necessarily unitary), a fixed-point of the circuit can be (implicitly) computed in polynomial space. This theorem might have independent applications in quantum information.

1 Introduction

The possibility of closed timelike curves (CTCs) within general relativity and quantum gravity theories has been studied for almost a century [11, 15, 13]. A different line of research has sought to understand the *implications* of CTCs, supposing they existed, for quantum mechanics, computation, and information [9, 8, 5].

In this paper we contribute to the latter topic, by giving the first complete characterization of the computational power of CTCs. We show that if CTCs existed, then both classical and quantum computers would have exactly the power of the complexity class PSPACE, which consists of all problems solvable on a classical computer with a polynomial amount of memory. To put it differently, CTCs would make polynomial time equivalent to polynomial space as computational resources, and would also make quantum and classical computers equivalent to each other in their computational power. Our results treat CTCs using the “causal consistency” framework of Deutsch [9].

It will not be hard to show that classical computers with CTCs can simulate PSPACE and be simulated in it (though as far as we know, this result is new). The main difficulty will be to show that quantum computers with CTCs can be simulated in PSPACE. To prove this, we need to give

*Email: aaronson@csail.mit.edu.

†Email: watrous@cs.uwaterloo.ca.

an algorithm for (implicitly) computing fixed-points of superoperators in polynomial space. Our algorithm relies on fast parallel algorithms for linear algebra due to Borodin, Cook, and Pippenger [7], and might be of independent interest.

The paper is organized as follows. In Section 2, we explain needed background about Deutsch’s causal consistency framework and computational complexity, and review previous work by Bacon [5], Brun [8], and Aaronson [1]. In Section 3, we show that classical computers with CTCs have exactly the power of PSPACE. Section 4 extends the analysis of Section 3 to show that *quantum* computers with CTCs have exactly the power of PSPACE. In that section, we make the simplifying assumption that all quantum gates can be applied perfectly and that amplitudes are rational. In Section 5, we consider what happens when gates are subject to finite error, and extend previous work of Bacon [5] to show that quantum computers with CTCs can solve PSPACE problems in a “fault-tolerant” way. We conclude in Section 6 with some general remarks and open problems.

2 Background

2.1 Causal Consistency

It was once believed that CTCs would lead inevitably to logical inconsistencies such as the Grandfather Paradox. But in a groundbreaking 1991 paper, Deutsch [9] showed that this intuition fails, provided the physics of the CTC is quantum-mechanical. Deutsch’s insight was that a CTC should simply be regarded as a region of spacetime where Nature enforces a requirement of *causal consistency*: in other words, that the evolution operator within that region should map the state of the initial hypersurface to itself. Given the evolution operator f , Nature’s “task” is thus to find a fixed-point of f : that is, an input x such that $f(x) = x$. Of course, not every deterministic evolution operator f has a fixed-point: that is just one way of stating the Grandfather Paradox. On the other hand, it is a basic linear-algebra fact that every quantum operation Φ has a fixed-point: that is, a density matrix ρ such that $\Phi(\rho) = \rho$. For any Φ , such a ρ can then be used to produce a CTC evolution that satisfies the causal consistency requirement. So for example, a consistent resolution of the Grandfather Paradox is that you are born with 1/2 probability, and if you are born you go back in time to kill your grandfather, therefore you are born with 1/2 probability, etc.

Notice that Deutsch’s resolution works just as well in classical probabilistic theories as in quantum-mechanical ones. For just as every quantum operation has a fixed-point, so every Markov chain has a stationary distribution. What matters is simply that the state space and the set of transformations are such that fixed-points exist.

Although CTCs need not lead to inconsistencies, Deutsch pointed out that they *would* have striking consequences for the theory of computing. As an example, CTCs could be exploited to solve NP-complete and other “intractable” computational problems using only polynomial resources. To see this, suppose some integers $x \in \{0, 1, \dots, 2^{n-1}\}$ are “solutions” and others are not, and that our goal is to find a solution in time polynomial in n , assuming solutions exist and can be recognized efficiently. Then we could build a machine M that applied the following transformation to its input x : if x is a solution then $M(x) = x$, while if x is not a solution then $M(x) = (x + 1) \bmod 2^n$. Now suppose we use a CTC to feed M its own output as input. Then it is not hard to see that the only way for the evolution to satisfy causal consistency is for M to input, and output, a solution.

In this way, an exponentially-hard computational problem could get solved without exponential effort ever being invested to solve it, merely because that is the only way to satisfy causal consis-

tency. A rough analogy would be Shakespeare’s plays being written by someone from the present going back in time and dictating the plays to him.

It is sometimes said that if CTCs existed, then one could *obviously* do computations of unlimited length in an instant, by simply computing the answer, then sending it back in time to before one started. However, this proposal does not work for two reasons. First, it ignores the Grandfather Paradox: what happens if, on receiving the output, one goes back in time and changes the input? Second, it is perhaps unclear why a computation lasting 10^{1000} years should be considered “feasible,” merely because we are able to obtain the solution *before* performing the computation. It seems that an honest accounting should require the computations performed inside the CTC to be efficient (say, polynomial-time), with any computational speedup coming from the requirement of causal consistency.

2.2 Complexity Theory

For background on classical computational complexity theory, see for example Arora and Barak [4]; for a recent survey of quantum complexity theory, see Watrous [18]. Here, we briefly describe the main complexity classes we will consider. **PSPACE** (Polynomial Space) is the class of decision problems that are solvable by a classical computer, using an amount of memory that is bounded by a polynomial function of the size of the input n (but possibly an exponential amount of time). An example of such a problem is, given a configuration of an $n \times n$ Go board, to decide whether White has a winning strategy using n^2 or fewer moves. **NP** (Nondeterministic Polynomial-Time) is the class of decision problems for which every “yes” answer has a polynomial-time-checkable, polynomial-size proof or witness. **NP**-complete problems are, loosely speaking, the “hardest” problems in **NP**: that is, those **NP** problems to which all other **NP** problems can be efficiently reduced. An example is, given a graph, to decide whether it has a Hamiltonian cycle (that is, a cycle that visits each vertex exactly once). **PSPACE** contains **NP** (thus, in particular, the **NP**-complete problems)—since in polynomial space, one can simply loop over all possible witnesses and see if any of them are correct. However, **PSPACE** is believed to be considerably larger than **NP**. So in saying that computers with CTCs can efficiently solve **PSPACE** problems, we are saying something stronger than just that they can solve **NP**-complete problems.

Our main result is that computers with CTCs have precisely the power of **PSPACE**, and that this is true whether the computers are classical or quantum. Previously, Watrous [17] showed that **BQPSPACE** (Bounded-Error Quantum Polynomial Space) is equal to **PSPACE**: that is, any problem solvable by a quantum computer with polynomial memory is also solvable by a classical computer with polynomial memory. (By contrast, quantum computers are conjectured to offer an exponential improvement over classical computers in *time*.) Here, we show that quantum computers are polynomially equivalent to classical computers in the CTC setting as well.

2.3 Related Work

Besides Deutsch’s paper [9], we know of three other works directly relevant to computational complexity in the presence of CTCs. First, Bacon [5] showed that **NP**-complete problems can be solved with polynomial resources, even using CTCs that are only “one bit wide” (i.e., able to transmit a single qubit or probabilistic classical bit back in time).¹ Bacon also showed that,

¹On the other hand, Bacon’s approach would require a polynomial number of such CTC’s, rather than a single CTC as in Deutsch’s approach.

using his approach, one can solve not only NP problems but even #P problems, which involve *counting* solutions rather than just finding one. (The class #P—or more formally its decision version P^{#P}—is a subclass of PSPACE, with the containment believed to be strict.) Finally, Bacon showed that techniques from the theory of quantum fault-tolerance could be used to make certain CTC computations, including the ones used to solve #P problems, robust to small errors.

Second, Brun [8] claimed to show that CTCs would allow the efficient solution of any problem in PSPACE. However, Brun did not specify the model of computation underlying his results, and the most natural interpretation of his “CTC algorithms” would appear to preclude their solving PSPACE problems. For Brun, a fixed-point of a CTC evolution seems to be necessarily *deterministic*—in which case, finding such a fixed-point is an NP problem (note that NP is almost universally believed to be smaller than PSPACE). Thus, to prove that classical *or* quantum computers with CTCs give the full power of PSPACE, it seems essential to adopt Deutsch’s causal consistency model (which Brun does not discuss).

Third, as part of a survey on “NP-complete Problems and Physical Reality” [1], Aaronson sketched the definitions of P_{CTC} and BQP_{CTC} (classical and quantum polynomial time with CTCs) that we adopt in this paper. He also sketched a proof that PSPACE = P_{CTC} ⊆ BQP_{CTC} ⊆ EXP. That is, classical computers with CTCs have exactly the power of polynomial space, while quantum computers with CTCs have at least the power of polynomial space and at most the power of classical exponential time. The key problem that Aaronson left open was to pin down the power of quantum computers with CTCs precisely. This is the problem we solve in this paper.

3 The Classical Case

To state our results, it is crucial to have a formal model of computation in the presence of CTCs.

We define a *deterministic CTC algorithm* \mathcal{A} to be deterministic polynomial-time algorithm that takes as input a string $x \in \{0, 1\}^n$, and that produces as output a Boolean circuit $C = C_x$, consisting of AND, OR, and NOT gates. The circuit C acts on bits in two registers: a *CTC register* \mathcal{R}_{CTC} , and a *causality-respecting register* \mathcal{R}_{CR} . The registers \mathcal{R}_{CTC} and \mathcal{R}_{CR} consist of $p(n)$ and $q(n)$ bits respectively, for some polynomials p and q depending on \mathcal{A} . Thus, C can be seen as a Boolean function $C : \{0, 1\}^{p(n)+q(n)} \rightarrow \{0, 1\}^{p(n)+q(n)}$, which maps an ordered pair $\langle y, z \rangle \in \mathcal{R}_{CTC} \times \mathcal{R}_{CR}$ to another ordered pair $C(\langle y, z \rangle)$.

For convenience, we assume that the causality-respecting register \mathcal{R}_{CR} is initialized to $0^{q(n)}$. The CTC register, on the other hand, must be initialized to some probability distribution over $p(n)$ -bit strings that will ensure causal consistency. More formally, let \mathcal{D} be a probability distribution over $\mathcal{R}_{CTC} \times \mathcal{R}_{CR}$, and let $C(\mathcal{D})$ be the distribution over $\mathcal{R}_{CTC} \times \mathcal{R}_{CR}$ induced by drawing a sample from \mathcal{D} and then applying C to it. Also, let $[\cdot]_{CTC}$ be an operation that discards the causality-respecting register (i.e., marginalizes it out), leaving only the CTC register. Then we need the initial probability distribution \mathcal{D} over $\mathcal{R}_{CTC} \times \mathcal{R}_{CR}$ to satisfy the following two conditions:

- (i) \mathcal{D} has support only on pairs of the form $\langle y, 0^{q(n)} \rangle$.
- (ii) \mathcal{D} satisfies the *causal consistency equation* $[\mathcal{D}]_{CTC} = [C(\mathcal{D})]_{CTC}$.

We claim that such a \mathcal{D} always exists. This is easy to prove: let $C'(y) := [C(\langle y, 0^{q(n)} \rangle)]_{CTC}$ be the induced circuit that acts only on the CTC register. Then it suffices to find a distribution \mathcal{D}' over \mathcal{R}_{CTC} such that $C'(\mathcal{D}') = \mathcal{D}'$. To find such a \mathcal{D}' , we consider the graph of the function

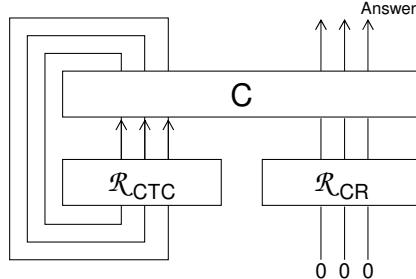


Figure 1: Diagram of a classical CTC computer. A circuit C performs a polynomial-time computation involving “closed timelike curve bits” (the register \mathcal{R}_{CTC}) as well as “causality-respecting bits” (the register \mathcal{R}_{CR}). Nature must then find a probability distribution over \mathcal{R}_{CTC} that satisfies Deutsch’s causal consistency equation. The final answer is read out from \mathcal{R}_{CR} .

$C' : \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^{p(n)}$, find a cycle in that graph (which must exist, since the graph is finite), and let \mathcal{D}' be the uniform distribution over points in the cycle. Finally we set $\mathcal{D} = \langle \mathcal{D}', 0^{q(n)} \rangle$.

We are now ready to define the complexity class P_{CTC} , of problems solvable using classical computers with CTCs. We say that a CTC algorithm \mathcal{A} *accepts* the input x if, for every distribution \mathcal{D} satisfying conditions (i) and (ii) above, $C(\mathcal{D})$ has support only on pairs of the form $\langle y, z1 \rangle$ (i.e., such that the last bit of the causality-respecting register is a 1). Likewise, we say \mathcal{A} *rejects* x if for every \mathcal{D} satisfying (i) and (ii), $C(\mathcal{D})$ has support only on pairs of the form $\langle y, z0 \rangle$. (Of course, it is possible that $C(\mathcal{D})$ has support on both kinds of pairs, in which case \mathcal{A} neither accepts nor rejects.) We say \mathcal{A} *decides the language* $L \subseteq \{0, 1\}^*$ if \mathcal{A} accepts every input $x \in L$, and rejects every input $x \notin L$. Then P_{CTC} is the class of all languages L that are decided by some deterministic CTC algorithm.

Let us make a few remarks about the definition. First, the requirement that some polynomial-time algorithm \mathcal{A} output the circuit $C = C_x$ is intended to prevent hard-to-compute information from being hard-wired into the circuit. This requirement is standard in complexity theory; it is also used, for example, in the definition of BQP. Second, our definition required C to succeed with certainty, and did not allow C to introduce its own randomness, besides that produced by the causal consistency condition. We could relax these requirements to obtain the complexity class BPP_{CTC} , or bounded-error probabilistic polynomial time with access to a CTC. However, it will turn out that $\text{P}_{CTC} = \text{BPP}_{CTC} = \text{PSPACE}$ anyway.

3.1 Results

We now prove $\text{P}_{CTC} = \text{PSPACE}$.

Lemma 1 $\text{P}_{CTC} \subseteq \text{PSPACE}$.

Proof. Let C be a polynomial-size circuit that maps $\mathcal{R}_{CTC} \times \mathcal{R}_{CR}$ to itself, as in the definition of P_{CTC} . Then our PSPACE simulation algorithm is as follows. First, let $C'(y) := [C(\langle y, 0^{q(n)} \rangle)]_{CTC}$ be the induced circuit that acts only on \mathcal{R}_{CTC} . Then given a string $y \in \{0, 1\}^{p(n)}$, say y is *cyclic* if $C'^{(k)}(y) = y$ for some positive integer k . In other words, y is cyclic if repeated application of C'

takes us from y back to itself. Clearly every C' has at least one cyclic string. Furthermore, it is clear from the definition of P_{CTC} that if $x \in L$ then every cyclic string must lead to an output of 1 in the last bit of \mathcal{R}_{CR} , while if $x \notin L$ then every cyclic string must lead to an output of 0. Hence the problem essentially reduces to finding a cyclic string.

But it is easy to find a cyclic string in polynomial space: the string $y^* := C'^{(2^{p(n)})}(y)$ will be cyclic for any y . The one remaining step is to compute $C(\langle y^*, 0^{q(n)} \rangle)$, and then output the last bit of \mathcal{R}_{CR} . ■

We are indebted to Lance Fortnow for the following lemma.

Lemma 2 $\text{PSPACE} \subseteq \text{P}_{\text{CTC}}$.

Proof. For some polynomial p , let M be a $p(n)$ -space Turing machine (i.e. every configuration of M takes $p(n)$ bits to describe). We can assume without loss of generality that M includes a “clock,” which is incremented at every time step, and which causes M to accept automatically once it reaches its maximum value. This prevents M from ever going into an infinite loop, regardless of its starting configuration.

Let m_1, \dots, m_T be the successive configurations of M when run on an input $x \in \{0, 1\}^n$. Then our task is to decide, using a CTC computer, whether m_T is an accepting or a rejecting configuration.

Our CTC algorithm \mathcal{A} will produce a circuit C that acts on two registers: a $(p(n) + 1)$ -bit CTC register \mathcal{R}_{CTC} , and a one-bit causality-respecting register \mathcal{R}_{CR} . For simplicity, we start by describing the induced circuit C' that acts on \mathcal{R}_{CTC} . Given a configuration m of M , let $S(m)$ be the successor of m : that is, the configuration obtained from m by incrementing the clock and performing one step of computation. Then the circuit C' acts as follows, on ordered pairs $\langle m, b \rangle$ consisting of a configuration m and a “control bit” b :

- If m is neither an accepting nor a rejecting configuration, then $C'(\langle m, b \rangle) = \langle S(m), b \rangle$.
- If m is an accepting configuration, then $C'(\langle m, b \rangle) = \langle m_1, 1 \rangle$.
- If m is a rejecting configuration, then $C'(\langle m, b \rangle) = \langle m_1, 0 \rangle$.

In other words, if m produces an output then C' sets the control bit to that output and goes back to the starting configuration; otherwise C' increments the computation and leaves the control bit unchanged (see Figure 2).

Now consider the graph of the function $C' : \{0, 1\}^{p(n)+1} \rightarrow \{0, 1\}^{p(n)+1}$. It is not hard to see that the only cycle in this graph is $(\langle m_1, 1 \rangle, \dots, \langle m_T, 1 \rangle)$ if m_T accepts, or $(\langle m_1, 0 \rangle, \dots, \langle m_T, 0 \rangle)$ if m_T rejects. Indeed, this is true even if there are configurations that are not reachable from $\langle m_1, 1 \rangle$ or $\langle m_1, 0 \rangle$, since those configurations will ultimately lead back to either $\langle m_1, 1 \rangle$ or $\langle m_1, 0 \rangle$ and therefore not produce new cycles. In other words, the only cycle is a loop over m_1, \dots, m_T , with the control bit b set to M 's final output. Therefore, the only probability distribution \mathcal{D}' over $\{0, 1\}^{p(n)+1}$ that is *stationary*, in the sense that $C'(\mathcal{D}') = \mathcal{D}'$, is the uniform distribution over $\langle m_1, b \rangle, \dots, \langle m_T, b \rangle$ where b is M 's final output.

Finally, the full circuit C simply applies C' to \mathcal{R}_{CTC} , and then copies the control bit into the causality-respecting register. ■

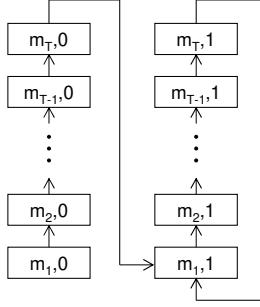


Figure 2: To simulate a PSPACE machine with a CTC, we perform a computation for which the only causally consistent evolution is a loop over all configurations of the machine, with a control bit b set to its final value (in this example $b = 1$).

4 The Quantum Case

Let G be a universal set of quantum gates, with amplitudes having rational real and imaginary parts. Then a *quantum CTC algorithm* \mathcal{A} is a deterministic polynomial-time algorithm that takes as input a string $x \in \{0, 1\}^n$, and that produces as output an encoding of a unitary quantum circuit $Q = Q_x$ with gates from G .

The circuit Q acts on two quantum registers: a $q(n)$ -qubit CTC register \mathcal{R}_{CTC} and an $r(n)$ -qubit causality-respecting register \mathcal{R}_{CR} . The causality-respecting register \mathcal{R}_{CR} is initialized to $|0\rangle^{\otimes r(n)}$, while the CTC register must be initialized to some $q(n)$ -qubit mixed state ρ that will ensure causal consistency. More formally, we require that

$$\text{Tr}_{\mathcal{R}_{CR}} \left(Q \left(\rho \otimes (|0\rangle\langle 0|)^{\otimes r(n)} \right) Q^\dagger \right) = \rho, \quad (1)$$

which is equivalent to ρ being a fixed-point of the quantum operation defined as

$$\Phi(\rho) := \text{Tr}_{\mathcal{R}_{CR}} \left(Q \left(\rho \otimes (|0\rangle\langle 0|)^{\otimes r(n)} \right) Q^\dagger \right).$$

Deutsch [9] proved that every such quantum operation *has* a fixed-point, and an alternate proof of this fact follows from our results in Section 4.3 below.

We can now define the complexity class BQP_{CTC} , of problems solvable using quantum computers with CTCs. Let \mathcal{M} be a measurement of the last qubit of \mathcal{R}_{CR} in the computational basis. Then we say the algorithm \mathcal{A} *accepts* x if for every mixed state ρ satisfying equation (1) above, $\mathcal{M} \left(Q \left(\rho \otimes (|0\rangle\langle 0|)^{\otimes r(n)} \right) Q^\dagger \right)$ results in output 1 with probability at least $2/3$. We say \mathcal{A} *rejects* x if for every ρ satisfying the equation, $\mathcal{M} \left(Q \left(\rho \otimes (|0\rangle\langle 0|)^{\otimes r(n)} \right) Q^\dagger \right)$ results in output 1 with probability at most $1/3$. We say \mathcal{A} decides the language $L \subseteq \{0, 1\}^*$ if \mathcal{A} accepts every input $x \in L$, and rejects every input $x \notin L$. Then BQP_{CTC} is the class of all languages L that are decided by some quantum CTC algorithm.

In what follows, we develop some needed background, and then prove the main result that $\text{BQP}_{CTC} \subseteq \text{PSPACE}$.

4.1 Matrix Representation of Superoperators

We will make use of a simple way of representing quantum operations as matrices. This representation begins with a representation of density matrices as vectors by the linear function defined on standard basis states as

$$\text{vec}(|x\rangle\langle y|) = |x\rangle\langle y|.$$

If ρ is an $N \times N$ density matrix, then $\text{vec}(\rho)$ is the N^2 -dimensional column vector obtained by stacking the rows of ρ on top of one another. For example,

$$\text{vec} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}.$$

Now, suppose that Φ is a given quantum operation acting on an N dimensional system, meaning that $\Phi : \mathbb{C}^{N \times N} \rightarrow \mathbb{C}^{N \times N}$ is linear, completely positive, and trace-preserving. Given that the effect of Φ on density matrices is linear, there must exist an $N^2 \times N^2$ matrix $K(\Phi)$ that satisfies

$$K(\Phi) \text{vec}(\rho) = \text{vec}(\Phi(\rho))$$

for every possible $N \times N$ density matrix ρ . The matrix $K(\Phi)$ is called the *natural matrix representation* of the quantum operation Φ , and is uniquely determined by Φ .

The natural matrix representation can easily be calculated from other standard forms. For example, if an operation Φ is represented in the usual Kraus form as

$$\Phi(\rho) = \sum_{j=1}^k A_j \rho A_j^\dagger,$$

then it holds that

$$\text{vec}(\Phi(\rho)) = \left(\sum_{j=1}^k A_j \otimes \overline{A_j} \right) \text{vec}(\rho),$$

and therefore

$$K(\Phi) := \sum_{j=1}^k A_j \otimes \overline{A_j}.$$

(Here $\overline{A_j}$ represents the entry-wise complex conjugate of A_j .)

In the section that follows, we will make use of the following simple way to calculate the natural matrix representation of a quantum operation that is specified by a quantum circuit. Suppose that \mathcal{R} is an r -qubit system, \mathcal{S} is an s -qubit system, and U is a unitary operation on $r+s$ qubits. Then for the quantum operation Φ defined as

$$\Phi(\rho) = \text{Tr}_{\mathcal{S}} \left[U (\rho \otimes (|0\rangle\langle 0|)^{\otimes s}) U^\dagger \right],$$

we have

$$K(\Phi) = M_1 (U \otimes \overline{U}) M_0 \tag{2}$$

for

$$M_1 = \sum_{y \in \{0,1\}^s} I \otimes \langle y | \otimes I \otimes \langle y | \quad \text{and} \quad M_0 = I \otimes |0\rangle^{\otimes s} \otimes I \otimes |0\rangle^{\otimes s}.$$

(In both cases, each identity matrix I acts on \mathcal{R} , or equivalently is the $2^r \times 2^r$ identity matrix.)

4.2 Space-Bounded and Depth-Bounded Computations

When we speak of a family $\{C_n : n \in \mathbb{N}\}$ of Boolean circuits, we assume that each C_n is an acyclic circuit, composed of AND, OR, NOT, and constant-sized fanout gates, with n input bits and an arbitrary number of output bits. Such a family computes a function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ if, for each $n \in \mathbb{N}$ and string $x \in \{0, 1\}^n$, the circuit C_n outputs $f(x)$ when given input x . The *depth* of a Boolean circuit C is the length of the longest path in C from an input bit to an output bit. The *size* of C is the sum of the number of input bits, output bits, and gates.

For a given function $s : \mathbb{N} \rightarrow \mathbb{N}$, we say that a Boolean circuit family $\{C_n : n \in \mathbb{N}\}$ is *space $O(s)$ -uniform* if there exists a deterministic Turing machine M that runs in space $O(s)$, and that outputs a description of C_n on input 1^n for each $n \in \mathbb{N}$. As is usual when discussing space-bounded computation, a deterministic Turing machine is assumed to be equipped with a read-only input tape that does not contribute to the space it uses, so it is meaningful to consider sublinear space bounds. Given a space $O(s)$ -uniform family $\{C_n : n \in \mathbb{N}\}$, the size of C_n can be at most $2^{O(s(n))}$.

We say a function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is in the class $\text{NC}(s)$ if there exists a space $O(s)$ -uniform family of Boolean circuits $\{C_n : n \in \mathbb{N}\}$ that computes f , and where the depth of C_n is at most $s(n)^{O(1)}$.² Also, a language L is in $\text{NC}(s)$ if its characteristic function is in $\text{NC}(s)$. We write NC for $\text{NC}(\log n)$, and $\text{NC}(\text{poly})$ for the union of $\text{NC}(n^c)$ over all constants c . Borodin [6] proved that if s satisfies $s(n) = \Omega(\log n)$, then every function in $\text{NC}(s)$ is computable by a deterministic Turing machine in space $s(n)^{O(1)}$. It follows that $\text{NC}(\text{poly}) \subseteq \text{PSPACE}$. (The reverse containment also holds, so in fact we have $\text{NC}(\text{poly}) = \text{PSPACE}$.)

It is clear that if $f \in \text{NC}(\text{poly})$ and $g \in \text{NC}$, then their composition $g \circ f$ is in $\text{NC}(\text{poly})$, since we can create a circuit for $g \circ f$ by composing the circuits for f and g in the obvious way.

Many functions of matrices are known to be computable in NC . These include sums and products of matrices, inverses, and the trace, determinant, and characteristic polynomial, all over a wide range of fields for which computations can be efficiently performed. (See von zur Gathen [10].) In particular, we will rely on a fact that follows from a result of Borodin, Cook, and Pippenger [7, Section 4]:

Theorem 3 ([7]) *The determinant of an $n \times n$ matrix whose entries are rational functions in an indeterminate z can be computed in NC .*

4.3 Projecting Onto Fixed Points

In this subsection, we prove a general theorem about efficient construction of quantum operations that project onto the fixed-points of other quantum operations. This theorem is the technical core of our $\text{BQP}_{\text{CTC}} \subseteq \text{PSPACE}$ result, but it might be of independent interest as well.

Theorem 4 *Suppose that $\Phi : \mathbb{C}^{N \times N} \rightarrow \mathbb{C}^{N \times N}$ is a given quantum operation acting on an N -dimensional system, meaning that it is a completely positive and trace-preserving linear map. We will prove that there exists another quantum operation*

$$\Lambda : \mathbb{C}^{N \times N} \rightarrow \mathbb{C}^{N \times N}$$

that satisfies three properties:

² NC stands for “Nick’s Class”; the term is historical. Also, what we call $\text{NC}(s)$ is called $\text{NC}(2^s)$ by Borodin, Cook, and Pippenger [7].

- (1) For every density matrix $\sigma \in \mathbb{C}^{N \times N}$, it holds that $\rho = \Lambda(\sigma)$ is a fixed-point of Φ .
- (2) Every density matrix ρ that is a fixed-point of Φ is also a fixed point of Λ .
- (3) Λ can be computed from Φ in NC.

In essence, Λ is a (non-orthogonal) projection onto fixed-points of Φ , so if we want a fixed-point of Φ it suffices to compute $\Lambda(\sigma)$ for any density matrix σ , and moreover every fixed-point ρ of Φ arises in this way from some density matrix σ (which always includes the choice $\sigma = \rho$).

Proof. The operation Λ is defined as follows. First, for each real number $z \in (0, 1)$, we define a superoperator $\Lambda_z : \mathbb{C}^{N \times N} \rightarrow \mathbb{C}^{N \times N}$ as

$$\Lambda_z = z \sum_{k=0}^{\infty} (1-z)^k \Phi^k.$$

Here Φ^k represents the k -fold composition of Φ and Φ^0 is the identity operation. Each Φ^k is obviously completely positive and trace-preserving. Given that $z(1-z)^k \in (0, 1)$ for each choice of $z \in (0, 1)$ and $k \geq 0$, and that $\sum_{k=0}^{\infty} z(1-z)^k = 1$ for every $z \in (0, 1)$, we have that Λ_z is a convex combination of completely positive and trace-preserving maps. Thus, Λ_z is completely positive and trace-preserving as well. Finally, we take

$$\Lambda = \lim_{z \downarrow 0} \Lambda_z.$$

We must of course prove that this limit exists—and in the process, we will prove that Λ can be produced from Φ by an NC computation, which is an important ingredient of our simulation of BQP_{CTC} in PSPACE. Once this is done, the required properties of Λ will be easily verified.

We assume that Φ is represented by the $N^2 \times N^2$ complex matrix $M = K(\Phi)$ as discussed in Section 4.1. Since Φ is a quantum operation, every eigenvalue of M lies within the unit circle.³ It follows that the matrix $I - (1-z)M$ is invertible for every real $z \in (0, 1)$, and moreover there is a convergent series for its inverse:

$$(I - (1-z)M)^{-1} = I + (1-z)M + (1-z)^2 M^2 + \dots \quad (3)$$

Now, for every $z \in (0, 1)$ we define an $N^2 \times N^2$ matrix R_z as follows:

$$R_z := z(I - (1-z)M)^{-1}.$$

We note that $R_z = K(\Lambda_z)$ for Λ_z as defined above—and as each Λ_z is completely positive and trace-preserving, each entry of R_z must be bounded in absolute value by 1.

Next, by Cramer's rule, we have

$$z(I - (1-z)M)^{-1}[i,j] = (-1)^{i+j} \frac{z \det((I - (1-z)M)_{j,i})}{\det(I - (1-z)M)}, \quad (4)$$

where $(I - (1-z)M)_{j,i}$ denotes the $(N^2 - 1) \times (N^2 - 1)$ matrix obtained by removing the j^{th} row and i^{th} column from $I - (1-z)M$. It follows that each entry of R_z is given by a rational function

³This fact is proved in [16]. An alternate proof follows from the fact that $\|\Phi\|_{\diamond} = 1$ and that the diamond norm is submultiplicative (see Theorem 5.6.9 of [12]).

in the variable z having degree at most N^2 . As the entries of R_z are rational functions that are bounded for all $z \in (0, 1)$, we have that the limit $\lim_{z \downarrow 0} R_z$ exists. Define

$$R := \lim_{z \downarrow 0} R_z,$$

and note that $R = K(\Lambda)$. We have therefore proved that the limit $\Lambda = \lim_{z \downarrow 0} \Lambda_z$ exists as claimed.

The fact that R can be computed from M in NC follows from the above discussion, together with Theorem 3. In particular, equation (4) above expresses the entries of R_z as ratios of polynomials of degree at most N^2 in z having coefficients with rational real and imaginary parts. It remains to compute the limit, which is also done symbolically for the real and imaginary parts of each entry. To compute

$$\lim_{z \downarrow 0} \frac{f(z)}{g(z)}$$

for polynomials $f(z) = \sum_i c_i z^i$ and $g(z) = \sum_i d_i z^i$, we perform a binary search on the coefficients of g to find the smallest k for which $d_k \neq 0$, and then output the ratio c_k/d_k . Each of the required computations can be done in NC , and can be applied in parallel for each entry of R to allow R to be computed from M in NC .

Finally, we verify the required properties of Λ . It is clear that every fixed-point ρ of Φ is also a fixed-point of Λ , since

$$\Lambda_z(\rho) = z \sum_{k=0}^{\infty} (1-z)^k \Phi^k(\rho) = z \sum_{k=0}^{\infty} (1-z)^k \rho = \rho,$$

and therefore $\Lambda(\rho) = \lim_{z \downarrow 0} \Lambda_z(\rho) = \rho$. To prove that $\rho = \Lambda(\sigma)$ is a fixed-point of Φ for every density matrix σ , it suffices to prove $\Phi\Lambda = \Lambda$. For each $z \in (0, 1)$ we have

$$\Phi\Lambda_z = z \sum_{k=0}^{\infty} (1-z)^k \Phi^{k+1} = \frac{z}{1-z} \sum_{k=1}^{\infty} (1-z)^k \Phi^k = \frac{1}{1-z} \Lambda_z - \frac{z}{1-z} I,$$

and therefore

$$\Phi\Lambda = \lim_{z \downarrow 0} \Phi\Lambda_z = \lim_{z \downarrow 0} \left(\frac{1}{1-z} \Lambda_z - \frac{z}{1-z} I \right) = \Lambda$$

as claimed. ■

4.4 Proof of Containment

We can now complete the proof that quantum computers with CTCs are simulable in PSPACE .

Theorem 5 $\text{BQP}_{\text{CTC}} \subseteq \text{PSPACE}$.

Proof. Let $L \in \text{BQP}_{\text{CTC}}$ be given, and assume that \mathcal{A} is a quantum CTC algorithm for L . As discussed in Section 4.2, it suffices to prove $L \in \text{NC}(\text{poly})$.

Assume for simplicity that an input x of length n has been fixed. Let Q be the unitary quantum circuit that is output by \mathcal{A} on input x ; then as in the definition of BQP_{CTC} , define a quantum operation

$$\Phi(\rho) := \text{Tr}_{\mathcal{R}_{CR}} \left(Q \left(\rho \otimes (|0\rangle\langle 0|)^{\otimes r(n)} \right) Q^\dagger \right).$$

Our goal will be to compute the probability

$$\Pr \left[\mathcal{M} \left(Q \left(\rho \otimes (|0\rangle\langle 0|)^{\otimes r(n)} \right) Q^\dagger \right) = 1 \right] \quad (5)$$

for some arbitrary fixed-point ρ of Φ . This value can then be compared to $1/2$ to decide whether to accept or reject. This computation will be performed in a uniform manner, in $\text{NC}(\text{poly})$, therefore establishing that $L \in \text{NC}(\text{poly})$.

The first step is to compute the matrix representation $M = K(\Phi)$ of the operation Φ . This can be done by a polynomial-space uniform family of Boolean circuits with exponential size and polynomial depth, since M is expressible as in equation (2), and Q is expressible as a product of a polynomial number of exponential-size matrices determined by the gates of Q .

Next we compute the matrix representation $R = K(\Lambda)$, where Λ is the quantum operation that projects onto fixed-points of Φ discussed in Section 4.3. We have argued that R can be computed from M in NC , and therefore by composing this computation with the $\text{NC}(\text{poly})$ computation of M , we have that R can be computed in $\text{NC}(\text{poly})$.

Finally, we compute a fixed-point ρ of Φ using R along with an arbitrary choice of a density matrix input for Λ . For instance, we may take $\rho = \Lambda \left((|0\rangle\langle 0|)^{\otimes q(n)} \right)$, so that $\text{vec}(\rho) = R |0\rangle^{\otimes 2q(n)}$. The probability (5) can then be evaluated in $\text{NC}(\text{poly})$ by performing matrix-vector multiplication.

■

5 Dealing With Error

Recall that, in defining the class BQP_{CTC} , we required the quantum circuits to involve amplitudes with rational real and imaginary parts. However, while this assumption is mathematically convenient, it is also “unphysical.” Even in a CTC universe, quantum operations can presumably only be implemented to finite precision.

The trouble is that, in a CTC universe, two quantum operations that are arbitrarily close can produce detectably different outcomes. As an example, consider the stochastic matrices

$$\begin{pmatrix} 1 & \varepsilon \\ 0 & 1 - \varepsilon \end{pmatrix}, \begin{pmatrix} 1 - \varepsilon & 0 \\ \varepsilon & 1 \end{pmatrix}.$$

As $\varepsilon \rightarrow 0$, these matrices become arbitrarily close to each other and to the identity. Yet their fixed-points remain disjoint: the first has a unique fixed-point of $(1, 0)^T$, while the second has a unique fixed-point of $(0, 1)^T$. Hence, were an algorithm to apply one of these matrices inside a CTC, an arbitrarily small error could completely change the outcome of the computation.

We will show that, while this “pathological” situation can arise in principle, it does not arise in our simulation of PSPACE by a CTC computer in Lemma 2.

Call ρ an ε -fixed-point of Φ if $\|\rho - \Phi(\rho)\|_{\text{tr}} \leq \varepsilon$.

Proposition 6 *Suppose ρ is a fixed-point of Φ and $\|\Phi - \Phi'\|_\diamond \leq \varepsilon$. Then ρ is an ε -fixed-point of Φ' .*

Proof. Since $\rho = \Phi(\rho)$, we have $\|\rho - \Phi'(\rho)\|_{\text{tr}} = \|\Phi(\rho) - \Phi'(\rho)\|_{\text{tr}} \leq \varepsilon$. ■

Lemma 7 *Let Φ be a classical operation mapping a finite set \mathcal{B} to itself, and let ρ be an ε -fixed-point of Φ . Then $\|\rho - \sigma\|_{\text{tr}} \leq 2|\mathcal{B}|\varepsilon$ for some state σ supported only on the cycles of Φ .*

Proof. We prove the contrapositive. Let \mathcal{C} be the union of all cycles of Φ , and let $\overline{\mathcal{C}} = \mathcal{B} \setminus \mathcal{C}$. Also, for each element $x \in \mathcal{B}$, let $p_x = \langle x | \rho | x \rangle$. Suppose ρ is not δ -close to any state supported only on \mathcal{C} , where $\delta = 2|\mathcal{B}|\varepsilon$. Then $\sum_{x \in \overline{\mathcal{C}}} p_x > \delta$. Hence

$$\begin{aligned} \|\rho - \Phi(\rho)\|_{\text{tr}} &\geq \frac{1}{2} \sum_{x \in \overline{\mathcal{C}}} |p_x - p_{\Phi(x)}| \\ &\geq \frac{1}{2} \max_{x \in \overline{\mathcal{C}}} p_x \\ &> \frac{1}{2} \cdot \frac{\delta}{|\overline{\mathcal{C}}|} \\ &\geq \varepsilon. \end{aligned}$$

■

Let C' be the circuit from Lemma 2 that maps \mathcal{R}_{CTC} to itself. As part of the proof of Lemma 2, we showed that the graph of $C' : \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^{p(n)}$ has a unique cycle L , in which every configuration leads to the desired output. Now let C'' be a corrupted version of C' that satisfies $\|C' - C''\|_{\diamond} \leq \varepsilon$, and let ρ be any fixed-point of C'' . Then ρ is an ε -fixed-point of C' by Proposition 6. By Lemma 7, this in turn means that $\|\rho - \sigma\|_{\text{tr}} \leq 2^{p(n)+1}\varepsilon$ for some state σ supported only on L . So provided $\varepsilon \ll 2^{-p(n)-1}$, a CTC algorithm that uses C'' in place of C' will still produce the correct answer with high probability.

Moreover, as pointed out by Bacon [5], even if every gate in our quantum circuit is subject to constant error, we can still use standard results from the theory of quantum error-correction [3] to ensure that $\|C' - C''\|_{\diamond}$ is exponentially small, where C' and C'' are the quantum circuits acting on the *logical* (encoded) qubits.

One might also ask whether our proof of $\text{BQP}_{CTC} \subseteq \text{PSPACE}$ in Section 4 is affected by precision issues. However, the key point there is that, since the amplitudes are assumed to be complex rational numbers, we are able to use the algorithm of Borodin, Cook, and Pippenger [7] to compute a fixed-point *symbolically* rather than just numerically.

6 Discussion and Open Problems

6.1 CTCs in Other Computational Models

In the proof that $\text{PSPACE} \subseteq \text{P}_{CTC}$, we did not actually need the full strength of polynomial-time computation inside the CTC: rather, all we needed was the ability to update the configuration of a PSPACE machine and increment a counter. Thus, our proof also shows (for example) that $\text{PSPACE} = \text{AC}_{CTC}^0$, where AC^0 denotes the class of problems solvable by constant-depth, polynomial-size circuits consisting of AND, OR, and NOT gates, and AC_{CTC}^0 is defined the same way as P_{CTC} but with AC^0 circuits instead of arbitrary polynomial-size circuits.

In the other direction, we could also define PSPACE_{CTC} the same way as P_{CTC} , but with PSPACE machines in place of polynomial-size circuits. Then it is evident that our proof generalizes to show $\text{PSPACE}_{CTC} = \text{PSPACE}$.

It would be extremely interesting to study the consequences of Deutsch's causal consistency assumption in other settings besides polynomial-time computation: for example, communication complexity, branching programs, and finite automata.

6.2 Narrow CTCs

What is the power of classical CTCs with a single bit, or of quantum CTCs with a single qubit (as studied by Bacon [5])? Let $\text{P}_{\text{CTC}1}$, $\text{BPP}_{\text{CTC}1}$, and $\text{BQP}_{\text{CTC}1}$ be the corresponding deterministic, randomized, and quantum complexity classes. Then it is not hard to show that $\text{NP} \cap \text{coNP} \subseteq \text{BPP}_{\text{CTC}1}$: that is, a single use of a one-bit CTC is enough to solve all problems in the class $\text{NP} \cap \text{coNP}$. For we can guess a random string $w \in \{0, 1\}^{p(n)}$, then set the CTC bit b to 1 if w is a yes-witness or to 0 if w is a no-witness, and leave b unchanged if w is neither. If there exists a yes-witness but not a no-witness, then the only fixed-point of the induced stochastic evolution is $b = 1$, while if there exists a no-witness but not a yes-witness, then the only fixed-point is $b = 0$. Indeed, a simple extension of this idea yields $\text{NP} \subseteq \text{BPP}_{\text{CTC}1}$: we set $b = 1$ if a yes-witness $w \in \{0, 1\}^{p(n)}$ was guessed, and set $b = 0$ with some tiny probability $\varepsilon \ll 2^{-p(n)}$ independent of the witness. Again, the unique fixed-point of the induced stochastic map will fix $b = 1$ with overwhelming probability if there exists a yes-witness, or $b = 0$ with certainty if not. Fully understanding the power of “bounded-width CTCs” remains a problem for the future.

6.3 CTC Computers With Advice

Let $\text{BPP}_{\text{CTC}/\text{rpoly}}$ be defined the same way as BPP_{CTC} , except that instead of being initialized to $0^{q(n)}$, the chronology-respecting register \mathcal{R}_{CR} is initialized to a probability distribution \mathcal{D}_n which depends only on the input length n , but can otherwise be chosen arbitrarily to help the CTC algorithm. Then we claim that $\text{BPP}_{\text{CTC}/\text{rpoly}} = \text{ALL}$: in other words, $\text{BPP}_{\text{CTC}/\text{rpoly}}$ contains *every* computational problem! To see this, let \mathcal{R}_{CR} be initialized to the uniform distribution over all ordered pairs $\langle z, f(z) \rangle$, where z is an n -bit input and $f(x) \in \{0, 1\}$ encodes whether $x \in L$. Also, let the CTC register \mathcal{R}_{CTC} contain a single bit b . Then given an input x , our circuit C acts on b as follows: if $z = x$ then C sets $b = f(x)$; otherwise C leaves b unchanged. It is easy to see that the unique fixed-point of the induced stochastic map on \mathcal{R}_{CTC} fixes $b = f(x)$ with certainty.

While it demonstrates that CTCs combined with randomized advice yield staggering computational power, this result is not *quite* as surprising as it seems: for it was previously shown by Aaronson [2] that $\text{PP}/\text{rpoly} = \text{ALL}$, and by Raz [14] that $\text{IP}(2)/\text{rpoly} = \text{ALL}$. In other words, randomized advice has a well-known tendency to yield unlimited computational power when combined with certain other resources.

7 Acknowledgments

We thank Lance Fortnow for suggesting the proof of Lemma 2, and Dave Bacon for helpful discussions.

References

- [1] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, March 2005. quant-ph/0502072.
- [2] S. Aaronson. QMA/qpoly is contained in PSPACE/poly: de-Merlinizing quantum protocols. In *Proc. IEEE Conference on Computational Complexity*, pages 261–273, 2006. quant-ph/0510230.

- [3] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proc. ACM STOC*, pages 176–188, 1997. quant-ph/9906129.
- [4] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. 2008. To be published. Online draft at <http://www.cs.princeton.edu/theory/complexity/>.
- [5] D. Bacon. Quantum computational complexity in the presence of closed timelike curves. *Phys. Rev. A*, 70(032309), 2004. quant-ph/0309189.
- [6] A. Borodin. On relating time and space to size and depth. *SIAM J. Comput.*, 6(4):733–744, 1977.
- [7] A. Borodin, S. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58(1-3):113–136, 1983.
- [8] T. Brun. Computers with closed timelike curves can solve hard problems. *Foundations of Physics Letters*, 16:245–253, 2003. gr-qc/0209061.
- [9] D. Deutsch. Quantum mechanics near closed timelike lines. *Phys. Rev. D*, 44:3197–3217, 1991.
- [10] J. von zur Gathen. Parallel linear algebra. In J. Reif, editor, *Synthesis of Parallel Algorithms*, chapter 13. Morgan Kaufmann Publishers, 1993.
- [11] K. Gödel. An example of a new type of cosmological solution of Einstein’s field equations of gravitation. *Rev. Mod. Phys.*, 21:447, 1949.
- [12] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
- [13] M. S. Morris, K. S. Thorne, and U. Yurtsever. Wormholes, time machines, and the weak energy condition. *Phys. Rev. Lett.*, 61:1446–1449, 1988.
- [14] R. Raz. Quantum information and the PCP theorem. In *Proc. IEEE FOCS*, 2005. quant-ph/0504075.
- [15] W. J. van Stockum. The gravitational field of a distribution of particles rotating around an axis of symmetry. *Proc. Roy. Soc. Edinburgh A*, 57:135, 1937.
- [16] B. Terhal and D. DiVincenzo. On the problem of equilibration and the computation of correlation functions on a quantum computer. *Phys. Rev. A*, 61:022301, 2000. quant-ph/9810063.
- [17] J. Watrous. Space-bounded quantum complexity. *J. Comput. Sys. Sci.*, 59(2):281–326, 1999.
- [18] J. Watrous. Quantum computational complexity. In *Encyclopedia of Complexity and Systems Science*. Springer, 2009. To appear. arXiv:0804.3401.