# Computability Theory of Closed Timelike Curves

Scott Aaronson[*] Mohammad Bavarian[†] Giulio Gueltrini[‡]

**Abstract**

We ask, and answer, the question of what's computable by Turing machines equipped with time travel into the past: that is, closed timelike curves or CTCs (with no bound on their size). We focus on a model for CTCs due to Deutsch, which imposes a probabilistic consistency condition to avoid grandfather paradoxes. Our main result is that computers with CTCs can solve exactly the problems that are Turing-reducible to the halting problem, and that this is true whether we consider classical or quantum computers. Previous work, by Aaronson and Watrous, studied CTC computers with a polynomial size restriction, and showed that they solve exactly the problems in $\mathsf{PSPACE}$, again in both the classical and quantum cases.

Compared to the complexity setting, the main novelty of the computability setting is that not all CTCs have fixed-points, even probabilistically. Despite this, we show that the CTCs that *do* have fixed-points suffice to solve the halting problem, by considering fixed-point distributions involving infinite geometric series. The tricky part is to show that even quantum computers with CTCs can be simulated using a HALT oracle. For that, we need the Riesz representation theorem from functional analysis, among other tools.

We also study an alternative model of CTCs, due to Lloyd et al., which uses postselection to "simulate" a consistency condition, and which yields $\mathsf{BPP_{path}}$ in the classical case or $\mathsf{PP}$ in the quantum case when subject to a polynomial size restriction. With no size limit, we show that postselected CTCs yield only the computable languages if we impose a certain finiteness condition, or all languages nonadaptively reducible to the halting problem if we don't.

## 1 Introduction

How would time travel to the past change the theory of computation? Many people's first thought is that, in a world with closed timelike curves (CTCs),[1] one could simply do an arbitrarily long computation, then "send the answer back in time to before the computer started." Or one could do a few steps of a computation, then "set back the clock" and do a few more steps, and so on, reusing the same time interval as often as one liked.

[*]University of Texas at Austin. Email: aaronson@cs.utexas.edu. Supported by a Vannevar Bush Fellowship (formerly NSSEFF grant) from the US Department of Defense, and by the Simons Foundation's It from Qubit Collaboration.

[†]MIT. Email: bavarian@mit.edu. Supported by NSF grant CCF-1420692.

[‡]MIT. Email: gueltro@mit.edu.

[1]As its name suggests, a CTC is a cycle in the spacetime manifold that's locally timelike, and that can therefore carry a particle to its own past. For the sake of completeness, we mention that whether CTCs can exist, and if not why not, is one of the great unsolved problems of theoretical physics. CTCs can formally appear in solutions to Einstein's field equations of general relativity; but creating CTCs in a spacetime that initially lacks them seems to require exotic conditions (such as regions of negative energy density). In any case, a definitive answer as to why CTCs can or can't exist would require a quantum theory of gravity.

The problem with both of these ideas is that they implicitly treat time like just an additional dimension of space. Indeed, they make sense only if we posit a second "meta-time," which governs a computer's evolution through the "original" time dimension. A different way to see the difficulty is that these ideas fail to grapple with the basic conceptual difficulty of CTCs, namely the *Grandfather Paradox*. This is the scenario where a person travels back in time to kill her grandfather and thereby prevent her own birth—but because she succeeds at this mission, there's "no longer"[2] anyone to go back in time and kill her grandfather, so she *is* born after all, so there *is* someone to kill her grandfather, etc.—a logical contradiction. In the present context, think of a computer in a CTC that stores a single bit $b$, and sends back in time the instruction $b := \text{not}\,(b)$.

Clearly, even to discuss such scenarios, we need a formal model that specifies how CTCs behave. In his seminal 1991 paper "Quantum mechanics near closed timelike lines," David Deutsch [12] proposed just such a model, and thereby set the stage for much of the modern discussion about CTCs. Deutsch's insight was that, by enlarging the set of *states* we're willing to consider—from deterministic states to probabilistic or quantum states—we can ensure that every finite-dimensional evolution operator has at least one fixed-point. In other words, every finite-dimensional Markov chain has at least one stationary distribution; and Deutsch likewise observed that every finite-dimensional quantum operation (not necessarily unitary) has at least one stationary mixed state. And thus, he said, we can resolve the usual paradoxes of time travel by simply positing that Nature has to find a fixed-point of whatever operator acts around the CTC. So for example, the "resolution" of the Grandfather Paradox, on this account, is that the time-traveler is born with probability 1/2, and *if* she's born, then she travels back in time to kill her grandfather; therefore she's born with probability 1/2, and so on.

Yet as Deutsch [12] pointed out, even if we accept that this fixed-point proposal "solves the Grandfather Paradox," it gives rise to a second "paradox" of its own. We can illustrate this informally, with the story of the $21^{st}$-century time-traveler who goes back to Elizabethan England and dictates Shakespeare's plays to him. Shakespeare thanks the time-traveler for saving him so much effort and writes down the plays exactly as dictated, whereupon the plays are passed down to the $21^{st}$ century and sent back in time, etc. Note that, in contrast to Grandfather Paradox, here there's no logical contradiction: we've described a fully consistent evolution, i.e. a fixed-point. What's "paradoxical" about this scenario, we might say, is simply that somehow *Hamlet* appeared in the universe, but without anyone having labored to write it, or any other visible process having created the play.

We can translate this "Shakespeare Paradox" into the framework of computational complexity. Thus, imagine a CTC computer that takes as input a string $y \in \{0,1\}^n$, and that checks to see whether (let's say) $y$ satisfies a 3SAT instance $\phi$. If $y$ satisfies $\phi$, then the computer's output, which it sends back in time to its own input port, is simply $y$. If, on the other hand, $y$ doesn't satisfy $\phi$, then the output is $(y+1) \bmod 2^n$: that is, the lexicographically-next $n$-bit string, looping back to the beginning when the last string is reached.

Provided $\phi$ is satisfiable, it's easy to see that all the fixed-points of the above evolution are concentrated on satisfying assignments. In other words, the only logically consistent story is that a satisfying assignment "appears out of the blue," is checked and found to be valid, and is therefore sent back in time to appear again. Meanwhile, if $\phi$ is unsatisfiable, then the unique fixed-point is the uniform distribution over $\{0,1\}^n$. Our conclusion is that, in a universe with

[2] "No longer" isn't quite the right phrase, but will have to serve. It's common when discussing CTCs informally to exhaust the tense resources of the English language.

fully-programmable Deutschian CTCs, *it would be possible to solve* $\mathsf{NP}$*-complete problems using only "polynomial resources"* (by which we mean: a CTC that acts with a polynomial-size circuit on a polynomial number of bits).

The above reasoning was implicit in Deutsch's original paper. But it left open the question of what exactly one could and couldn't do using a CTC computer. That question was taken up in subsequent papers, which we'll now discuss, before how explaining how the present paper uncovers and resolves a major aspect of the question that had been overlooked.

## 1.1 Related Work

**The Complexity Theory of Deutschian CTCs.** In 2005, Aaronson [2] defined the complexity classes $\mathsf{P}_{\mathsf{CTC}}$ and $\mathsf{BQP}_{\mathsf{CTC}}$, to capture the decision problems that are solvable with polynomial resources, using a classical or quantum computer respectively that's enhanced by a Deutschian CTC. With a contribution from Lance Fortnow, Aaronson showed that $\mathsf{P}_{\mathsf{CTC}} = \mathsf{PSPACE}$, and also that $\mathsf{PSPACE} \subseteq \mathsf{BQP}_{\mathsf{CTC}} \subseteq \mathsf{EXP}$. In other words, CTC computers can do even more than $\mathsf{NP}$, but at any rate no more than $\mathsf{EXP}$. This left the problem of pinning down $\mathsf{BQP}_{\mathsf{CTC}}$ more precisely.

In 2008, Aaronson and Watrous [5] solved that problem by showing that $\mathsf{BQP}_{\mathsf{CTC}} = \mathsf{PSPACE}$. The technical core of their result was a new polynomial-space algorithm to find fixed-points of quantum operations that act on $n^{O(1)}$ qubits (or rather, to compute properties of those fixed-points). This, in turn, required some structural results about *superoperators*, which are the most general formalism for quantum channels (not necessarily unitary). It also required parallel linear-algebra algorithms, such as those of Borodin, Cook, and Pippenger [11], applied to exponentially-large matrices whose entries are limits of rational functions.

The result of Aaronson and Watrous [5] had at least two interesting implications. First, it said that once we have a Deutschian CTC, we get no *additional* advantage from using a quantum computer rather than a classical one.[3] Second, it said that the effect of CTCs on complexity theory is ultimately to "make time equivalent to space as a computational resource"—i.e., exactly what one might have guessed from the naïve intuition with which we opened this paper, but for a much less obvious reason.

Over the last decade, various authors have questioned the assumptions behind Deutsch's model of CTCs, and/or proposed alternative CTC models and studied *their* effects on computational complexity. We'll mention the works most relevant to this paper.

**Decomposition Uniqueness.** In 2009, Bennett et al. [9] published a critique of Deutsch's CTC model. Their central objection was that Deutschian CTCs violate the statistical interpretation of quantum mixed states. In other words, if a mixed state $\rho$ can be decomposed as $\sum_i p_i \left|\psi_i\right\rangle\left\langle\psi_i\right|$, then normally we interpret $\rho$ as an ensemble where the pure state $\left|\psi_i\right\rangle$ occurs with probability $p_i$. However, if $S$ is an evolution operation produced using a Deutschian CTC, then it could easily happen that

$$S\left(\rho\right) \neq \sum_i p_i S\left(\left|\psi_i\right\rangle\left\langle\psi_i\right|\right).$$

In Bennett et al.'s view, this prevents us from interpreting (say) a CTC computer that receives an input $x$ drawn from a probability distribution, and that then produces a corresponding distribution

[3]Yet, even though Deutschian CTCs would "render quantum computing useless," it seems unlikely that the skeptics of quantum computing would be satisfied with this outcome!

over outputs. However, Bennett et al.'s proposed remedy for this problem was, in our view, extremely drastic. They held that CTCs should be allowed only if they receive no input from any other part of the universe—in which case, each CTC could simply be replaced by a static "quantum advice state" [1], and would barely seem to qualify as a CTC at all.

One response to Bennett et al. is to observe that the same difficulty arises, not only with Deutschian CTCs, but with essentially *any* model of CTCs—and more broadly, with any essentially modification to the rules of quantum mechanics or probability theory that has the effect of making them nonlinear. In other words, this "decomposition dependence" seems less like a fixable technical issue, than simply like an inevitable byproduct of wanting to discuss a world with CTCs in the first place. Fortunately, the issue doesn't prevent us from formulating rich and well-defined models of computation (i.e., complexity classes) involving CTCs.

**Postselected CTCs.** In 2011, Lloyd et al. [13] proposed an alternative to Deutsch's CTC model, based on what they called "postselected teleportation." The basic idea here is to start the CTC qubits in some quantum state $|\psi\rangle$, then do a computation on the qubits, and finally use a projective measurement to *postselect* on the qubits being returned to the state $|\psi\rangle$ at the end. Two drawbacks of this model are

(i) that the results could depend on the choice of initial state $|\psi\rangle$, and

(ii) that this model doesn't "resolve the Grandfather Paradox" in the sense that Deutsch's does. (For the outcome is undefined if the final measurement returns $|\psi\rangle$ with probability 0, so we need to add a condition by hand saying that this doesn't happen.)

Despite these drawbacks, Lloyd et al. [13] presented it as a selling point for their model that it "ameliorates" the computational complexity situation, if only slightly: instead of $\mathsf{PSPACE}$ as with Deutsch's model, the postselected CTC model yields "merely" the complexity class $\mathsf{PostBQP} = \mathsf{PP}$ in the quantum case, or $\mathsf{PostBPP} = \mathsf{BPP}_{\mathsf{path}}$ in the classical case.

**Narrow CTCs.** A different variation that's been studied involves Deutschian CTCs that are restricted to sending just one bit or qubit back in time. This line of work was initiated in 2003 by Bacon [8], who showed that repeated use of 1-bit Deutschian CTCs is already enough to solve $\mathsf{NP}$-complete problems in polynomial time. Subsequently, Say and Yakaryılmaz [16] showed that 1-bit Deutschian CTCs give precisely the power of $\mathsf{PP}$ in the quantum case, and of $\mathsf{BPP}_{\mathsf{path}}$ in the classical case.[4] Next O'Donnell and Say [14] showed that, in the classical case, the power remains $\mathsf{BPP}_{\mathsf{path}}$ (rather than $\mathsf{PSPACE}$) even if the Deutschian CTC has as many as $O(\log n)$ qubits. In a recent manuscript [15], O'Donnell and Say have extended this to show that even the quantum case, the power remains $\mathsf{PP}$ even if the Deutschian CTC has $O(\log n)$ qubits.

**Spurious Fixed-Points.** Finally, we should mention that the computational power of Deutschian CTCs has been questioned, on the ground that Nature might find "spurious fixed-points" of the physical evolution around a CTC that fail to respect the computational evolution (for example, a fixed-point in which the computer never turns on). For more see Deutsch [12] or Aaronson [3, Section 10] for example. In general, whether spurious fixed-points exist might depend on the detailed laws of physics, and the amount of control they allow over the microscopic state inside the CTC—an issue to which we'll return in Section 9.1.

[4]That paper also gave a model—involving the ability to send 1 bit back in time, $O(1)$ steps into the past, an unlimited number of times—that allowed all decidable languages to be decided in only $O(1)$ time.

## 1.2 Our Results

In this work, for the first time, we take up the question of whether and how CTCs affect *computability* theory.[5] We know that computability is insensitive to many distinctions that matter enormously in complexity theory: for example, the distinctions among deterministic, nondeterministic, randomized, and quantum Turing machines. On the other hand, Aaronson and Watrous's upper bound of $\mathsf{PSPACE}$ on the power of CTCs [5] crucially relied on all the computation inside the CTC being polynomially bounded. So what happens if we relax that requirement, and allow CTC computations on strings of arbitrary lengths (or on distributions or superpositions over strings of different lengths)? Could the ability to find fixed-points of such computations let us, for example, solve the halting problem in finite time?

In considering this question, the first difficulty is that, unlike with the finite case, transformations on an infinite set need not *have* fixed-points—not even probabilistic ones. (In other words, infinite Markov chains can lack stationary distributions.) To illustrate, consider a computation $S$ that takes as input a natural number $n$, and that outputs $n+1$. In the finite case, such a computation would eventually need to "wrap around" to $n = 0$, in which case the uniform distribution over $n$ would be a fixed-point. In the infinite case, by contrast, there must be a least natural number $n$ on which a distribution $\mathcal{D}$ has support, but then $S(\mathcal{D})$ has no support on $n$ so is different from $\mathcal{D}$.

However, a simple fix for that problem is to consider only CTC computations that *do* have fixed-points. We stress that, apart from the existence of a fixed-point now necessarily being an assumption rather than a theorem, our model of CTC computation is identical to the model studied previously by Aaronson and Watrous [5] (or rather, is its obvious computability analogue). In particular, we allow probabilistic fixed-points in both the complexity and the computability cases.

We can now ask, for example: given a probabilistic computation $S$ mapping $\{0,1\}^*$ to $\{0,1\}^*$, and promised that

(a) $S$ has at least one probabilistic fixed-point, and

(b) either all of $S$'s fixed-points lead to some other computation $C$ accepting with high probability, or all of them lead to $C$ rejecting with high probability,

how hard is it to distinguish the accept case from the reject case?

Our first result, in Section 4, says that it's hard indeed: *a Turing machine with a Deutschian CTC can solve the halting problem in finite time.* In fact, with just a single use of a Deutschian CTC, a Turing machine can solve any problem that's Turing-reducible to the halting problem.

Given a Turing machine $\langle P\rangle$ for which we want to decide halting, the idea here is to construct a CTC computation $S$ that takes as input an alleged transcript of the first $t$ steps of $P$'s execution, where $t$ could be any positive integer. If the transcript is invalid, then $S$ returns the first step of $P$'s execution; if the transcript shows $P$ halting, then $S$ returns the transcript unmodified; and finally, if the transcript is valid but doesn't show $P$ halting, then $S$ returns the first step *or* the first $t+1$ steps of $P$'s execution with equal probabilities. This sets up a situation where, if $P$ halts, then the unique fixed-point of $S$ is a transcript of $P$ halting, while if $P$ doesn't halt, then

[5]Even though this paper studies computability theory, it's *computability theory directly inspired by complexity theory* (in an ironic reversal of the usual historical path). For that reason, we hope that the paper can be considered in scope for the CCC conference.

the unique fixed-point is a geometric distribution, showing the first $t$ steps of $P$'s execution with probability $2^{-t}$ for all $t > 0$.

Of course, if $P$ halts, then this procedure gives us no computable upper bound on how *long* the certificate of its halting might be. In that one respect, we haven't gained over the trivial solution of just simulating $P$ until it possibly halts. Where we did gain, though, is that we now have a finite-sized certificate for $P$'s running forever, not only for $P$'s halting. Granted, the certificate that $P$ runs forever is a sample from an infinite distribution $\mathcal{D}$ over strings. But each string in $\mathcal{D}$'s support is finite—and not only that, but the *expected* string-length is finite and very small.

In Section 4.1, we show that our algorithm to solve the halting problem using a Deutschian CTC is "optimal," in the following senses. If we demanded either that there exist a fixed-point with finite support, *or* that there exist a fixed-point with a computable upper bound on its expected string-length, then Deutschian CTCs would no longer take us beyond the computable languages.

Our central result, in Section 5, says that even a quantum Turing machine with a Deutschian CTC can't solve much *more* than the halting problem. (By analogy, in the complexity case, the central result of Aaronson and Watrous [5] said that a quantum Turing machine with a Deutschian CTC can't solve more than $\mathsf{PSPACE}$.) This result is not obvious. If CTCs let us solve the halting problem, then a priori, why shouldn't they take us all the way up through the arithmetical hierarchy?

To rule that possibility out, we'll need to show that *there is an algorithm, using an oracle for the halting problem, that finds fixed-points of quantum Turing machines whenever they exist.* This is the main technical contribution of the paper.

The intuition is simple: even if our evolution operator $S$ acts on an infinite-dimensional Hilbert space, one can still enumerate over finite approximations to all possible fixed-points of $S$. For each candidate fixed-point $\sigma$, one can then use an oracle for the halting problem to check whether there exists a time $t > 0$ such that $S^t(\sigma)$ is far from $\sigma$.

In one direction, if $\sigma$ is close to a fixed-point of $S$, then it's easy to see that $S^t(\sigma)$ will be close to $\sigma$ for all $t > 0$. The difficulty concerns the converse direction. If $S^t(\sigma)$ is close to $\sigma$ for all $t$, how do we know that $\sigma$ is close to an actual fixed-point of $S$? In finite dimensions, we can prove that implication using an infinite series that defines a projection onto the $+1$ eigenspace of $S$. But in infinite dimensions the series need not converge.

Nevertheless, we recover the desired conclusion using the Riesz Representation Theorem, a key result in functional analysis. Even though the relevant infinite series need not converge, the Riesz Representation Theorem will imply that it converges *componentwise*, and that will be enough to give us a fixed-point $\rho$ close to $\sigma$. The proof requires switching between two different norms: the trace norm, which is used for measuring distances between quantum mixed states; and the usual Hilbert space norm applied to the "vectorizations" of those mixed states.

In Section 6, we move on to consider the computability theory of *postselected* CTCs. There, we point out an ambiguity that doesn't arise in the complexity case, but is specific to computability. Namely, are we allowed to postselect on the event that a probabilistic Turing machine $M$ halts, ignoring the branches where $M$ runs forever? Or, what turns out to be equivalent: can we have a prior probability distribution that's not normalized (e.g., where the probabilities sum to $\infty$), insisting only that the postselected distribution be normalized? We call this "$\infty$-postselection."

Our results, in Section 7, are as follows. If $\infty$-postselection is not allowed, then classical and quantum Turing machines with postselected CTCs yield only the computable languages. If, on the other hand, $\infty$-postselection is allowed, then classical and quantum Turing machines with postselected CTCs again let us solve the halting problem in finite time. Indeed, they let us solve all problems that are nonadaptively (or "weakly truth-table") reducible to the halting problem. But this is the limit: even an $\infty$-postselected CTC can be simulated using nonadaptive queries to HALT. Thus, $\infty$-postselected CTCs have an enormous power that's still strictly less than the power of Deutschian CTCs—much like what happens in the complexity setting (assuming the containments $\mathsf{BQP} \subseteq \mathsf{PP} \subseteq \mathsf{PSPACE}$ are strict).

The computational power of CTCs under various assumptions (classical vs. quantum, bounded vs. unbounded size, and postselected vs. $\infty$-postselected vs. Deutschian) is summarized in the following table. The results for the unbounded-size case are all new to this paper.

| | **Bounded-Size CTC** | | **Unbounded-Size CTC** | | |
|---|---|---|---|---|---|
| | Postselected | Deutschian | Postselected | $\infty$-Postselected | Deutschian |
| **Classical** | $\mathsf{BPP}_{\mathsf{path}}$ [9] | PSPACE [2] | Computable | $\mathsf{Computable}_{\|\|}^{\mathrm{HALT}}$ | $\mathsf{Computable}^{\mathrm{HALT}}$ |
| **Quantum** | PP [9] | PSPACE [5] | Computable | $\mathsf{Computable}_{\|\|}^{\mathrm{HALT}}$ | $\mathsf{Computable}^{\mathrm{HALT}}$ |

To summarize, we show for the first time that CTCs would violate the Church-Turing Thesis (rather than "just" the Extended or Polynomial-Time Church-Turing Thesis). We also give fairly sharp results about which assumptions are necessary and sufficient for such a violation, and the precise effects that different kinds of CTCs would have on computability.

Some might object to the Deutschian CTC model considered in this paper, on the grounds of *non-robustness*: i.e., that the fixed-points of a CTC could be sensitive to arbitrarily small changes to the superoperator acting around the CTC. So in Section 8, we discuss this issue in detail. We first show that there is indeed a robustness issue for our unmodified CTC algorithm to solve the halting problem—and that in some sense, the issue is "worse" than the corresponding robustness issue for solving PSPACE-complete problems with a CTC. (Robustness for the latter was discussed by Aaronson and Watrous [5].) However, following a proposal by Bacon [8], we then explain why error-correction can provide robustness even inside a CTC—and furthermore, only classical error-correction is needed in this case. Finally, we discuss a 1996 result of Aharonov et al. [6], which shows (roughly speaking) that exponentially-long reversible computations are impossible in the presence of noise and the absence of ancilla bits. We explain why this result is consistent with the possibility of robust computation in the Deutschian CTC model.

Stepping back, what is the significance of our results? Some, perhaps, will interpret the results as additional evidence against the possibility of CTCs in our world, alongside the other "extravagant" consequences of CTCs that are known. Those consequences include not only the ability to solve PSPACE-complete problems [2, 5], but also the ability to, for example, violate the No-Cloning Theorem and other basic principles of quantum mechanics [7]. Others might count our results as evidence, not against *all* CTCs, but only against CTCs that allow our sort of construction to go through. In particular, they might hold that CTCs are physically acceptable as long as they admit spurious fixed-points, or as long as they can't accept strings of unbounded length—though note that the latter would "merely" decrease CTCs' power from uncomputable back down to PSPACE. Finally, for those who don't care about CTCs, we hope the results will still be of interest, since they determine how hard it is to find a fixed-point of an arbitrary computable quantum channel, a problem that seems likely to find other applications.

## 2 Preliminaries

We review some basics of computability theory in Section 2.1, and of quantum information in Section 2.2.

### 2.1 Computability

For concreteness, throughout we'll assume Turing machines over the alphabet $\{0, 1, \#\}$, though everything we say would generalize to most other reasonable models of computation.[6]

Given a Turing machine $P$, we use $\langle P\rangle$ to denote a description of $P$, and $P(\ )$ to denote $P$ run on a blank input. Then we can define

$$\textsc{Halt} = \{\langle P\rangle : P(\ ) \text{ halts}\}.$$

A language $L \subseteq \{0, 1\}^*$ is *computable* if there's some Turing machine $P$ that accepts every $x \in L$ and rejects every $x \notin L$. We let $\mathsf{Computable}$ denote the class of computable languages, also called the recursive languages. We let $\mathsf{Computable}^{\textsc{Halt}}$ denote the class of languages that are computable with a $\textsc{Halt}$ oracle: in computability theory, this is also called $\Delta_2$, a level of the arithmetical hierarchy. Note that $\mathsf{Computable}^{\textsc{Halt}}$ can also be defined as the class of languages $L$ that are Turing-reducible to $\textsc{Halt}$, i.e. for which $L \leq_T \textsc{Halt}$.

There are other reductions in computability theory; one that we'll need is *weak truth-table reductions*. We say that $L$ is weakly truth-table reducible to $L'$, or $L \leq_{\text{wtt}} L'$, if there's a Turing machine $P$ that decides $x \in L$ with the help of an $L'$-oracle, and has the additional property that all of its queries to $L'$ are submitted in parallel. The "weak" refers to the fact that we don't require $P$ to define a language (i.e., to halt for every input $x$) if $L'$ is replaced by some other oracle. Borrowing notation from complexity theory, we'll let $\mathsf{Computable}_{||}^{\textsc{Halt}}$ denote the class of languages that are weakly truth-table reducible to $\textsc{Halt}$.

The following is proved only for completeness:

**Proposition 1** *$\mathsf{Computable}_{||}^{\textsc{Halt}}$ is a strict subset of $\mathsf{Computable}^{\textsc{Halt}}$.*

**Proof.** Let $\textsc{Halt}_{||}^{\textsc{Halt}}$ be the halting problem for Turing machines with weak truth-table access to a $\textsc{Halt}$ oracle. Then clearly $\textsc{Halt}_{||}^{\textsc{Halt}} \notin \mathsf{Computable}_{||}^{\textsc{Halt}}$, since we can relativize the proof that $\textsc{Halt} \notin \mathsf{Computable}$. But $\textsc{Halt}_{||}^{\textsc{Halt}} \in \mathsf{Computable}^{\textsc{Halt}}$, because given an instance $\langle P\rangle$ of $\textsc{Halt}_{||}^{\textsc{Halt}}$, a Turing machine with a $\textsc{Halt}$ oracle can first obtain the parallel queries $q_1, \ldots, q_k$ that $P(\ )$ submits to its $\textsc{Halt}$ oracle, then use its own $\textsc{Halt}$ oracle to obtain the answers $a_1, \ldots, a_k$ to those queries, and finally use its $\textsc{Halt}$ oracle again to decide whether $P(\ )$ halts given those answers. ■

[6]As discussed in Sections 1.1 and 9.1, because of the possibility of CTCs having "spurious fixed-points," this statement does *not* follow immediately from the Church-Turing Thesis: indeed, one can design Turing-universal models of computation for which our results fail. But the statement is nevertheless true for models that allow arbitrary bit manipulations.

## 2.2 Quantum Information

Here we just review a few key concepts. A *superoperator* is a function \$ that maps one quantum mixed state to another. It has the form

$$\$\left(\rho\right)=\sum_i E_i \rho E_i^\dagger$$

where

$$\sum_i E_i^\dagger E_i = I$$

is the identity. Superoperators are the most general kind of transformation allowed in quantum mechanics.

Given two mixed states $\rho$ and $\sigma$, the *trace distance* $\left\|\rho-\sigma\right\|_{\text{tr}}$—the quantum generalization of the total variation distance—measures the maximum possible bias with which $\rho$ and $\sigma$ can be distinguished by a measurement. It can be computed as

$$\frac{1}{2}\sum_i \left|\lambda_i\right|,$$

where $\lambda_1,\ldots,\lambda_n$ are the eigenvalues of $\rho-\sigma$.

A fundamental fact is that for all mixed states $\rho,\sigma$ and all superoperators \$,

$$\left\|\$\left(\rho\right)-\$\left(\sigma\right)\right\|_{\text{tr}} \leq \left\|\rho-\sigma\right\|_{\text{tr}}.$$

## 3 Turing Machines with Deutschian CTCs

This section defines the computational model that plays the central role in this paper. We start with the classical case, then move on the quantum case in Section 3.1.

A *classical Turing machine with Deutschian CTC*, or $\mathsf{TM}_{\mathsf{CTC}}$, is a probabilistic Turing machine $M$ whose memory is divided into two registers:

- a "causality-respecting register" $\mathcal{R}_{\text{CR}}$, which contains $M$'s input, and
- a "closed timelike curve register" $\mathcal{R}_{\text{CTC}}$.

We can think of both registers as containing infinitely many squares, though just like with an ordinary Turing machine, only finitely many squares can be accessed after a finite number of steps. Thus, the deterministic state of a $\mathsf{TM}_{\mathsf{CTC}}$ is an ordered pair of binary strings $(x,y)$—both finite, but with no upper bound on their length—where $x$ is the contents of $\mathcal{R}_{\text{CR}}$, and $y$ is the contents of $\mathcal{R}_{\text{CTC}}$. We can imagine, for example, that $x$ and $y$ are both marked off by the special delimiter symbol #, which isn't used for any other purpose, and that after the #'s are just infinite sequences of 0's. Any time $M$ expands or contracts $x$ or $y$, it moves the # delimiters accordingly.

That $M$ is probabilistic means that it has an operation that sets the current square to 0 or 1 with equal probability 1/2. We could also allow probabilistic transitions with (say) arbitrary rational probabilities, or even algebraic or computable probabilities, but it will follow from our results in this paper that none of these choices increase a $\mathsf{TM}_{\mathsf{CTC}}$'s power.

To be a valid $\mathsf{TM}_{\mathsf{CTC}}$, the machine $M$ must satisfy the following condition:

(i) $M(x,y)$ halts with probability 1,[7] outputting either 0 or 1, for every input pair $(x,y) \in \mathcal{R}_{\text{CR}} \times \mathcal{R}_{\text{CTC}}$.

We can think of "outputting either 0 or 1" as setting some designated output square in $\mathcal{R}_{\text{CR}}$ to 0 or 1 respectively, though this won't matter for anything that follows.

Because of condition (i), for every input $x$ to $\mathcal{R}_{\text{CR}}$, there's some infinite-dimensional stochastic matrix $S_x$—that is, some probabilistic mapping from the set $\{0,1\}^*$ to itself—that $M$ induces on the $\mathcal{R}_{\text{CTC}}$ register, if we run $M$ on $(x,y)$ and ignore its effects on $\mathcal{R}_{\text{CR}}$. Now let $\mathcal{D}$ be a probability distribution over $\{0,1\}^*$. Then we call $\mathcal{D}$ a *fixed-point* of $S_x$ if it satisfies the consistency equation

$$S_x(\mathcal{D}) = \mathcal{D}.$$

As discussed earlier, in finite dimensions every stochastic matrix $S$ has a fixed-point, but in infinite dimensions this is false. Thus, to be a valid $\text{TM}_{\text{CTC}}$, we require $M$ to satisfy a second condition:

(ii) $S_x$ has at least one fixed-point, for every input $x \in \{0,1\}^*$.

Given an input $x$ and a probability distribution $\mathcal{D}$ over $\{0,1\}^*$, we call $\mathcal{D}$ *accepting* if

$$\Pr_{y \sim \mathcal{D}}[M(x,y) \text{ outputs } 1] \geq \frac{2}{3},$$

and *rejecting* if

$$\Pr_{y \sim \mathcal{D}}[M(x,y) \text{ outputs } 0] \geq \frac{2}{3}.$$

Here the probabilities are over $y \sim \mathcal{D}$ as well as any internal randomness used by $M$. Then to be a valid $\text{TM}_{\text{CTC}}$, we require $M$ to satisfy a third and final condition:

(iii) For every input $x \in \{0,1\}^*$, either every fixed-point of $S_x$ is accepting, or every fixed-point of $S_x$ is rejecting.

We say that $M$ *accepts* $x$ in the first case, and that it *rejects* $x$ in the second case. Finally, given a language $L \subseteq \{0,1\}^*$, we say that $M$ *decides* $L$ if it accepts every $x \in L$ and rejects every $x \notin L$. Then $\mathsf{Computable}_{\mathsf{CTC}}$ is the class of all languages that are decided by some $\text{TM}_{\text{CTC}}$.

## 3.1 Quantum Turing Machines with Deutschian CTCs

A quantum Turing machine with Deutschian CTC, or $\text{QTM}_{\text{CTC}}$, is in some sense just the quantum generalization of the $\text{TM}_{\text{CTC}}$ defined above. That is, it's a machine that implements a unitary transformation on a tensor product Hilbert space $\mathcal{R}_{\text{CR}} \otimes \mathcal{R}_{\text{CTC}}$, where $\mathcal{R}_{\text{CR}}$ is the causality-respecting part and $\mathcal{R}_{\text{CTC}}$ is the closed timelike curve part. Both registers have countably infinite dimension, and are spanned by basis states corresponding to the binary strings:

$$|\epsilon\rangle, |0\rangle, |1\rangle, |00\rangle, |01\rangle, |10\rangle, \ldots$$

[7]Throughout this paper, "halts with probability 1" simply means: for all $\varepsilon > 0$, there exists a $t$ such that after $t$ steps, the machine has halted with probability at least $1 - \varepsilon$.

Again, one can imagine each such basis state $|x\rangle$ as really corresponding to an *infinite* binary string, with a special delimiter symbol separating $x$ itself from an infinite sequence of trailing zeroes:

$$|01\rangle = |01\#00\cdots\rangle$$

Given a normalized quantum superposition over this countable infinity of strings, of the form

$$|\psi\rangle = \sum_{x\in\{0,1\}^*} \alpha_x |x\rangle, \qquad \sum_{x\in\{0,1\}^*} |\alpha_x|^2 = 1,$$

one can act on $|\psi\rangle$ using a *quantum Turing machine*: that is, a finite control, with a finite-dimensional internal Hilbert space, which can move back and forth over the qubits,[8] applying unitary transformations that affect only $O(1)$ contiguous tape qubits at a time. We can assume that these local unitary transformations have rational entries only (and can therefore be specified using finitely many bits); it will follow from our results that allowing more general unitaries wouldn't increase the power of the model, even in the presence of CTCs.

Besides the tape qubits, the local unitary transformations also act on the machine's internal Hilbert space, as well as on a subspace spanned by $|\mathrm{L}\rangle, |\mathrm{R}\rangle, |\mathrm{Acc}\rangle, |\mathrm{Rej}\rangle$ which determines whether, at the next time step, the machine will move one square left on the tape, one square right, halt and accept, or halt and reject. This subspace is measured at each time step to determine whether to halt and accept, halt and reject, or continue. Before the machine halts, the tape contents, internal Hilbert space, and $\{|\mathrm{L}\rangle, |\mathrm{R}\rangle\}$ state can all be in superposition and all be entangled with one another.

For a more detailed definition of the quantum Turing machine model, see Bernstein and Vazirani [10].[9]

Given any quantum Turing machine $M$, and any initial state $|x\rangle$ of $\mathcal{R}_{\mathrm{CR}}$, one can associate a superoperator $\$_x$ that's induced on $\mathcal{R}_{\mathrm{CTC}}$. In the finite-dimensional case, it would follow from the work of Deutsch [12] that $\$_x$ must have at least one fixed-point: that is, a mixed state $\rho$ such that $\$_x(\rho) = \rho$. In the infinite-dimensional case, by contrast, fixed-points need not always exist, for the same reason why they need not exist in the classical case: suppose, for example, that

$$\$_x(|y\rangle\langle y|) = |y+1\rangle\langle y+1|$$

for all $y$, where $y$ is an integer written in binary. Thus, in order for $M$ to define a valid $\mathrm{QTM}_{\mathrm{CTC}}$, we impose three conditions on it, which parallel the conditions in the classical case:

(i) $M$ halts with probability 1, outputting either $|\mathrm{Acc}\rangle$ or $|\mathrm{Rej}\rangle$, for every input pair $|\psi\rangle\otimes|\varphi\rangle \in \mathcal{R}_{\mathrm{CR}}\otimes\mathcal{R}_{\mathrm{CTC}}$.

(ii) $\$_x$ has at least one fixed-point $\rho$, for every input $x\in\{0,1\}^*$.

[8] Or actually qutrits, because of the delimiter symbol #.

[9] For the past 23 years, Bernstein and Vazirani's original quantum Turing machine model has almost never been used—simply because it's much more convenient to talk about *classical* Turing machines that output descriptions of quantum circuits, a model known to be equivalent to QTMs [17]. However, allowing CTCs forces a return to the QTM model, at least temporarily, since the whole question we're trying to answer is whether fixed-points of infinite-dimensional quantum operations might be harder to find than fixed-points of infinite-dimensional classical ones.

(iii) For every input $x \in \{0,1\}^*$, either every fixed-point $\rho$ of $\$_x$ is "accepting"—that is,

$$\Pr\left[M\left(|x\rangle\langle x| \otimes \rho\right) \text{ outputs } |\text{Acc}\rangle\right] \geq \frac{2}{3},$$

or every fixed-point $\rho$ is "rejecting"—that is,

$$\Pr\left[M\left(|x\rangle\langle x| \otimes \rho\right) \text{ outputs } |\text{Rej}\rangle\right] \geq \frac{2}{3}.$$

Again, we say that $M$ accepts $x$ in the first case and rejects $M$ in the second; and that $M$ *decides* the language $L \subseteq \{0,1\}^*$ if accepts every $x \in L$ and rejects every $x \notin L$. We then let $\mathsf{QComputable}_{\mathsf{CTC}}$ be the class of all languages decided by some $\text{QTM}_{\text{CTC}}$.

## 4 Solving Uncomputable Problems with Deutschian CTCs

Having defined $\text{TM}_{\text{CTC}}$'s, we're now ready to see what they can do. Let's start by proving that a Turing machine with a Deutschian CTC can solve the halting problem.

**Lemma 2** $\text{H}\small{\text{ALT}}$$\in \mathsf{Computable}_{\mathsf{CTC}}$.

**Proof.** We're given a Turing machine $\langle P \rangle$, and need to decide whether $P(\,)$ halts. Suppose $P(\,)$ runs for at least $t$ steps; then let $\sigma_t$ be a string that encodes the first $t$ steps of $P(\,)$'s execution history in some canonical way. Thus, $\sigma_0$ encodes a blank input tape that hasn't yet been acted upon, with $P$ in its initial configuration; and $\sigma_{t+1}$ can easily be obtained from $\sigma_t$ by appending the result of one additional $P$ step. Note that, given an arbitrary string $y$, together with knowledge of $\langle P \rangle$, it's easy to decide whether $y = \sigma_t$ for some $t$, and if so which $t$. Call $\sigma_t$ a *halting history* if it shows $P$ halting at its $t^{th}$ step, and a non-halting history otherwise.

Now our $\text{TM}_{\text{CTC}}$, $M$, takes $\langle P \rangle$ as input in its causality-respecting register $\mathcal{R}_{\text{CR}}$, and some string $y$ as input in its CTC register $\mathcal{R}_{\text{CTC}}$, and does the following:

```
If $y$ is a halting history, then leave $y$ unchanged in the CTC register, and output "HALT"
If $y = \sigma_t$ is a non-halting history, then set $y := \sigma_{t+1}$ with probability $1/2$ and $y := \sigma_0$ with probability $1/2$, and output "LOOP"
If $y$ is not a valid execution history for $P(\,)$, then set $y := \sigma_0$, and output "LOOP"
```

There are now two cases. If $P(\,)$ halts, say in $t$ steps, then the unique fixed-point of the induced operation on $y$ is a singleton distribution concentrated on $y = \sigma_t$. Therefore $M$ outputs "HALT" with certainty. If, on the other hand, $P(\,)$ runs forever, then one can check that the unique fixed-point is a geometric distribution:

- $y = \sigma_0$ with probability $1/2$
- $y = \sigma_1$ with probability $1/4$
- $y = \sigma_2$ with probability $1/8$
- etc.

Therefore $M$ outputs "LOOP" with certainty. What rules out the possibility of additional fixed-points is the third step, which returns any invalid execution history back to the initial state $\sigma_0$.[10] ■

Next, we generalize Lemma 2, to show that a Turing machine with a Deutschian CTC can decide any language that's computable with a HALT oracle. As we'll see in Section 5, this turns out to be the exact power of Deutschian CTCs.

**Theorem 3** $\mathsf{Computable}_{\mathsf{CTC}}$ *contains* $\mathsf{Computable}^{\mathrm{HALT}} = \Delta_2$ *(that is, the class of languages that are Turing-reducible to* HALT*).*

**Proof.** We're given as input an oracle Turing machine $\langle P \rangle$, which repeatedly queries a HALT oracle, and we need to decide whether $P\,(\,)$ accepts or rejects, promised that one of those is the case. We can assume without loss of generality that $P$ queries its HALT oracle at, and only at, certain predetermined time steps $\tau_1 < \tau_2 < \cdots$, such as the perfect squares or the powers of 2. For $P$ can always delay a query until the next such time step $\tau_k$, and it can also always insert irrelevant dummy queries when needed.

Just like in the proof of Lemma 2, given an ordinary (non-oracle) Turing machine $B$, let $\sigma_{B,t}$ be a string that encodes the first $t$ steps of $B\,(\,)$'s execution history in a canonical way. Also, let $B_i$ be the $i^{th}$ Turing machine that $P$ submits to its oracle to find out whether it halts. Then our $\mathrm{TM}_{\mathrm{CTC}}$, $M$, will act on strings in its CTC register that look like

$$\sigma = \langle \sigma_{B_1,t_1}, \ldots, \sigma_{B_k,t_k} \rangle$$

—that is, $k$-tuples of execution histories. For all $i \in \{1, \ldots, k\}$, define $q_i := 1$ if $\sigma_{B_i,t_i}$ is a halting execution history and $q_i := 0$ otherwise. Then we call a $k$-tuple $\sigma$ *valid* if:

- Each $\sigma_{B_i,t_i}$ is a valid execution history (not necessarily until completion), representing the first $t_i$ steps of some Turing machine $B_i$.
- If we run $P\,(\,)$, treating its $i^{th}$ oracle query as returning the result $q_i$, then $B_1, \ldots, B_k$ really are the first $k$ machines that $P$ submits to its oracle, at time steps $\tau_1, \ldots, \tau_k$ respectively.

A few more definitions:

We call $\sigma$ *halting* if, when we run $P\,(\,)$, treating its $i^{th}$ query as returning the result $q_i$, the machine halts by time step $\tau_{k+1}$. In such a case, we call $\sigma$ *accepting* if $P\,(\,)$ accepts and *rejecting* otherwise. Otherwise we call $\sigma$ non-halting. If $\sigma$ is non-halting, then let $B_{k+1}$ be the machine that $P\,(\,)$ queries about at time step $\tau_{k+1}$, assuming again that $P\,(\,)$'s first $k$ queries were answered with $q_1, \ldots, q_k$ respectively.

Let $\delta = \langle\,\rangle$ be the unique valid $k$-tuple for $k = 0$: that is, a string that encodes $P\,(\,)$ in its initial state, before $P$ has made any queries to its HALT oracle.

Finally, let $S_B$ be the probabilistic mapping of $\mathcal{R}_{\mathrm{CR}}$ to itself from Lemma 2, assuming that $\mathcal{R}_{\mathrm{CTC}}$ is initialized to $\langle B \rangle$.

We can now define our $\mathrm{TM}_{\mathrm{CTC}}$ $M$. This machine takes $\langle P \rangle$ as input in its causality-respecting register $\mathcal{R}_{\mathrm{CR}}$, and some string $\sigma$ as input in its CTC register $\mathcal{R}_{\mathrm{CTC}}$, and does the following:

[10] A different way to prove Lemma 2 would be via an algorithm that sends just a single positive integer $t$ around the CTC (rather than the entire $t$-step Turing machine history), and that then computes the Turing machine history for itself. We thank Michael Devin for this observation.

```
If $\sigma$ is not valid, then set $\sigma := \delta$
If $\sigma = \langle \sigma_{B_1,t_1}, \ldots, \sigma_{B_k,t_k} \rangle$ is valid and halting, then:
    • Set $\sigma := \langle S_{B_1}(\sigma_{B_1,t_1}), \ldots, S_{B_k}(\sigma_{B_k,t_k}) \rangle$
    • Accept if $\sigma$ is accepting and reject otherwise
If $\sigma = \langle \sigma_{B_1,t_1}, \ldots, \sigma_{B_k,t_k} \rangle$ is valid and non-halting, then:
    • Set $\sigma := \left\langle S_{B_1}(\sigma_{B_1,t_1}), \ldots, S_{B_k}(\sigma_{B_k,t_k}), \sigma_{B_{k+1},0} \right\rangle$
```

It's clear that $M$ can be constructed, since checking the validity of a $k$-tuple, applying the probabilistic iterative maps $S_{B_i}$, and simulating $P\left(\,\right)$ to find the next machine $B_{k+1}$ that it queries about are all computable operations. Note that it's crucial, for this, that we treat the first $k$ queries as having the responses $q_1, \ldots, q_k$ (which we can read from $\sigma$), rather than the "true" responses, which we could learn only by querying a Halt oracle.

Now suppose that in a valid execution, $P\left(\,\right)$ makes exactly $k$ queries to its Halt oracle, about machines $B_1, \ldots, B_k$ respectively. Also, let $\mathcal{D}_{B_i}$ be the unique fixed-point of $S_{B_i}$—which, by Lemma 2, encodes the answer to whether $B_i\left(\,\right)$ halts. Finally, let $S'$ be the probabilistic mapping that $M\left(\langle P \rangle, \sigma\right)$ induces on $\sigma$, the state of its CTC register.

Then we claim that the unique fixed-point of $S'$ is $\langle \mathcal{D}_{B_1}, \ldots, \mathcal{D}_{B_k} \rangle$: that is, a tensor product of $k$ independent distributions, which are the fixed-point distributions of $B_1, \ldots, B_k$ respectively.

It's clear that the theorem follows from this claim—since if $\sigma$ is sampled from $\langle \mathcal{D}_{B_1}, \ldots, \mathcal{D}_{B_k} \rangle$, then $M\left(\langle P \rangle, \sigma\right)$ accepts with certainty if $P^{\text{Halt}}\left(\,\right)$ accepts, and rejects with certainty if $P^{\text{Halt}}\left(\,\right)$ rejects.

To prove the claim: first, it's clear that $\langle \mathcal{D}_{B_1}, \ldots, \mathcal{D}_{B_k} \rangle$ *is* a fixed-point of $S'$. For any sample $\sigma$ from $\langle \mathcal{D}_{B_1}, \ldots, \mathcal{D}_{B_k} \rangle$ is both valid and halting. But this means that $S'$ just applies $S_{B_1}, \ldots, S_{B_k}$ to $\mathcal{D}_{B_1}, \ldots, \mathcal{D}_{B_k}$ respectively. But we know that $\mathcal{D}_{B_i}$ is a stationary distribution of $S_{B_i}$ for all $i$.

To see that there are no *other* fixed-points of $S'$: first, any fixed-point must have support only on valid $k$-tuples, since $M$ maps any invalid $k$-tuple back to $\delta$. Second, if $\mathcal{D}$ is a fixed-point of $S'$, then $\mathcal{D}$ marginalized to its first coordinate must be the unique fixed-point of $S_{B_1}$, since otherwise applying $S_{B_1}$ to the first coordinate would change $\mathcal{D}$. But by induction, this means that $\mathcal{D}$ marginalized to its *second* coordinate must be the unique fixed-point of $S_{B_2}$, and so on, where $B_1, \ldots, B_k$ are the actual machines that $P\left(\,\right)$ queries about, assuming that the previous queries were answered correctly. ■

## 4.1 On the Need for Unbounded Fixed-Points

Stepping back, Lemma 2—that is, our algorithm to solve the halting problem using Deutschian CTCs—has at least two striking features. First, the algorithm can produce fixed-points that have no finite support (i.e., that are probability distributions over infinitely many strings). Second, while the fixed-points $\mathcal{D} = (p_x)_{x \in \{0,1\}^*}$ produced by the algorithm always have finite *expected* string-length,

$$|\mathcal{D}| := \sum_{x \in \{0,1\}^*} p_x \, |x| \, ,$$

no upper bound on $|\mathcal{D}|$ is computable a priori. (This is because, in the halting case, $\mathcal{D}$ is a singleton distribution concentrated on a single string $\sigma_t$, but the length $|\sigma_t|$ of that string scales like $t$, the number of steps that the input machine $P$ makes before halting.)

Philosophically, one might object to either of these features: even in a hypothetical universe with CTCs, is it reasonable to imagine a computer that might suddenly need to accept, say, a $10^{10^{10^{10}}}$-bit string in its CTC register? What would it even physically mean to engineer such a computer?

There are two points to stress in response. First, even an ordinary Turing machine, with no CTC, can accept inputs of unbounded length, so in some sense it's not CTCs that are producing the unbounded aspect here, just the Turing machine model itself! Second, *despite* the unbounded aspect, our CTC algorithm clearly solves the halting problem in a sense that would be impossible for Turing machines without CTCs: namely, regardless of whether $P\left(\ \right)$ halts or runs forever, the CTC outputs a finite string that certifies that fact. (With an ordinary TM, by contrast, finite certificates exist only in the case that $P\left(\ \right)$ halts.)

In any case, let's now show that both "unbounded" features are necessary, if we want to solve the halting problem using a Turing machine with a Deutschian CTC.

**Proposition 4** *There is an algorithm to decide whether a* $\mathrm{TM}_{\mathrm{CTC}}$ $M$ *accepts or rejects, provided that* $M$ *is promised to have a fixed-point with finite support.*

**Proof.** The algorithm is simply to iterate over all $k \geq 0$, and for each $k$, search for a fixed-point of $M$ that has support only on $\{0,1\}^{\leq k}$. By assumption, such a fixed-point $\mathcal{D}$ must exist for *some* $k$. Furthermore, we can recognize such a $\mathcal{D}$ when we find one, because it will have its support on a finite, strongly-connected component in $\{0,1\}^{\leq k}$, with no transitions that lead to $\{0,1\}^{>k}$ with any nonzero probability. Indeed, every such strongly-connected component gives rise to a fixed-point of $M$. Once we find such a $\mathcal{D}$—which we will, after finite time—we then simply need to check whether $\mathcal{D}$ leads to acceptance or rejection with high probability. ■

One particular consequence of Proposition 4 concerns "narrow" Deutschian CTCs, as studied by Bacon and others [8, 16, 14] and discussed in Section 1.1. These are Deutschian CTCs that can only send 1 bit back in time—or more generally, at most $f(n)$ bits back in time, for *any* function $f$. It follows from Proposition 4 that narrow Deutschian CTCs don't let us solve any uncomputable problems.

A second consequence concerns Deutschian CTCs that are promised to have *deterministic* fixed-points (i.e., fixed-points concentrated on a single string). Since any such fixed-point has finite support, it follows from Proposition 4 that these CTCs don't let us solve uncomputable problems either.[11] Note that, in the complexity setting, restricting to deterministic fixed-points decreases the power of Deutschian CTCs from $\mathsf{PSPACE}$ to $\mathsf{NP} \cap \mathsf{coNP}$. Note also that, in finite dimensions, the "purpose" of allowing probabilistic or quantum fixed-points was to ensure that a fixed-point always exists. In infinite dimensions, as we've seen, probabilistic fixed-points no longer serve that original purpose, but they're still relevant to determining CTCs' computational power.

The following is a bit more nontrivial.

**Theorem 5** *There is an algorithm to decide whether a* $\mathrm{TM}_{\mathrm{CTC}}$ $M$ *accepts or rejects, provided that* $M$ *is promised to have a fixed-point* $\mathcal{D}$ *with expected string-length* $|\mathcal{D}| \leq \ell$*, where* $\ell$ *is any upper bound that's computable given* $\langle M \rangle$.

[11] This can also be observed directly: just iterate over all strings $y \in \{0,1\}^*$, halting when a deterministic fixed-point is found.

**Proof.** Let $\mathcal{D}$ be a fixed-point with $|\mathcal{D}| \leq \ell$. Then by Markov's inequality, for any given $\varepsilon > 0$ we have

$$\Pr_{x \sim \mathcal{D}}\left[|x| > \frac{\ell}{\varepsilon}\right] < \varepsilon.$$

So there exists a distribution $\mathcal{D}'$, with support only on $\{0,1\}^{\leq \ell/\varepsilon}$, such that $\|\mathcal{D}' - \mathcal{D}\| < \varepsilon$, where $\|\cdot\|$ denotes total variation distance.

Now, let $U$ be a finite collection of probability distributions over $\{0,1\}^{\leq \ell/\varepsilon}$, which contains an $\varepsilon$-approximation to *every* probability distributions over $\{0,1\}^{\leq \ell/\varepsilon}$. Then $U$ must contain a distribution $\mathcal{E}$ such that

$$\|\mathcal{E} - \mathcal{D}\| \leq \left\|\mathcal{E} - \mathcal{D}'\right\| + \left\|\mathcal{D}' - \mathcal{D}\right\| < 2\varepsilon.$$

Fix (say) $\varepsilon = 0.01$. Then if $\mathcal{E}$ leads $M$ to accept with high probability, so does $\mathcal{D}$; if $\mathcal{E}$ leads $M$ to reject with high probability, so does $\mathcal{D}$.

Thus, the sole remaining question is how we can recognize such an $\mathcal{E}$ within $U$. Let $S$ be the evolution operator induced by $M$ on $\mathcal{R}_{\text{CTC}}$. Suppose we simply apply $S$ repeatedly to $\mathcal{E}$, producing $S(\mathcal{E})$, $S(S(\mathcal{E}))$, and so on. If $\|\mathcal{E} - \mathcal{D}\| < 2\varepsilon$ for some fixed-point $\mathcal{D}$, then for all $t$ we have

$$\begin{aligned}\left\|S^{(t)}(\mathcal{E}) - \mathcal{E}\right\| &< \left\|S^{(t)}(\mathcal{E}) - S^{(t)}(\mathcal{D})\right\| + \left\|S^{(t)}(\mathcal{D}) - \mathcal{D}\right\| \\ &\leq \|\mathcal{E} - \mathcal{D}\| \\ &< 2\varepsilon.\end{aligned}$$

So if, at any point, we reach an iterate $S^{(t)}(\mathcal{E})$ such that

$$\left\|S^{(t)}(\mathcal{E}) - \mathcal{E}\right\| > 2\varepsilon,$$

then we've refuted the hypothesis that $\mathcal{E}$ was $2\varepsilon$-close to a fixed-point of $S$. But Lemma 7 in Section 5 will imply that the converse also holds: if

$$\left\|S^{(t)}(\mathcal{E}) - \mathcal{E}\right\| \leq 2\varepsilon$$

for all $t$, then $\mathcal{E}$ is $2\varepsilon$-close to a fixed-point of $S$.

So suppose that, for every distribution $\mathcal{E} \in U$ in parallel, we search for a $t$ such that

$$\left\|S^{(t)}(\mathcal{E}) - \mathcal{E}\right\| > 2\varepsilon,$$

eliminating an $\mathcal{E}$ from consideration whenever such a $t$ is found. Then eventually, we must either eliminate all $\mathcal{E} \in U$ that lead to acceptance with high probability, or else eliminate all $\mathcal{E} \in U$ that lead to rejection with high probability. At that point we know whether to reject or accept, respectively. ■

Theorem 5 can also be generalized to the quantum case (i.e., to $\text{QTM}_{\text{CTC}}$'s), with essentially the same proof.

## 5 Upper Bound on the Power of Deutschian CTCs

We now prove a converse of Theorem 3, by showing that $\mathsf{Computable}_{\mathsf{CTC}}$ is contained in $\Delta_2 = \mathsf{Computable}^{\text{HALT}}$. Indeed, we'll prove the stronger result that even $\mathsf{QComputable}_{\mathsf{CTC}}$, the languages decidable by a quantum Turing machine with a Deutschian CTC, can be decided with a HALT oracle. Combined with Theorem 3, this will imply that

$$\mathsf{Computable}_{\mathsf{CTC}} = \mathsf{QComputable}_{\mathsf{CTC}} = \mathsf{Computable}^{\text{HALT}}.$$

To prove this result, a central ingredient will be a certain operator-theory lemma. Intuitively, the lemma says that, if applying a linear transformation $F$ over and over to a vector $u$ never takes us far from $u$, then $u$ must be close to an actual fixed-point of $F$ (that is, a vector $w$ satisfying $Fw = w$). Furthermore, this is true even in infinite-dimensional Hilbert spaces, despite the fact that there, linear transformations need not even *have* nonzero fixed-points in general.

Throughout this section, we let $||$ denote the usual Hilbert space norm (that is, the 2-norm). We'll need the following standard result from functional analysis.

**Theorem 6 (Riesz Representation Theorem)** *Let $\mathcal{H}$ be a (possibly infinite-dimensional) Hilbert space, and let $\varphi : \mathcal{H} \to \mathbb{C}$ be a linear functional (i.e. an element of the dual space $\mathcal{H}^*$). Then there's a unique vector $w \in \mathcal{H}$ representing $\varphi$, i.e. such that for all $v \in \mathcal{H}$,*

$$\langle w, v \rangle = \varphi(v).$$

We can now prove the lemma we need.

**Lemma 7** *Let $\mathcal{H}$ be a Hilbert space, let $F$ be a linear operator on $\mathcal{H}$, and let $\epsilon > 0$. Let $u \in \mathcal{H}$ be a unit vector such that for all $t \geq T$, we have $\left|F^t u - u\right| \leq \varepsilon$. Then there exists a vector $w \in \mathcal{H}$ such that $|w - u| \leq \varepsilon$ and $Fw = w$ (i.e., $w$ is a fixed-point of $F$).*

**Proof.** The basic idea is to introduce a linear functional $\varphi : \mathcal{H} \to \mathbb{C}$ based on the sequence $F^t u$. The Riesz Representation Theorem (Theorem 6) then gives us a candidate for the fixed-point $w$.

Let

$$\varphi(v) := \lim_{t \to \infty} \frac{1}{t} \sum_{i=1}^{t} \left\langle F^i u, v \right\rangle.$$

This limit always converges (in general for any bounded sequence $a_i = \left\langle F^i u, v \right\rangle$, the sequence of averages $b_t = \frac{1}{t} \sum_{i=1}^{t} a_i$ is convergent). Also since the inner term in the limit is linear in $v$, we have that $\varphi$ itself is also linear, i.e.

$$\varphi(c_1 v_1 + c_2 v_2) = c_1 \varphi(v_1) + c_2 \varphi(v_2).$$

By Theorem 6, it follows that there exists a $w \in \mathcal{H}$ such that for all $v \in \mathcal{H}$,

$$\langle w, v \rangle = \varphi(v).$$

The crucial claim is now that $w$ is a fixed point of $F$; that is, $Fw = w$.

Let's first show that given this claim, the lemma follows. For this, we just need to upper-bound $|w - u|$:

$$|w - u| = \sup_{|v|=1} \langle w - u, v \rangle = \sup_{|v|=1} \varphi(v) - \langle u, v \rangle.$$

By the definition of $\varphi$, we have

$$\varphi(v) - \langle u, v\rangle = \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \left\langle F^i u - u, v \right\rangle$$

We can bound each term above by

$$\left|\left\langle F^i u - u, v\right\rangle\right| \leq \left|F^i u - u\right| \cdot |v| = \left|F^i u - u\right|.$$

Since this term is bounded by $\varepsilon$ for $t \geq T$, it follows that

$$|\varphi(v) - \langle u, v\rangle| \leq \lim_{t\to\infty} \frac{2T + \varepsilon\,(t - T)}{t} = \varepsilon.$$

Therefore

$$|w - u| \leq \varepsilon$$

which means that $u$ is only $\varepsilon$ away from the fixed-point $w$, which is the desired result.

It remains only to prove the claim. For any $v \in \mathcal{H}$ we have

$$\langle Fw, v\rangle = \left\langle w, F^\dagger v\right\rangle = \varphi(F^\dagger v) = \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \left\langle F^i u, F^\dagger v\right\rangle = \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \left\langle F^{i+1} u, v\right\rangle.$$

The point is that

$$\left|\frac{1}{t} \sum_{i=1}^{t} \langle F^{i+1} u, v\rangle - \sum_{i=1}^{t} \langle F^i u, v\rangle\right| = \frac{\left|\langle F^{t+1} u, v\rangle - \langle Fu, v\rangle\right|}{t} \leq \frac{2}{t},$$

which vanishes as $t \to \infty$. It follows that for all $v \in \mathcal{H}$,

$$\langle Fw, v\rangle = \langle w, v\rangle.$$

Taking $v = Fw - w$, we have

$$\langle Fw - w, Fw - w\rangle = 0.$$

The claim follows. ■

Following Aaronson and Watrous [5], given a mixed state $\rho$ over some fixed orthonormal basis, we denote by $\operatorname{vec}(\rho)$ the *vectorization* of $\rho$—that is, the flattening out of $\rho$ from an $N \times N$ matrix into an $N^2$-length vector with the same list of entries. Note that $\operatorname{vec}(\rho)$ still makes sense even if $\rho$ lives in a countable infinity of dimensions.

Now, for any superoperator \$ acting on such mixed states, we have the following crucial fact: there exists a matrix $\operatorname{mat}(\$)$ such that for all $\rho$,

$$\operatorname{mat}(\$)\operatorname{vec}(\rho) = \operatorname{vec}(\$(\rho)).$$

In other words, when we flatten out density matrices into vectors, superoperators just become linear transformations. Again, this still makes sense even in infinite dimensions.

Using this notation, we can now make an additional observation about the proof of Lemma 7. Namely, if $u = \operatorname{vec}(\sigma)$ is the vectorization of a mixed state, and $F = \operatorname{mat}(\$)$ is the "matricization" of a superoperator, then the fixed-point $w$ that we construct is also the vectorization of a mixed state. That is, we have $w = \mathrm{vec}(\rho)$, for some $\rho$ satisfying $\$(\rho) = \rho$ and hence $Fw = w$ as well. This is so because $w$ was defined by taking an entrywise limit of a convex combination of vectors of the form $F^i u = \mathrm{vec}\left(\$^i(\sigma)\right)$. But each of these vectors is the vectorization of a mixed state, so their limiting convex combination is the vectorization of a mixed state as well.

Before proceeding to the main result, we'll need two more propositions, which relate the trace distance between two mixed states $\sigma$ and $\rho$ to the Euclidean distance $|\mathrm{vec}(\sigma) - \mathrm{vec}(\rho)|$ between their vectorizations.

**Proposition 8** *Let $\sigma$ and $\rho$ be any two mixed states. Then*

$$|\mathrm{vec}(\sigma) - \mathrm{vec}(\rho)| \leq \|\sigma - \rho\|_{\mathrm{tr}}.$$

**Proof.** Observe that

$$|\mathrm{vec}(\sigma) - \mathrm{vec}(\rho)| = \|\sigma - \rho\|_F,$$

where

$$\|A\|_F = \sqrt{\sum_{ij} |a_{ij}|^2}$$

is the so-called *Frobenius norm* of a matrix $A$. The proposition now follows from the general matrix inequality $\|A\|_F \leq \|A\|_{\mathrm{tr}}$ (which also holds in the infinite-dimensional case). ■

The other direction is a bit more involved:

**Proposition 9** *Let $\sigma$ be a mixed state on $\{0,1\}^{\leq k}$, and let $\rho$ be a mixed state on $\{0,1\}^*$. Then*

$$\|\sigma - \rho\|_{\mathrm{tr}} \leq 2^{k/4+2}\sqrt{|\mathrm{vec}(\sigma) - \mathrm{vec}(\rho)|}.$$

**Proof.** Let $\widetilde{\rho}$ be the unnormalized truncation of $\rho$ to have support only on $\{0,1\}^{\leq k}$. Then

$$\begin{aligned}\|\sigma - \rho\|_{\mathrm{tr}} &\leq \|\sigma - \widetilde{\rho}\|_{\mathrm{tr}} + \|\widetilde{\rho} - \rho\|_{\mathrm{tr}} \\ &\leq |\mathrm{vec}(\sigma) - \mathrm{vec}(\widetilde{\rho})|_1 + \|\widetilde{\rho} - \rho\|_{\mathrm{tr}},\end{aligned}$$

where the second line just uses a general inequality for matrix norms. Furthermore,

$$\begin{aligned}|\mathrm{vec}(\sigma) - \mathrm{vec}(\widetilde{\rho})|_1 &\leq \sqrt{2^{k+1}-1}\,|\mathrm{vec}(\sigma) - \mathrm{vec}(\widetilde{\rho})| \\ &\leq \sqrt{2^{k+1}}\,|\mathrm{vec}(\sigma) - \mathrm{vec}(\rho)|\end{aligned}$$

by Cauchy-Schwarz. Meanwhile, the "Almost As Good As New Lemma" (see for example [4]) implies that

$$\|\widetilde{\rho} - \rho\|_{\mathrm{tr}} \leq 2\sqrt{\varepsilon},$$

where

$$\begin{aligned}\varepsilon &= \sum_{x \in \{0,1\}^{>k}} \langle x|\rho|x\rangle \\ &= 1 - \sum_{x \in \{0,1\}^{\leq k}} \langle x|\rho|x\rangle \\ &= \sum_{x \in \{0,1\}^{\leq k}} \langle x|\sigma|x\rangle - \sum_{x \in \{0,1\}^{\leq k}} \langle x|\widetilde{\rho}|x\rangle \\ &\leq |\mathrm{vec}(\sigma) - \mathrm{vec}(\widetilde{\rho})|_1.\end{aligned}$$

Hence

$$\begin{aligned}\|\sigma-\rho\|_{\mathrm{tr}} &\leq \left|\operatorname{vec}(\sigma)-\operatorname{vec}(\widetilde{\rho})\right|_1+2\sqrt{\left|\operatorname{vec}(\sigma)-\operatorname{vec}(\widetilde{\rho})\right|_1}\\ &\leq 3\sqrt{\left|\operatorname{vec}(\sigma)-\operatorname{vec}(\widetilde{\rho})\right|_1}\\ &\leq 2^{k/4+2}\sqrt{\left|\operatorname{vec}(\sigma)-\operatorname{vec}(\rho)\right|},\end{aligned}$$

where the second line assumed $\left|\operatorname{vec}(\sigma)-\operatorname{vec}(\widetilde{\rho})\right|_1 \leq 1$ (if this fails, then the proposition holds for trivial reasons). ■

We can now prove the main result.

**Theorem 10** $\mathsf{QComputable}_{\mathsf{CTC}} \subseteq \mathsf{Computable}^{\textsc{Halt}}$.

**Proof.** Given a language $L \in \mathsf{QComputable}_{\mathsf{CTC}}$ and an input $x \in \{0,1\}^*$, the problem of deciding whether $x \in L$ can be boiled down to the following. We're given a superoperator \$ on the set of all binary strings $\{0,1\}^*$, via a quantum Turing machine that implements \$. We're promised that \$ has at least one fixed-point: that is, a mixed state $\rho$ such that $\$(\rho)=\rho$. We're also given a quantum Turing machine $Q$, and are promised that either

(i) $\Pr[Q(\rho) \text{ accepts}] \geq \frac{2}{3}$ for all fixed-points $\rho$ of \$, or

(ii) $\Pr[Q(\rho) \text{ rejects}] \geq \frac{2}{3}$ for all fixed-points $\rho$ of \$.

(We further know that $Q(\rho)$ halts with probability 1, but our proof will go through even if we drop this assumption.) The problem is to decide whether (i) or (ii) holds.

Let $\mathcal{M}_k$ be the set of all mixed states over $\{0,1\}^{\leq k}$ that have rational entries only, when written out in the $\{0,1\}^{\leq k}$ basis. Also, let $\mathcal{M}=\bigcup_{k\geq 1}\mathcal{M}_k$. Then clearly $\mathcal{M}$ is countably infinite, with a computable enumeration, and is also dense in the set of mixed states—i.e., it can approximate any mixed state over $\{0,1\}^*$ arbitrarily closely.

We need to design an oracle Turing machine, call it $A$, that takes descriptions of \$ and $Q$ as input, and that distinguishes case (i) from case (ii) with help from a Halt oracle. The machine $A$ will do the following.

```
Dovetail over all $k \geq 1$ and all mixed states $\sigma \in \mathcal{M}_k$.   For each one:

• Use the HALT oracle to check whether there exists a $t \geq 0$ such that

      $\left\|\sigma - \$^t(\sigma)\right\|_{\mathrm{tr}} > \frac{1}{2^{k+12}}$

• If no such $t$ exists, then:

    – Halt and accept if $\Pr[Q(\sigma) \text{ accepts}] \geq 0.55$
    – Halt and reject if $\Pr[Q(\sigma) \text{ rejects}] \geq 0.55$
```

To perform the required test using its HALT oracle, $A$ simply constructs an ordinary Turing machine that dovetails over all $t \geq 0$, and for each one, derives better and better lower bounds on $\left\|\sigma - \$^t(\sigma)\right\|_{\text{tr}}$—halting when and if any of these lower bounds exceed $\frac{1}{2^{k+12}}$.

To complete the proof, we need to show both that $A$ always halts, and that its answer is correct.

$A$ **always halts:** By assumption, \$ has a fixed-point—that is, a $\rho$ such that $\$(\rho) = \rho$. Also, by choosing $k$ sufficiently large, we can clearly find a $k$ and $\sigma \in \mathcal{M}_k$ such that

$$\|\sigma - \rho\|_{\text{tr}} \leq \frac{1}{2^{k+12}}.$$

This then implies that for all $t \geq 0$,

$$\begin{aligned} \left\|\$^t(\sigma) - \rho\right\|_{\text{tr}} &= \left\|\$^t(\sigma) - \$^t(\rho)\right\|_{\text{tr}} \\ &\leq \|\sigma - \rho\|_{\text{tr}} \\ &\leq \frac{1}{2^{k+12}}, \end{aligned}$$

where the second line used the fact that superoperators can't increase trace distance. So by Proposition 8, for all $t \geq 0$ we also have

$$\left|\text{vec}\left(\$^t(\sigma)\right) - \text{vec}(\rho)\right| \leq \frac{1}{2^{k+12}}.$$

Thus, $A$ will halt when it reaches $\sigma$, if not sooner.

$A$**'s output is correct:** Suppose $A$ halts, having found a $\sigma \in \mathcal{M}_k$ such that

$$\left\|\$^t(\sigma) - \sigma\right\|_{\text{tr}} \leq \frac{1}{2^{k+12}}$$

for all $t \geq 0$. By Proposition 8, we then have

$$\left|\text{vec}\left(\$^t(\sigma)\right) - \text{vec}(\sigma)\right| \leq \frac{1}{2^{k+12}}$$

for all $t \geq 0$ as well. By Lemma 7, this implies that there exists a fixed-point $\rho$ of \$ such that

$$|\text{vec}(\sigma) - \text{vec}(\rho)| \leq \frac{1}{2^{k+12}}.$$

By Proposition 9, this in turn implies

$$\|\sigma - \rho\|_{\text{tr}} \leq 2^{k/4+2}\sqrt{\frac{1}{2^{k+12}}} \leq \frac{1}{16}.$$

So if $\Pr[Q(\rho) \text{ accepts}] \geq \frac{2}{3}$, then

$$\Pr[Q(\sigma) \text{ accepts}] \geq \frac{2}{3} - \frac{1}{16},$$

while if $\Pr[Q(\rho) \text{ rejects}] \geq \frac{2}{3}$, then

$$\Pr[Q(\sigma) \text{ rejects}] \geq \frac{2}{3} - \frac{1}{16}.$$

Since we're promised that either all fixed-points $\rho$ satisfy the former or else they all satisfy the latter, this means that $A$ correctly decides whether $x \in L$. ■

Of course, Theorem 10 immediately implies that $\mathsf{Computable}_{\mathsf{CTC}} \subseteq \mathsf{Computable}^{\text{HALT}}$ as well. If we only cared about the classical case, we could somewhat simplify the proof of Theorem 10, although the proof would still require reasoning about infinite-dimensional Markov chains.

## 6 Turing Machines with Postselected CTCs

As mentioned in Section 1.1, Lloyd et al. [13] defined an alternative model of CTCs, based on postselection—and showed that, with a polynomial-size quantum circuit in the CTC, their model decides exactly the problems in the class $\mathsf{PostBQP} = \mathsf{PP}$. We'll now define a computability version of their model.

**The Classical Case.** A *classical Turing machine with (finitely) postselected CTC*, or $\text{TM}_{\text{PCTC}}$, is a probabilistic Turing machine $M$ that takes an input $x \in \{0,1\}^*$, and whose memory is divided into two registers $\mathcal{R}_{\text{CR}}$ and $\mathcal{R}_{\text{CTC}}$ like before. We assume that $\mathcal{R}_{\text{CR}}$ is initialized to $x$, while $\mathcal{R}_{\text{CTC}}$ is initialized to the empty string $\epsilon$.[12] We require $M$ to halt with probability 1 for all possible inputs of the form $(x, \epsilon)$. When $M$ halts, it can output either 0 (for Reject) or 1 (for Accept). When this happens, we call the run *selected* if $\mathcal{R}_{\text{CTC}}$ again contains $\epsilon$ at the time of halting. In order for $M$ to define a valid $\text{TM}_{\text{PCTC}}$, we require that for all inputs $x$:

(1) $\Pr[\text{selected run}] > 0$, and

(2) $\Pr[M(x) \text{ outputs } 1 \mid \text{selected run}] \in \left[0, \frac{1}{3}\right] \cup \left[\frac{2}{3}, 1\right]$,

where the probabilities are over $M$'s internal randomness.

We say that $M(x)$ *accepts* if it outputs 1 on at least 2/3 of selected runs, and that it *rejects* if it outputs 0 on at least 2/3 of selected runs. As usual, $M$ *decides* a language $L \subseteq \{0,1\}^*$ if $M$ accepts all $x \in L$ and rejects all $x \notin L$. Then $\mathsf{Computable}_{\mathsf{PCTC}}$ is the class of all languages decided by some $\text{TM}_{\text{PCTC}}$.

As a remark, the point of doing postselection is to force $M$ to have the same state at the beginning and the end of the postselected phase. In this way, we can "simulate" a CTC computation, which maps a fixed-point to itself.

**The Quantum Case.** A *quantum Turing machine with (finitely) postselected CTC*, or $\text{QTM}_{\text{PCTC}}$, is the straightforward quantum generalization of the above. That is, it's a quantum Turing machine $M$ whose (pure) state at any time lives in a tensor-product Hilbert space $\mathcal{R}_{\text{CR}} \otimes \mathcal{R}_{\text{CTC}}$, where $\mathcal{R}_{\text{CR}}$ and $\mathcal{R}_{\text{CTC}}$ are both of countably infinite dimension, and are spanned by the set of all finite binary strings. We assume that $\mathcal{R}_{\text{CR}}$ is initialized to $|x\rangle$ while $\mathcal{R}_{\text{CTC}}$ is initialized to $|\epsilon\rangle$.

As before, $M$ is required to halt with probability 1 for all inputs $|x\rangle \otimes |\epsilon\rangle$. When $M$ halts, it can either reject or accept, by placing a designated output qubit into the state $|0\rangle$ or $|1\rangle$ respectively. When this happens, we call the run *selected* if a measurement of $\mathcal{R}_{\text{CTC}}$ in the computational basis yields the outcome $|\epsilon\rangle$. In order for $M$ to define a valid $\text{QTM}_{\text{PCTC}}$, we require the same conditions (1) and (2) as in the classical case. The only difference is that here, the probabilities are over possible quantum measurement outcomes.

We say that $M(x)$ accepts if it outputs 1 on at least 2/3 of selected runs, and rejects if it outputs 0 on at least 2/3 of selected runs. Again, $M$ decides $L$ if it accepts all $x \in L$ and rejects all $x \notin L$; and $\mathsf{QComputable}_{\mathsf{PCTC}}$ is the class of all languages decided by some $\text{QTM}_{\text{PCTC}}$.

[12]In the treatment of Lloyd et al. [13], the initial state was a maximally entangled state. However, it's not hard to see that *any* fixed state that we can both prepare and project onto would lead to the same model of computation.

**$\infty$-Postselection.** Above, when we called the CTCs "(finitely) postselected," the adjective "finitely" referred to the fact that $M$ was required to halt with probability 1. However, we could also remove that requirement while keeping everything else the same. That is, we could let $M$ have a nonzero probability of running forever, but then still condition on the selected runs (where $M$ necessarily halts), and impose the same conditions (1) and (2) as before. This yields classes of languages that we'll call $\mathsf{Computable}_{\infty\mathsf{PCTC}}$ and $\mathsf{QComputable}_{\infty\mathsf{PCTC}}$ in the classical and quantum cases respectively, where the $\infty\mathsf{P}$ stands for "infinitely postselected."

There's a different way to understand $\infty$-postselection, although we won't develop it in detail. Namely, instead of saying that $M$ is allowed to have nonzero probability of running forever, we could require $M$ to halt with probability 1 on every input, but then let $M$ start with a *non-normalized "probability distribution" over strings* in some auxiliary register. For example, we could feed $M$ the "distribution" in which every string $y \in \{0,1\}^*$ occurs with the same probability $p > 0$. We require only that *after postselection*, we're left with an ordinary posterior distribution—and in particular, that

$$\sum_{y \in \{0,1\}^*} \Pr\left[\text{selected run} \mid y\right] \leq 1.$$

We then have the usual conditions (1) and (2) on the posterior distribution. In other words, we temporarily entertain the fiction that there exists a uniform distribution over $\{0,1\}^*$, knowing that we're going to postselect away all but a finite part of the infinite probability mass anyway, and thereby obtain a probability distribution that *does* exist.

Given an unnormalized initial distribution, we can simulate $\infty$-postselection, by interpreting $y$ as the sequence of random coin tosses made by $M$, and counting a run as selected only in cases where $y$ causes $M$ to halt. Conversely, given $\infty$-postselection, we can simulate an unnormalized initial distribution, by dovetailing over all $y \in \{0,1\}^*$ and all possible choices of random bits, and halting with the appropriate probability whenever we encounter a new selected run.

## 6.1 CTC-less Characterizations

Above, we defined the classes $\mathsf{Computable}_{\mathsf{PCTC}}$, $\mathsf{QComputable}_{\mathsf{PCTC}}$, etc. using "postselected CTCs," in order to make it as clear as possible that we're following the proposal of Lloyd et al. [13], and simply extending it to the computability setting. However, it's not hard to show that, as far as computability is concerned, the "CTC" aspect of these definitions can be jettisoned entirely.

More concretely, let a *(finitely) postselected TM* be a probabilistic or quantum Turing machine $M$ that halts with probability 1 on all inputs $x \in \{0,1\}^*$, and that when it halts, outputs either 0 (for Reject), 1 (for Accept), or $*$ (for No Response). Let

$$p := \Pr\left[M(x) \text{ outputs } 1\right], \qquad q := \Pr\left[M(x) \text{ outputs } 0\right].$$

Then we require that, for all inputs $x$:

(1) $p + q > 0$.

(2) Either $p \geq 2q$ or $q \geq 2p$.

If $p \geq 2q$, then we say $M$ *accepts* $x$, while if $q \geq 2p$ then we say $M$ *rejects* $x$. We say that $M$ *decides* the language $L$ if it accepts all $x \in L$ and rejects all $x \notin L$. Then $\mathsf{PostComputable}$ is the class of all languages decided by some postselected classical TM, and $\mathsf{PostQComputable}$ is the class of all languages decided by some postselected quantum TM.

We can also consider $\infty$-postselected TMs, which are the same as the above except that we drop the assumption that $M$ halts with probability 1. We call the resulting complexity classes $\infty\mathsf{PostComputable}$ and $\infty\mathsf{PostQComputable}$.

We then have the following equivalences.

**Lemma 11**

$$\begin{array}{c}\mathsf{Computable}_{\mathsf{PCTC}} = \mathsf{PostComputable} \\ \shortparallel \\ \mathsf{QComputable}_{\mathsf{PCTC}} = \mathsf{PostQComputable}.\end{array}$$

$$\begin{array}{c}\mathsf{Computable}_{\infty\mathsf{PCTC}} = \infty\mathsf{PostComputable} \\ \shortparallel \\ \mathsf{QComputable}_{\infty\mathsf{PCTC}} = \infty\mathsf{PostQComputable}.\end{array}$$

**Proof.** For the $\subseteq$ directions: it suffices to observe that a postselected Turing machine can just simulate a $\mathrm{TM}_{\mathrm{PCTC}}$, then postselect on the final state of $\mathcal{R}_{\mathrm{CTC}}$ being $\epsilon$ (or in the quantum case, on the measurement of $\mathcal{R}_{\mathrm{CTC}}$ in the computational basis returning the outcome $|\epsilon\rangle$). The acceptance and rejection conditions are the same.

For the $\supseteq$ directions: given a postselected Turing machine $M$, we just need to set up a postselected CTC computation in which $\mathcal{R}_{\mathrm{CTC}}$ is returned to its initial state $\epsilon$ if and only if $M$ either accepts or rejects. This is easy to arrange.

Finally, for $\mathsf{PostComputable} = \mathsf{PostQComputable}$ and $\infty\mathsf{PostComputable} = \infty\mathsf{PostQComputable}$, it suffices to observe that if $M$ is a quantum Turing machine, then

$$\Pr[M \text{ halts}] = \sum_{t=1}^{\infty} \Pr[M \text{ halts after exactly } t \text{ steps}],$$

and similarly for $\Pr[M \text{ accepts}]$ and $\Pr[M \text{ rejects}]$. Also, for each value of $t$, we can write $M$'s probability of halting after exactly $t$ steps as a sum of at most $\exp(t)$ complex numbers. But this means that we can build a classical Turing machine that has the same acceptance, rejection, and halting probabilities as $M$ does (or at any rate, probabilities multiplicatively close to the correct ones, which suffices for simulation purposes). This classical machine suffers an $\exp(t)$ factor slowdown relative to $M$, but that is irrelevant for computability. ■

## 7 The Power of Postselected CTCs

Lemma 11 tells us that, to understand the power of postselected CTCs (either quantum or classical), it suffices to understand the power of postselected probabilistic Turing machines. So in this section we solve the latter problem.

Let's start by showing that, in the computability setting, *finite* postselection yields no additional computational power.

**Proposition 12** $\mathsf{PostComputable} = \mathsf{Computable}$.

**Proof.** Let $M$ be a postselected TM, let $p = \Pr[M(\,)\text{ accepts}]$, and let $q = \Pr[M(\,)\text{ rejects}]$. Then to simulate $M(\,)$, we simply do breadth-first search over all possible computational paths of $M(\,)$, until we find a path that either accepts or rejects (which is promised to exist). This path gives us a positive lower bound, say $\varepsilon$, on $p+q$.

Next, we simulate $M(\,)$ until it's halted with probability at least $1-\varepsilon/10$—which must happen after some finite time $t$, since $M(\,)$ halts with probability 1. We then calculate $p^* = \Pr[M(\,)\text{ accepts by step } t]$ and $q^* = \Pr[M(\,)\text{ rejects by step } t]$. Our simulation accepts if $p^* > q^*$, and rejects if $p^* < q^*$. The reason this works is that if $p > 2q$ (the accept case), then

$$p^* \geq p - \frac{\varepsilon}{10} \geq p - \frac{p+q}{10} = 0.9p - 0.1q > 1.7q \geq q \geq q^*,$$

and likewise if $q > 2p$ (the reject case) then $q^* > p^*$. ■

An immediate corollary of Proposition 12 is that finitely postselected CTCs don't let us solve any uncomputable problems. By contrast, let's next consider $\infty$-postselection, and show why it *does* let us solve the halting problem (just as Deutschian CTCs do).

**Proposition 13** $\textsc{Halt} \in \infty\mathsf{PostComputable}$.

**Proof.** Let $P$ be a Turing machine for which we want to decide whether $P(\,)$ halts. Then consider an $\infty$-postselected Turing machine $M$ that does the following:

```
With probability $0.01$, halt and reject
Otherwise, simulate $P(\,)$.  If $P(\,)$ ever halts, then halt and accept
```

If $P(\,)$ halts, then $M$ accepts with probability 0.99 and rejects with probability 0.01. If $P(\,)$ doesn't halt, then $M$ accepts with probability 0 and rejects with probability 0.01. Either way, then, after we postselect on halting, $M$ decides whether $P(\,)$ halts with bounded error. ■

Again, a corollary of Proposition 13 is that $\infty$-postselected CTCs let us decide $\textsc{Halt}$. Let's now generalize Proposition 13, to show how to use $\infty$-postselection to solve *any* problem that's weakly truth-table reducible to $\textsc{Halt}$.

**Theorem 14** $\infty\mathsf{PostComputable}$ *contains* $\mathsf{Computable}_{||}^{\textsc{Halt}}$ *(that is, the class of languages nonadaptively reducible to* $\textsc{Halt}$*).*

**Proof.** Let $L \in \mathsf{Computable}_{||}^{\textsc{Halt}}$, and let $A$ be an oracle Turing machine that decides $L$. Then given an input $x$, let $B_1, \ldots, B_k$ be the machines that $A$ queries its $\textsc{Halt}$ oracle about (and recall that $A$ prepares this list in advance of making any queries). Let $f(b_1, \ldots, b_k)$ be a partial Boolean function that encodes $A$'s output (1 for accept, 0 for reject, undefined for loop forever), assuming that the responses to the $\textsc{Halt}$ queries are $b_1, \ldots, b_k$ respectively.

We now give an $\infty$-postselected Turing machine, $M$, to decide whether $x \in L$.

```
Form a string $b = b_1 \cdots b_k \in \{0,1\}^k$ by setting $b_i := 0$ with independent probability
$0.01/k$, and $b_i := 1$ otherwise, for each $i \in [k]$
In dovetailing fashion, simulate $B_i(\,)$ for each $i \in [k]$ such that $b_i = 1$
```

```
    If ever $B_i\left(\ \right)$ has halted for all $i$ such that $b_i = 1$, then halt and output $f\left(b_1,\ldots,b_k\right)$
(or loop forever if $f\left(b_1,\ldots,b_k\right)$ is undefined)
```

To see why $M$ is correct: first, suppose that $b_i$ happens to get set to 1 if and only if $B_i\left(\ \right)$ halts, for each $i \in [k]$. In that case, clearly $f\left(b_1,\ldots,b_k\right)$ is well-defined, and $M$ halts and outputs it. It follows that $M$ has a nonzero probability of halting, for each input $x$.

Second, $M$ can halt *only* if $B_i\left(\ \right)$ halts for all $i \in [k]$ such that $b_i = 1$. So conditioned on $M$ halting, we know that $b_i = 0$ whenever $B_i\left(\ \right)$ loops; the question is whether we also have $b_i = 1$ whenever $B_i\left(\ \right)$ halts. But by the union bound, again conditioned on $M$ halting, we have this with probability at least

$$1 - \sum_{i\ :\ B_i(\ )\text{ halts}} \Pr\left[b_i = 0\right] \geq 1 - k\left(\frac{0.01}{k}\right) \geq 0.99.$$

In other words, conditioned on $M$ halting, it outputs the correct answer $f\left(b_1,\ldots,b_k\right)$ with probability at least 0.99. ■

Note that Proposition 13 and Theorem 14 still go through, to imply that an $\infty$-postselected CTC lets us decide all languages in $\mathsf{Computable}_{||}^{\text{HALT}}$, even if the CTC is "narrow"—that is, even if it can send only a single bit back in time (as in the model of [8, 16, 14]). The reason for this is that we might as well postselect on just a single CTC bit retaining its original value of 0—and then ensure that the bit *does* retain its original value of 0, if and only if some desired other events occur.

Finally, we show a matching *upper* bound on the class $\infty\mathsf{PostComputable}$—thereby precisely characterizing the power of $\infty$-postselection, as equivalent to nonadaptive access to a HALT oracle.

**Theorem 15** *$\infty\mathsf{PostComputable} \subseteq \mathsf{Computable}_{||}^{\text{HALT}}$.*

**Proof.** Let $M$ be an $\infty$-postselected TM. Then the following $\mathsf{Computable}_{||}^{\text{HALT}}$ algorithm, $A$, decides whether $M\left(\ \right)$ accepts or rejects:

```
    Define $p := \Pr\left[M\left(\ \right) \text{ accepts}\right]$ and $q := \Pr\left[M\left(\ \right) \text{ rejects}\right]$
    Simulate $M\left(\ \right)$ until some nonzero lower bound, $\varepsilon > 0$, is derived on $p+q$
    For all integers $k \in \left\{-1,0,1,\ldots,\left\lceil \log_{1.1}\frac{2}{\varepsilon}\right\rceil\right\}$, query the HALT oracle to decide whether
$p > 1.1^k\frac{\varepsilon}{2}$, and also whether $q > 1.1^k\frac{\varepsilon}{2}$
    If these queries reveal that $p > 1.1q$, then accept
    If they reveal that $q > 1.1p$, then reject
```

We now analyze $A$. First, note that $A$ always halts, since we have the promise that $M$ halts with nonzero probability. And thus, if we do breadth-first search over all the possible computation paths of $M\left(\ \right)$, we must eventually find paths that accept or reject with nonzero probability.

Now suppose we've derived a lower bound $\varepsilon > 0$ on $p+q$. Then $\max\left\{p,q\right\} > \frac{\varepsilon}{2}$. From the promise on $M\left(\ \right)$, another thing we know is that either $p \geq 2q$ or $q \geq 2p$. This implies that if $p \geq 2q$, then there must exist a nonnegative integer $k \leq \log_{1.1}\frac{1}{\varepsilon}$ such that $p > 1.1^k\frac{\varepsilon}{2}$ but $q \leq 1.1^{k-1}\frac{\varepsilon}{2}$. Likewise, if $q \geq 2p$, then there must exist a $k \leq \log_{1.1}\frac{1}{\varepsilon}$ such that $q > 1.1^k\frac{\varepsilon}{2}$ but $p \leq 1.1^{k-1}\frac{\varepsilon}{2}$. Either way, then, the responses to $A$'s queries to its HALT oracle provide a certificate, proving either that $p \geq 2q$ and hence $x \in L$, or else that $q \geq 2p$ and hence $x \notin L$.

Two final observations. First, even if a probabilistic Turing machine $M$ can run for arbitrarily long times, we can still use a single query to a HALT oracle to decide whether $\Pr\left[M\left(\ \right) \text{ accepts}\right] > \beta$, for any computable real number $\beta$. The reason is simply that, if $\Pr[M() \text{ accepts}] > \beta$, then there must be a finite time $T$ such that already

$$\sum_{t=1}^{T} \Pr[M() \text{ accepts after exactly } t \text{ steps}] > \beta.$$

(Note that this would *not* be the case if we'd asked instead about $\Pr[M() \text{ accepts}] \geq \beta$.)

The second observation is that all of $A$'s queries to its HALT oracle can be made in parallel. So $A$ is truly a $\mathsf{Computable}_{||}^{\text{HALT}}$ algorithm. ■

## 8 Error-Correction and Robustness of CTC Computability

Focusing on our Deutschian CTC algorithm for the halting problem (Lemma 2), a natural question is to what extent the algorithm is robust against small errors in the quantum operation acting inside the CTC.

Aaronson and Watrous [5, Section 5] proved that their CTC algorithm for PSPACE-complete problems is at least robust against exponentially small errors. But does that conclusion carry over to the computability setting? Also, can we error-correct Deutschian CTC algorithms, including the algorithms of this paper?[13]

### 8.1 Weak Robustness

Aaronson and Watrous [5, Section 5] showed their algorithm for PSPACE problems to be robust against $1/\exp\left(n^{O(1)}\right)$ entrywise errors in the superoperator \$ applied inside the CTC. This sort of robustness applies directly to the original algorithm unmodified, with no need for error-correction procedures.

Unfortunately, the computability setting raises a new issue for this sort of robustness. Consider, for example, the algorithm $M$ from Lemma 2 to decide whether a Turing machine $\langle P \rangle$ halts. There's no a-priori upper bound on the number of configurations of $P()$ that might appear in a fixed-point (indeed, the number might be infinite). As a consequence, we also can't give any single $\varepsilon > 0$, such that the fixed-point will be robust against entrywise changes to the superoperator of order $\varepsilon$.

So for example, suppose $P()$ runs for $t$ steps and then halts. Then in our Markov chain, the probability mass flowing to the halting state per time step is about $1/2^t$, *until* eventually all the probability mass collects at the halting state, thereby giving us the unique fixed-point (namely, the point distribution concentrated on the halting state).

But suppose now that there's a small error, which causes $\varepsilon$ probability mass to flow from the halting state back to the initial state at each time step. In that case, if $\varepsilon \gg 1/2^t$, then "the drain is too wide for the bathtub to fill": appreciable probability mass will *never* collect at the halting state. So with high probability, measuring the fixed-point state won't give us any indication that $P()$ halts, even though it does.

If we want robustness for the unmodified algorithm $M$, then we seem to need some assumption like the following: "on input strings that are $k$ bits long, the entrywise error in applying the superoperator that acts inside the CTC is at most $1/\exp(k)$." In other words, rather than assuming a uniform error rate, we could assume an error that rapidly goes to 0 for longer strings.

[13]We thank Michael Nielsen for bringing these questions to our attention—and in particular, for spurring us to realize that the computability setting might raise new issues.

The converse direction is better. If $P\left(\ \right)$ runs forever, then the unique fixed-point is given by a geometric distribution (with probabilities $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$). And as long as we maintain the property that *every* state that isn't a halting state transitions back to the initial state with a reasonably large probability, a small entrywise change to the superoperator will indeed produce only a small change to the fixed-point.

In any case, note that the bound that Aaronson and Watrous [5, Section 5] were able to prove, on the induced error in the fixed-point, had a multiplicative factor of $|S|$, where $|S|$ was the total number of basis states. So perhaps it's no surprise that the case $|S| = \infty$ raises new issues.

## 8.2 CTC Fault-Tolerance

We now consider instead the following model. We get to specify a superoperator \$ acting inside the CTC, via a computer program. But then \$ gets composed with a noise operator, which (let's say) applies depolarizing and dephasing errors of $\varepsilon$ independently to each qubit. Then Nature returns a fixed-point of the composed (noisy) superoperator $\$'$.

This is of course a much more severe error model than the one considered in Section 8.1. In this case, there's clearly no chance that we'll have passive protection against error: indeed, we wouldn't have that even with no CTCs. But could we use active fault-tolerance to protect ourselves?

Following Bacon [8], we claim that the surprising answer is yes, we could!

First, recall that the useful CTC algorithms that we've found are all purely classical. So, for the purpose of showing what problems we could still solve using noisy CTCs, we might as well assume a classical computer inside the CTC. In that case, dephasing errors are irrelevant, and it suffices to consider bit-flip errors only.

But then a solution presents itself almost immediately. Namely, after the computer inside the CTC computes its output string $y$, it encodes $y$ using a constant-distance error-correcting code before sending the result, $e\left(y\right)$, back in time. Likewise, given an encoded input string $e\left(y\right)$ from the future, the computer first decodes it to $y$ before processing it.

There are a few things we need to be careful about here: for example, we need to rule out the possibility of "spurious fixed-points," involving strings that aren't in our code. But we can handle those by mapping non-codeword strings to some fixed codeword string—just like, in our CTC algorithms, we already map invalid inputs to some standard reference input.

Also, of course, even after we do the encoding, there's a $1/\exp\left(n\right)$ probability that a codeword will get corrupted so badly that it can no longer be decoded after we send it back in time. And we know that, in the CTC setting, even tiny errors can completely change the fixed-point structure.

Fortunately, it turns out that we remain protected, in the case of our algorithms from this paper, as well the algorithms of Aaronson and Watrous [5]. To see why, first consider the complexity setting ($\mathsf{PSPACE} \subseteq \mathsf{BQP}_{\mathsf{CTC}}$). There, as mentioned earlier, Aaronson and Watrous [5, Section 5] proved that $1/\exp\left(n\right)$ entrywise errors can be tolerated. So we simply need to choose an error-correcting code with a good enough distance for their bound to apply. This causes at most a polynomial blowup in the size of the circuit inside the CTC.

Next let's consider the use of CTCs to solve the halting problem, as in Lemma 2. Here, we're again fine as long for every $n \geq 0$, and every $y \in \{0,1\}^n$,

$$\Pr\left[\text{the computer inside the CTC mishandles } y\right] \leq \frac{1}{\exp\left(n\right)}.$$

But a standard error-correcting code can achieve this as well. I.e., it's actually true that the error probability decreases exponentially with the string length $n$.[14]

## 8.3 Entropy Buildup?

Michael Nielsen raised to us the following worry. In 1996, Aharonov et al. [6] proved a result that could be interpreted as saying that arbitrarily-long, closed, reversible computations are impossible in the presence of noise. So why doesn't their result also destroy CTC computation in the presence of noise?

The answer is simply that, in Deutsch's model, a CTC need not be "closed," in the sense of acting reversibly on its input. Instead, we can have an interaction between the CTC register $\mathcal{R}_{\text{CTC}}$ and the causality-respecting register $\mathcal{R}_{\text{CR}}$, whose effect on $\mathcal{R}_{\text{CTC}}$ is to apply a superoperator to it. Indeed, this is an essential feature of Deutsch's model, to make it nontrivial: if the CTC superoperator $\$$ needed to be unitary, then the maximally mixed state would always be a fixed-point.

So, as a string loops around the CTC getting processed—which of course is a crude way to imagine what's really going on, namely the computation of a fixed-point—there's no reason why entropy needs to keep building up until it crowds out all the useful information. Instead, $\mathcal{R}_{\text{CTC}}$ is allowed to "pump out entropy" into $\mathcal{R}_{\text{CR}}$. This is all automatically handled in the Deutsch formalism, and it doesn't lead to nonsense like an infinite amount of entropy getting expelled into $\mathcal{R}_{\text{CR}}$.

Admittedly, this is partly because of the way Deutsch's formalism summarily "severs the entanglement" between $\mathcal{R}_{\text{CTC}}$ and $\mathcal{R}_{\text{CR}}$, which is an aspect of the formalism that Bennett et al. [9] among others severely criticized. I.e., in Deutsch's model, we assume that what emerges, in $\mathcal{R}_{\text{CR}}$, is simply whatever results from first finding a fixed-point $\rho$ of the superoperator acting on $\mathcal{R}_{\text{CTC}}$, and then plugging $\rho$ into $\mathcal{R}_{\text{CTC}}$ and the initial state into $\mathcal{R}_{\text{CR}}$ and seeing what happens. Any entanglement that our circuit produces between $\mathcal{R}_{\text{CTC}}$ and $\mathcal{R}_{\text{CR}}$, in the course of operating on *non*-fixed-point inputs, is considered irrelevant to the final result.

On the other hand, even if it somehow made sense to retain such entanglement, it still doesn't seem that a proper analysis of the situation would result in an infinite amount of entropy spilling out into $\mathcal{R}_{\text{CR}}$. The reason goes back to one of Deutsch's central original insights: namely, in a world with CTCs, we must not imagine that we keep looping around a CTC, with the number of times we've looped governed by some second, meta-time! Instead we should imagine that a fixed-point of the CTC evolution just appears somehow, and is then processed "once." Of course, that fixed-point might be a superposition or probability distribution over many classical inputs, and it might even be natural to interpret the fixed-point in terms of "looping around." But as far as the external world (i.e., $\mathcal{R}_{\text{CR}}$) is concerned, whatever evolution happens in $\mathcal{R}_{\text{CTC}}$ happens only once—albeit, to a fixed-point state—rather than over and over.

[14]One might worry about cases where the CTC computer receives a short string as input and produces a long string as output, or vice versa. But the CTC computer knows both the input length $n$ and the output length $m$. So it simply needs to encode the output string in such a way that the error probability is at most $\frac{1}{\exp(\max\{n,m\})}$.

## 9 Discussion

In this paper, we gave the first treatment of computability theory for a world with closed timelike curves. We saw why Deutschian CTCs, with no upper bound on their size, would let us solve the halting problem, but also why CTCs wouldn't let us solve much *more* the halting problem. In addition, we saw why quantum CTC computers are no more powerful than classical CTC computers; why Deutschian CTC computers are more powerful than postselected CTC computers (either slightly more powerful or much more powerful, depending on how postselected CTC computers are defined); and why error-correction can be useful even inside a CTC. These results complement previous results on complexity theory in a world with CTCs [2, 5, 8, 16, 14, 13]. Compared to the complexity case, the central new aspect of the computability case is the need for linear algebra on infinite-dimensional Hilbert spaces, where fixed-points don't always exist.

There are various interesting directions one could pursue from here; we'll discuss two.

### 9.1 Spurious Fixed-Points

In his original paper on CTCs, Deutsch [12] observed that, depending on the detailed laws of physics in our universe, Nature might have an "out," which would let it accommodate CTCs without ever needing to solve a hard computational problem.[15] This would involve what we'll call *spurious fixed-points*: that is, fixed-points of the evolution acting on the full physical state inside the CTC, which happen not to be fixed-points of the "computational" part of the evolution that we care about.

To illustrate, suppose that (following Lemma 2) we build a computer whose unique fixed-point encodes whether Goldbach's Conjecture is true, and place that computer inside a CTC. Then Nature might return a fixed-point in which the computer has some mysterious hardware failure (say, melted circuit boards), and therefore loops inertly around the CTC. In that way, Nature could satisfy Deutsch's consistency condition without needing to pronounce on Goldbach's Conjecture.

Normally, of course, we'd consider worries like "what if the hardware fails?" to be outside the scope of theoretical computer science. For we have both theory and experience telling us the probability of a hardware failure can be made negligible—i.e., that we really *can* build machines with autonomous abstraction layers that behave very nearly like Turing machines. But this experience might not carry over to a universe with CTCs. If Nature's only choices were (1) to solve an instance of the halting problem, or (2) to "violate our abstraction boundaries" by melting the computer's circuit boards, who's to say that Nature wouldn't pick the latter?

Deutsch [12] actually elevates this idea to the status of a principle: he argues that we have no good reason to believe CTCs couldn't exist in our universe, and that if they *do* exist, then the natural assumption is that they use spurious fixed-points to escape the need for computational superpowers.

For us, though, this is an open question about physics. If the laws of physics let us build a CTC computer that acted directly on the fundamental degrees of freedom of the universe, then finding a fixed-point for that computer really *would* force Nature to solve a PSPACE-complete problem (in the bounded case) or the halting problem (in the unbounded case). On the other hand, we could also imagine physical laws where (say) every reliable computer memory had to spend many

[15]This point was independently brought to the attention of one of us (Aaronson) by the philosopher Tim Maudlin. For a more detailed discussion see [3, Section 10].

physical bits to encode each logical bit, and where every possible evolution of the physical bits had a trivial fixed-point, one that meant nothing in terms of the logical bits.

Note that, for the above reason, two sets of physical laws could have exactly the same computational power in "normal" spacetimes (say, they could both be Turing-universal), yet have completely different computational powers in CTC spacetimes (with one solving uncomputable problems and the other not).[16]

So it becomes interesting to ask: can we classify physical laws (or even just, say, cellular automaton evolution rules) according to whether every CTC evolution admits a spurious fixed-point? What are the necessary and sufficient conditions on a cellular automaton $C$, such that adding a Deutschian CTC to $C$ lets us actually implement the algorithm for HALT from Lemma 2, or the algorithm for $\mathsf{PSPACE}$-complete problems from [2, 5]?

## 9.2 New Directions for Computability

Compared to complexity theory, computability theory seems sparse in its remaining open problems.[17] Certainly, many of the distinctions that give complexity theory its whole subject matter—for example, between deterministic and randomized decision procedures, or between classical and quantum ones—vanish in computability theory. In this paper, however, we saw how simply taking known complexity theorems (e.g., $\mathsf{BQP}_{\mathsf{CTC}} = \mathsf{PSPACE}$) and asking for their computability analogues led to rich new questions.

For which other theorems in classical or quantum complexity theory can we do this? Which complexity theorems "computabilize" (that is, have natural computability-theoretic analogues), and which ones "fail to computabilize"? Can we classify the "non-computabilizing techniques"? Are there other senses, besides the one of this paper, in which $\Delta_2 = \mathsf{Computable}^{\text{HALT}}$ behaves like "the computability version of $\mathsf{PSPACE}$"?

Another question, brought to our attention by Matt Hastings, is whether there are natural models of CTCs that let us decide even more than $\mathsf{Computable}^{\text{HALT}}$. One candidate would be a recursive model, in which CTCs can sprout smaller CTCs inside of them, and so on.

A final remark: in this paper we found that, in the presence of Deutschian CTCs, classical and quantum computers give rise to the same computability theory: that is, $\mathsf{Computable}_{\mathsf{CTC}} = \mathsf{QComputable}_{\mathsf{CTC}}$. Granted, we "expect" classical and quantum computing to coincide in the computability setting, so one might say this was no surprise. What *is* somewhat surprising, though, was that *proving* quantum/classical equivalence in CTC computability theory was far from immediate. It wasn't enough to know that classical computers can simulate quantum computers with exponential slowdown, since conceivably finding fixed-points of QTMs could be vastly harder than finding fixed-points of classical TMs.

This raises a general question: is there *any* model of computability, as opposed to complexity, whose quantum version is strictly more powerful than its classical version?

[16]Conversely, and for a different reason, two sets of physical laws could also have different computational powers in an ordinary spacetime, but the same power in a CTC spacetime. For example, Aaronson and Watrous [5] observed that

$$\mathsf{AC}^0_{\mathsf{CTC}} = \mathsf{P}_{\mathsf{CTC}} = \mathsf{BQP}_{\mathsf{CTC}} = \mathsf{PSPACE}_{\mathsf{CTC}} = \mathsf{PSPACE},$$

since a Deutschian CTC obliterates the distinctions among complexity classes within $\mathsf{PSPACE}$ that can implement the step function of a $\mathsf{PSPACE}$ machine.

[17]Not counting questions about the computability of specific problems, like whether a given polynomial equation has a rational solution.

## 10 Acknowledgments

We thank Matt Coudron, Michael Devin, Matt Hastings, Michael Nielsen, Ryan O'Donnell, and Cem Say for helpful discussions.

## References

[1] S. Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1:1–28, 2005. Earlier version in Proc. IEEE Complexity'2004. quant-ph/0402095.

[2] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, March 2005. quant-ph/0502072.

[3] S. Aaronson. Why philosophers should care about computational complexity. In B. J. Copeland, C. Posy, and O. Shagrir, editors, *Computability: Turing, Gödel, Church, and Beyond*, pages 261–328. MIT Press, 2013.

[4] S. Aaronson. The complexity of quantum states and transformations: From quantum money to black holes, February 2016. Lecture Notes for the 28th McGill Invitational Workshop on Computational Complexity, Holetown, Barbados. With guest lectures by A. Bouland and L. Schaeffer. www.scottaaronson.com/barbados-2016.pdf.

[5] S. Aaronson and J. Watrous. Closed timelike curves make quantum and classical computing equivalent. *Proc. Roy. Soc. London*, (A465):631–647, 2009. arXiv:0808.2669.

[6] D. Aharonov, M. Ben-Or, R. Impagliazzo, and N. Nisan. Limitations of noisy reversible computation. arXiv:quant-ph/9611028, 1996.

[7] D. Ahn, C. R. Myers, T. C. Ralph, and R. B. Mann. Quantum state cloning in the presence of a closed timelike curve. *Phys. Rev. A*, 88(022332), 2013. arXiv:1207.6062.

[8] D. Bacon. Quantum computational complexity in the presence of closed timelike curves. *Phys. Rev. A*, 70(032309), 2004. quant-ph/0309189.

[9] C. H. Bennett, D. Leung, G. Smith, and J. A. Smolin. Can closed timelike curves or nonlinear quantum mechanics improve quantum state discrimination or help solve hard problems? *Phys. Rev. Lett.*, 103(170502), 2009. arXiv:0908.3023.

[10] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. Earlier version in Proc. ACM STOC'1993.

[11] 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.

[12] D. Deutsch. Quantum mechanics near closed timelike lines. *Phys. Rev. D*, 44:3197–3217, 1991.

[13] S. Lloyd, L. Maccone, R. Garcia-Patron, V. Giovannetti, and Y. Shikano. The quantum mechanics of time travel through post-selected teleportation. *Phys. Rev. D*, 84(025007), 2011. arXiv:1007.2615.

[14] R. O'Donnell and A. C. C. Say. One time-traveling bit is as good as logarithmically many. In *Proc. Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 469–480, 2014.

[15] R. O'Donnell and A. C. C. Say. The weakness of CTC qubits and the power of approximate counting. www.cs.cmu.edu/˜odonnell/papers/ctc-qubits.pdf, 2015.

[16] A. C. Cem Say and A. Yakaryılmaz. Computation with multiple CTCs of fixed length and width. *Natural Computing*, 11(4):579–594, 2012.

[17] A. C-C. Yao. Quantum circuit complexity. In *Proc. IEEE FOCS*, pages 352–361, 1993.