One Time-traveling Bit is as Good as Logarithmically Many*

Ryan O'Donnell[1] and A. C. Cem Say[2]

1 **Computer Science Department, Carnegie Mellon University**
Pittsburgh, USA
odonnell@cs.cmu.edu

2 **Department of Computer Engineering, Boğaziçi University**
İstanbul, Turkey
say@boun.edu.tr

Abstract

We consider computation in the presence of closed timelike curves (CTCs), as proposed by Deutsch. We focus on the case in which the CTCs carry *classical* bits (as opposed to qubits). Previously, Aaronson and Watrous showed that computation with polynomially many CTC bits is equivalent in power to PSPACE. On the other hand, Say and Yakaryılmaz showed that computation with just 1 classical CTC bit gives the power of "postselection", thereby upgrading classical randomized computation (BPP) to the complexity class $\mathsf{BPP}_{\mathsf{path}}$ and standard quantum computation (BQP) to the complexity class PP. It is natural to ask whether increasing the number of CTC bits from 1 to 2 (or 3, 4, etc.) leads to increased computational power. We show that the answer is no: randomized computation with logarithmically many CTC bits (i.e., polynomially many CTC states) is equivalent to $\mathsf{BPP}_{\mathsf{path}}$. (Similarly, quantum computation augmented with logarithmically many *classical* CTC bits is equivalent to PP.) Spoilsports with no interest in time travel may view our results as concerning the robustness of the class $\mathsf{BPP}_{\mathsf{path}}$ and the computational complexity of sampling from an implicitly defined Markov chain.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases Time travel, postselection, Markov chains, randomized computation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.469

1 On time travel

We begin with a discussion of time travel. Readers not interested in this concept may skip directly to Section 2, wherein we define the problem under consideration in a purely complexity-theoretic manner, with no reference to time travel.

Kurt Gödel [20] was the first to point out that Einstein's theory of general relativity is consistent with the existence of *closed timelike curves* (CTCs), raising the theoretical possibility of time travel. Any model of time travel must deal with the "Grandfather Paradox", wherein a trip to the past causes a chain of events that leads to a future in which that very trip does not take place. Assume that a time-traveler changes the state of the universe at the earlier end t_0 of a time loop from state s to some different state s'. Then just what is the state of the universe at time t_0: is it s or s'? Seeing a logical inconsistency in this scenario,

* Ryan O'Donnell's work performed while at the Boğaziçi University Computer Engineering Department, supported by Marie Curie International Incoming Fellowship project number 626373.

most thinkers of earlier generations concluded that time travel to the past must be impossible. There is, however, a way out. In an influential paper [18], Friedman *et al.* suggested Nature might allow CTCs as long as they do not "change the past", an idea that has come to be known as the *Novikov self-consistency principle*. The main two rivaling models of time travel – the "Deutschian model" (which we study in this work), and the "postselected CTC model" from [26] – both conform to the Novikov self-consistency principle.

In the model put forward by Deutsch [16], the universe need not be in a single deterministic state at time t_0. Rather, the state of the universe should be viewed as a probability distribution over several states (possibly even quantum states) like s and s' in the example above. The requirement that the past should not change is fulfilled by stipulating that Nature sets the state x of the portion of the universe affected by the CTC at time t_0 to a fixed point of the operator f describing the evolution in the CTC (meaning $x = f(x)$).

To take the traditional example, suppose a deranged scientist can access a CTC to the past century, and he sends through it a bomb that is programmed to kill his grandfather (who is only a child back then). We consider two (classical) states of the universe at the time of the bomb's arrival: state 1 is "grandfather dies" and state 2 is "grandfather lives". We assume the universe proceeds deterministically from that point on: if the grandfather is killed, then in the future no bomb is sent through the CTC; conversely, if the grandfather lives, then the deranged scientist is born and *does* send the bomb back in time. We can model this evolution by a 2-state Markov chain with the following transition matrix (that happens to be deterministic): $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. In Deutsch's model, Nature sets the state of the universe to be the stationary distribution for this chain: $\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$. That is, the bomb arrives to kill the grandfather with probability $\frac{1}{2}$.

1.1 Computation with CTCs

As the reader can see, Nature performs a kind of computation here, determining the stationary distribution of the Markov chain that has been arranged within the CTC by the deranged scientist. It is natural to wonder if Nature's power can be effectively harnessed by a computational device. Indeed, Deutsch [16] pointed out that in general his model involves Nature solving an NP-hard problem; later, Brun [15] discussed the possibility of using CTCs to solve the Factoring problem efficiently. The first clear model of computation with Deutschian CTCs was proposed by Bacon [11]. Both Deutsch and Bacon consider sending *qubits* through a CTC. However, as pointed out by Aaronson [3], it is also very interesting (and simpler) to consider only classical bits passing through a CTC. Indeed, as far as we aware, there are no results showing that time-traveling quantum bits confer a computational advantage over time-traveling classical bits. Therefore, in the rest of this section we will sketch the Deutschian model of computation with classical CTC bits, and mention prior work. A formal complexity-theoretic definition of the model (with no reference to time travel) is given in Section 2.

Suppose that a computational agent $\mathcal{A}$ has access to a CTC which is "wide" enough to support the transmission of w bits. Thus the physical object being sent through the CTC can be in one of $S = 2^w$ states. (We may also more generally consider values of S that are not powers of 2.) Let us think of $\mathcal{A}$ as a classical polynomial-time randomized Turing machine (though it might be of another type; e.g., a BQP-machine). Say that $\mathcal{A}$ is trying to decide if a given input $x \in \{0,1\}^n$ belongs to language L. The algorithm $\mathcal{A}$ can read the w bits in the CTC, perform some computation, and then send a new string of w bits through

the CTC. Since the incoming and outgoing bit strings can be in one of $2^w = S$ states, and since $\mathcal{A}$ is a randomized algorithm, the operation of $\mathcal{A}$ on the CTC constitutes an S-state Markov chain M_x, which depends on the input $x \in \{0,1\}^n$ to the L-decision problem.

In the Deutschian model, we assume that once the Markov chain M_x is defined, Nature sets the distribution of the bits in the CTC to *some* stationary distribution of M_x. We emphasize that it's merely *some* stationary distribution (at least one of which always exists) – we don't assume that M_x must have a unique stationary distribution. (Now is a good time to mention that if $\mathcal{A}$ is allowed to send qubits along the CTC, then its operation constitutes a *quantum channel*. It is also known [16, 39] that every quantum channel has at least one stationary mixed state, and we assume Nature selects one.) Finally, given that the incoming CTC bits are now presumed to be in a stationary distribution for the Markov chain M_x, the algorithm $\mathcal{A}$ effectively gets one *sample* from this stationary distribution. Using this sample, the algorithm $\mathcal{A}$ can output its decision on whether or not $x \in L$. When thinking of $\mathcal{A}$ as a BPP-type machine, this decision should be correct with probability at least $\frac{2}{3}$.

1.2 Prior work

Bacon [11] considered the case of a 1-qubit CTC, though his construction actually works equally well with a CTC supporting just 1 classical bit. However, Bacon's model was also more generous in that he allowed 1-bit CTC computations as "subroutines" within polynomial-time algorithms; in effect, he allowed the use of poly(n) many 1-(qu)bit CTCs. Bacon showed that in this model one can efficiently solve any NP problem. Subsequently, Aaronson and Watrous [3, 5] investigated the model in which the CTC supports poly(n) many bits (i.e., $S = 2^{\text{poly}(n)}$ many states). They showed that this model is extremely powerful: if $\mathcal{A}$'s computational power is anywhere between AC^0 and PSPACE (including the most usual choices of BPP or BQP), the result is that the model becomes equivalent in power to PSPACE. Actually, this result was not even the main one in their paper; their main result is that if poly(n) many CTC *qubits* are allowed, then the power of the model is still *only* that of PSPACE.

Regarding the difference between using a 1-bit CTC polynomially many times, and using a poly(n)-bit CTC once, Aaronson [3] remarked, "It is difficult to say which model is the more reasonable!" One can argue that both models are rather impractical in that they require constructing new/wider CTCs as the input length increases.[1] Indeed, the main technical question left open at the end of Aaronson and Watrous's work was to understand the computational power of the more realistic "narrow" CTCs; e.g., one-time-use CTCs that can only transmit a single bit, or a bounded number of bits. In this direction, Say and Yakaryılmaz [31] showed that augmenting standard complexity models with access to a 1-bit CTC is exactly equal in power to augmenting them with "postselection" [4] (defined in Section 3). In particular, this shows that classical randomized computation with a 1-bit CTC is equivalent to the complexity class $\mathsf{BPP}_{\mathsf{path}}$, and quantum computation with a 1-bit CTC is equivalent to the complexity class PP. We recall in further detail the class $\mathsf{BPP}_{\mathsf{path}}$ in Section 3. For now, suffice it to say that it contains NP and coNP, is likely equal to $\mathsf{P}_{||}^{\mathsf{NP}}$, and is very likely to be much smaller than PSPACE. In particular, randomized computation with access to a 1-bit CTC can efficiently solve the SAT problem; this is discussed below in Example 3.

[1] Bearing in mind the comment concerning practical considerations in the final paragraph of Bacon's work [11].

To summarize, the aforementioned results show that for classical polynomial-time randomized computation, adding a 1-bit CTC gives the power of $\mathsf{BPP}_{\mathsf{path}}$ and adding a poly(n)-bit CTC gives the power of PSPACE. What about in between (presuming of course that $\mathsf{BPP}_{\mathsf{path}} \neq \mathsf{PSPACE}$)? Sticking with the more "realistic" end of the spectrum, this is the question motivating our work:

▶ **Question.** *Are* 2*-bit (or* 3*-bit,* 4*-bit etc.) CTCs more powerful than* 1*-bit CTCs?*

2 Formal computational complexity statements

In what follows we formally define the complexity model of computing with CTCs. Our definitions are equivalent to those in [5, 31]; however we phrase them differently, in terms of Markov chains. Informally and in brief, $\mathsf{BPP}_{\mathsf{CTC}[w]}$ is the class of languages decidable by efficient randomized algorithms that are allowed to set up a 2^w-state Markov chain and then freely get one sample from the chain's stationary distribution.

▶ **Definition 1.** Let M be an S-state Markov chain. A *state-transition oracle* $\mathcal{M}$ for M is any algorithm that takes as input a state $i \in [S]$ and outputs the state resulting from taking one random step in M starting from state i. Most typically we think of $S = 2^w$ and $\mathcal{M}$ as being implemented by a w-bit-input/output *standard randomized circuit*; i.e., one with AND, OR, NOT, and "probability-$\frac{1}{2}$ coin-flip" gates. We might also consider *standard quantum circuits* $\mathcal{M}$ in which Hadamard and Toffoli gates (which are universal [33]) are also used.

▶ **Definition 2.** Let $w = w(n)$ be a "width" parameter. Consider a deterministic polynomial-time Turing Machine $\mathcal{A}$ that, on input $x \in \{0,1\}^n$, outputs the description of two standard randomized circuits, $\mathcal{M}_x$ and $\mathcal{D}_x$. The circuit $\mathcal{M}_x$ should have w input and output bits, thereby defining a state-transition oracle for a Markov chain M_x on $S = 2^w$ states. The "decision circuit" $\mathcal{D}_x$ should have w input bits and one output bit. We suppose computation proceeds as follows: First, an arbitrary stationary distribution π for M_x is chosen. Next, a sample $\boldsymbol{i} \sim \pi$ is chosen from this distribution and is fed as input to $\mathcal{D}_x$. Finally, $\mathcal{D}_x$'s output gate is considered to be the overall output of $\mathcal{A}$'s computation. We define $\mathsf{BPP}_{\mathsf{CTC}[w]}$ to be the class of all languages L such that there exists an $\mathcal{A}$ as above with the following property: for every x (and every stationary distribution π for M_x),

$$\mathop{\mathbf{Pr}}_{\boldsymbol{i}}[\mathcal{A} \text{ outputs } 1] \geq \tfrac{2}{3} \quad \text{when } x \in L, \qquad \mathop{\mathbf{Pr}}_{\boldsymbol{i}}[\mathcal{A} \text{ outputs } 1] \leq \tfrac{2}{3} \quad \text{when } x \notin L.$$

We may also analogously define $\mathsf{BQP}_{\mathsf{CTC}[w]}$ in case $\mathcal{M}_x$ and $\mathcal{D}_x$ are allowed to be standard quantum circuits.

▶ Remark. We warn the reader that our notation $\mathsf{C}_{\mathsf{CTC}[w]}$ is different from that in [5, 31], in that "$\mathsf{CTC}[w]$" signifies a CTC carrying w *classical* bits. We suggest notation such as $\mathsf{BQP}_{\mathsf{QCTC}[w]}$ for the case of CTCs carrying w qubits; however we neither define nor consider CTC-qubits in this paper (except in a concluding open problem).

▶ Remark. There is nothing special about considering Markov chains with S states where S is a power of 2. However we stick with the above notation for simplicity, and for consistency with similar complexity class definitions such as that of $\mathsf{P}^{\mathsf{NP}[w]}$ (polynomial-time computation with $2^w - 1$ parallel queries to an NP oracle).

▶ **Example 3.** Following [31], let us show that $\mathsf{NP} \subseteq \mathsf{BPP}_{\mathsf{CTC}[1]}$. Equivalently, we illustrate how SAT can be solved by a "1-bit CTC algorithm" $\mathcal{A}$, which can set up a 2-state Markov chain and get a sample from its stationary distribution. On input an n-variable CNF

formula ϕ, algorithm $\mathcal{A}$ constructs a state-transition circuit $\mathcal{M}_\phi$ for a certain 2-state Markov chain M_ϕ. Think of state 0 of M_ϕ as meaning "no evidence that ϕ is satisfiable" and state 1 as meaning "evidence that ϕ is satisfiable". The operation of $\mathcal{M}_\phi$ is as follows: On input state i, $\mathcal{M}_\phi$ first chooses a uniformly random string $\boldsymbol{y} \in \{0,1\}^n$ and checks if it satisfies ϕ. If $\boldsymbol{y}$ is satisfying, $\mathcal{M}_\phi$ outputs state 1. If $\boldsymbol{y}$ is unsatisfying, then $\mathcal{M}_\phi$ outputs state 0 with probability $\epsilon := 2^{-n^2}$ and outputs its input state i with probability $1-\epsilon$. It is clear that $\mathcal{A}$ can write down $\mathcal{M}_\phi$'s description in deterministic polynomial time. One can now check that the resulting Markov chains M_ϕ are as follows:

$$\text{if } \phi \text{ is satisfiable, } M_\phi = \begin{bmatrix} 1-2^{-n} & 2^{-n} \\ \epsilon' & 1-\epsilon' \end{bmatrix} \qquad (\text{where } \epsilon' := (1-2^{-n})\epsilon \approx 2^{-n^2});$$

$$\text{if } \phi \text{ is unsatisfiable, } M_\phi = \begin{bmatrix} 1 & 0 \\ \epsilon & 1-\epsilon \end{bmatrix}.$$

Now if ϕ is unsatisfiable, state 0 is absorbing and it's clear that the (unique) stationary distribution π of M_ϕ is entirely concentrated on state 0. On the other hand, suppose ϕ is satisfiable. Then since the $0 \to 1$ transition probability of M_ϕ is much higher (relatively speaking) than the $1 \to 0$ transition probability, the long-term (i.e., stationary) distribution π of M_ϕ will be almost entirely concentrated on state 1. (More precisely, π will put only probability $\frac{\epsilon'}{2^{-n}+\epsilon'} \approx 2^{-n^2+n}$ on state 0.) We now stipulate that for every ϕ, algorithm $\mathcal{A}$ outputs the same 1-bit decision circuit $\mathcal{D}_\phi$, which on input $\boldsymbol{i} \sim \pi$ simply outputs $\boldsymbol{i}$. From the above discussion, we see that this correctly indicates whether $\phi \in$ SAT except with negligible error probability.

The following theorems concerning $\mathsf{BPP}_{\mathsf{CTC}[w]}$ have previously been shown:

▶ **Theorem 4.** *(Aaronson–Watrous [5].)* $\mathsf{BPP}_{\mathsf{CTC}[\mathrm{poly}(n)]} = \mathsf{PSPACE}$.
(Indeed $\mathsf{P}_{\mathsf{CTC}[\mathrm{poly}(n)]} = \mathsf{BQP}_{\mathsf{QCTC}[\mathrm{poly}(n)]} = \mathsf{PSPACE}$*.)*

▶ **Theorem 5.** *(Say–Yakaryılmaz [31].)* $\mathsf{BPP}_{\mathsf{CTC}[1]} = \mathsf{BPP}_{\mathsf{path}}$.
(Indeed, adding 1 *CTC-bit generally confers the power of "postselection", discussed in Section 3. For example, it also holds that* $\mathsf{BQP}_{\mathsf{CTC}[1]} = \mathsf{PostBQP} = \mathsf{PP}$*.)*

As mentioned in Section 1.1 (and discussed further in Section 3), $\mathsf{BPP}_{\mathsf{path}}$ is likely equal to $\mathsf{P}_{||}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$, and is very likely to be much smaller than PSPACE.

2.1 Our theorem

Paraphrasing the above two theorems, we have that Markov chains with 2 states ($w = 1$) give the power of $\mathsf{BPP}_{\mathsf{path}}$, and Markov chains with exponentially many states ($w = \mathrm{poly}(n)$) give the power of PSPACE. What about in between? Are 3-state or 4-state ($w = 2$) Markov chains more powerful than 2-state chains? To take an analogy from another family of complexity classes, we remind the reader that it's widely believed that $\mathsf{P}^{\mathsf{NP}[1]} \subsetneq \mathsf{P}^{\mathsf{NP}[2]} \subsetneq \mathsf{P}^{\mathsf{NP}[3]} \subsetneq \cdots$ The main result of this paper is that in apparent contrast, "the hierarchy collapses" for $\mathsf{BPP}_{\mathsf{CTC}[w]}$; polynomially many states ($w = O(\log n)$) confer no more advantage than 2 states.

▶ **Main theorem.** $\mathsf{BPP}_{\mathsf{CTC}[O(\log n)]} \subseteq \mathsf{PostBPP} = \mathsf{BPP}_{\mathsf{path}}$*; thus*

$$\mathsf{BPP}_{\mathsf{CTC}[1]} = \mathsf{BPP}_{\mathsf{CTC}[2]} = \mathsf{BPP}_{\mathsf{CTC}[3]} = \cdots = \mathsf{BPP}_{\mathsf{CTC}[O(\log n)]} = \mathsf{BPP}_{\mathsf{path}}.$$

It will be clear from our proof that more generally, $O(\log n)$ CTC bits still only confer the power of postselection, and in particular $\mathsf{BQP}_{\mathsf{CTC}[O(\log n)]} = \mathsf{PostBQP} = \mathsf{PP}$. Our main theorem may also be seen as further demonstration of the robustness and naturalness of the class $\mathsf{BPP}_{\mathsf{path}}$.

2.2 Proof techniques

Here we briefly outline the proof of our theorem, with the actual proof being given in Section 4. Let's return to Example 3, which shows that $\mathrm{SAT} \in \mathsf{BPP}_{\mathsf{CTC}[1]}$. One might ask, why doesn't the proof show that $\mathrm{SAT} \in \mathsf{BPP}$? After all, the algorithm $\mathcal{A}$ simply constructs a 2-state Markov chain M and then takes a sample from its stationary distribution. Why doesn't $\mathcal{A}$ simply exactly solve for M's stationary distribution? The trouble of course is that even though $\mathcal{A}$ constructed M itself, in some sense M is still only "implicitly defined" from $\mathcal{A}$'s point of view. $\mathcal{A}$ cannot directly access the transition probabilities of M (doing so requires $\mathcal{A}$ to solve an NP-complete problem); rather, $\mathcal{A}$ can only "simulate" M, by use of the state-transition matrix $\mathcal{M}$ it constructed. Naively, this still might not seem like a problem; given the ability to simulate M, couldn't $\mathcal{A}$ find a (near-)stationary distribution π for M simply by simulating it for a long time? The trouble here is that even though M only has 2 states, it has some transition probabilities that are "exponentially small" (in n). Furthermore, the stationary distribution of M can be extremely sensitive to the *relative* exponential smallness of these transition probabilities – Example 3 illustrates exactly this.

Our proof that $\mathsf{BPP}_{\mathsf{CTC}[w]} \subseteq \mathsf{PostBPP} = \mathsf{BPP}_{\mathsf{path}}$ for $w \leq O(\log n)$ in some sense follows Say and Yakaryılmaz's proof [31] in the case of $w = 1$. They essentially observed that using the power of postselection (discussed further in Section 3), a randomized algorithm can get an exact sample from the stationary distribution of a Markov chain given only a state-transition oracle for it. Their proof of this was greatly facilitated by the fact that 2-state Markov chains are easy to analyze: If the $0 \to 1$ transition probability is p and the $1 \to 0$ transition probability is q, then the stationary distribution is $\pi = \begin{bmatrix} \frac{q}{p+q} \\ \frac{p}{p+q} \end{bmatrix}$. (This presumes we don't have $p = q = 0$, an important issue that we discuss later.) For our main theorem, we need a similar postselecting algorithm for general poly(n)-state Markov chains. The key technical tool for this will be the *Markov Chain Tree Theorem*, apparently first proved by Hill [23], and called by Aldous [6] "the most often rediscovered result in probability theory"; see also [36, 34, 24, 25, 7, 14, 28]. We state here the version for irreducible chains:

▶ **Markov Chain Tree Theorem.** *Let M be an S-state irreducible Markov chain with transition matrix $(p_{ij})_{i,j\in[S]}$. Let G_M be the underlying strongly connected digraph for M in which (i,j) is a directed edge if and only if $p_{ij} > 0$. Recall that a* rooted arborescence *T in G_M is a collection of edges forming a rooted spanning tree in which all edges are directed toward the root vertex. We write $\|T\| = \prod_{(i,j)\in T} p_{ij}$. Let $\mathcal{T}_i$ denote the set of all arborescences in G_M rooted at $i \in [S]$, and write $\mathcal{T} = \cup_i \mathcal{T}_i$. Then if π denotes the (unique) stationary distribution of M, we have the formula $\pi_i = \left(\sum_{T\in\mathcal{T}_i} \|T\|\right) \Big/ \left(\sum_{T\in\mathcal{T}} \|T\|\right)$.*

We add that this theorem plays an important role in the theory of exact sampling from unknown Markov chains [8, 27, 29, 38]. That theory is concerned with a problem similar to ours; however, there are two main differences: i) That theory involves only traditional algorithms, and therefore by necessity the running time may be exponential if the chain's mixing time is exponential. By contrast, we are using postselecting algorithms and therefore have the chance to run in polynomial time. ii) That theory is concerned with *exact* sampling

from the stationary distribution. By contrast, we actually only need approximate sampling from the stationary distribution.

Finally, we mention one challenge for our proof that at first seems like a technicality but in fact proves to be quite a nuisance: There is no promise in the definition of $\mathsf{BPP}_{\mathsf{CTC}[w]}$ that the Markov chains M_x be irreducible. This is precisely the "$p = q = 0$ issue" elided in the discussion of 2-state stationary distributions above. We overcome this difficulty by proving a somewhat technical lemma that allows us to perturb general Markov chains into irreducible ones.

2.3 Outline of the remainder of the paper

The aforementioned technical lemma on Markov chain perturbations, which allows us to work only with irreducible Markov chains, is omitted for reasons of space; it can be found in the arXiv version of the paper. In Section 3 we recall $\mathsf{BPP}_{\mathsf{path}}$ and postselection in more detail, and we also describe the "restarting" view of postselection (from [40]) that will be helpful in the proof of our main theorem. Finally, we give the proof of the main theorem in Section 4, and then end with an open question.

3 $\mathsf{BPP}_{\mathsf{path}}$, postselection, and restarts

In this section we describe three different viewpoints on the class $\mathsf{BPP}_{\mathsf{path}}$.

The complexity class $\mathsf{BPP}_{\mathsf{path}}$ was originally defined by Han, Hemaspaandra, and Thierauf [22, 21], in a paper also concerned with certain cryptographic problems. (It was also independently defined much later in a paper by Aspnes, Fischer, Fischer, Kao, and Kumar [9, 10] on the computational complexity of the stock market.) We quote Fortnow's explanation of the original definition when he named it "Complexity Class of the Week" [17]:

"Let us call a nondeterministic Turing machine M balanced if for every input x, all of its computational paths have the same length. [We can define the] class BPP as follows: L is in BPP if there is a balanced nondeterministic polynomial-time M such that:

- If x is in L then there are at least twice as many accepting as rejecting paths of $M(x)$.
- If x is not in L then there are at least twice as many rejecting as accepting paths of $M(x)$.

Suppose we use the same definition without the "balanced" requirement. This gives us the class $\mathsf{BPP}_{\mathsf{path}}$."

Interestingly, the analogous class "$\mathsf{PP}_{\mathsf{path}}$" – for which $x \in L$ iff $M(x)$ has more accepting than rejecting (unbalanced) paths – was defined much earlier in Simon's 1975 thesis [35]. Simon showed that $\mathsf{PP}_{\mathsf{path}}$ is equal to the class PP (which had recently been defined by Gill [19]). By way of contrast, $\mathsf{BPP}_{\mathsf{path}}$ is very unlikely to equal BPP, as it is known [22] that $\mathsf{BPP}_{\mathsf{path}}$ contains both MA and $\mathsf{P}_{||}^{\mathsf{NP}}$. $\mathsf{BPP}_{\mathsf{path}}$ is also known [22] to be contained in $\mathsf{BPP}_{||}^{\mathsf{NP}}$. Indeed, under the standard complexity assumptions used to derandomize AM, Shaltiel and Umans [32] showed that $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}_{||}^{\mathsf{NP}}$. For a related class known as SBP, which sits between MA and $\mathsf{BPP}_{\mathsf{path}}$, see [13, 12].

Another characterization of $\mathsf{BPP}_{\mathsf{path}}$ was given by Aaronson, via the notion of adding *postselection* (see [4]) to a complexity class. Suppose that you have a probabilistic algorithm that can end in three kinds of final states: accepting, rejecting, and indecisive. We assume the probability of ending in a decisive state is guaranteed to be nonzero. "Postselection" refers to the (nonrealistic) ability to *condition* the computation on ending in a decisive state. This yields probabilistic computation with just the usual two kinds of final states. For example, one says that $L \in \mathsf{PostBPP}$ if there is a polynomial-time randomized Turing

machine as described above which, for each input x, gives the correct answer about $x \in L$ with probability at least $\frac{2}{3}$, *conditioned* on not ending in an indecisive state. More generally, if C is a probabilistic or quantum complexity class, PostC is the class of languages decided by C-machines with the ability to postselect on ending in a decisive state. In [1], Aaronson proved that $\mathsf{PostBQP} = \mathsf{PP}$; later [2], he observed that $\mathsf{PostBPP} = \mathsf{BPP}_{\mathsf{path}}$.

In the derivation of our main theorem we will prefer a third perspective on $\mathsf{BPP}_{\mathsf{path}}$ and postselection, introduced by Yakaryılmaz and Say [40]: that of randomized algorithms with *restarts*. For some probabilistic complexity class C, suppose again that we have C-machines that can end in one of three states: accept, reject, or indecisive. We think of the third state as the *restart* state, imagining that whenever the C-machine enters such a state, it immediately restarts its computation from the initial configuration, using no information that it may have gathered up to that point.[2] As observed in [40], the class of languages decided by such a machine is again PostC. In particular, $\mathsf{BPP}_{\mathsf{path}}$ is the class of languages that are decided by bounded-error probabilistic polynomial-time Turing machines with this ability to restart. This perspective seems most useful for algorithm design. As an illustration, we believe it is fairly "obvious" that the following restarting-algorithm decides SAT with very high probability, thereby showing $\mathsf{NP} \subseteq \mathsf{BPP}_{\mathsf{path}}$:

> "On input formula ϕ with n variables, randomly choose an assignment $\boldsymbol{y}$.
> If $\boldsymbol{y}$ satisfies ϕ, accept. (1)
> Otherwise, restart with probability $1 - 2^{-n^2}$ and reject with probability 2^{-n^2}."

The reader may compare (1) with the 1-bit CTC algorithm for SAT from Example 3, which also shows $\mathsf{NP} \subseteq \mathsf{BPP}_{\mathsf{path}}$ in light of Say and Yakaryılmaz's Theorem 5, $\mathsf{BPP}_{\mathsf{CTC}[1]} = \mathsf{BPP}_{\mathsf{path}}$.

▶ Remark. For all three definitions of $\mathsf{BPP}_{\mathsf{path}}$ described above, it is easy to see that the "$\frac{2}{3}$ cutoff" for success could equivalently be an "α cutoff" for any fixed constant $\frac{1}{2} < \alpha < 1$, just as is the case for the class BPP.

3.1 Remarks on random coins for $\mathsf{BPP}_{\mathsf{path}}$ algorithms

In informal descriptions of randomized algorithms, it's typical to make statements like, "Next, with probability $\frac{1}{3}$ the algorithm…" Such statements sweep a well-known, minor detail under the rug; namely, the traditional BPP model (based on nondeterministic branching) only has "access" to probability-$\frac{1}{2}$ coin flips. Of course, this is not an essential problem, since one can simulate a $\frac{1}{3}$-biased coin flip to error δ in time $O(\log \frac{1}{\delta})$, and δ needn't be smaller than $1/\mathrm{poly}(n)$. Here we remark that in the context of *restarting* algorithms, the problem is not just inessential, it's literally no problem at all:

▶ **Lemma 6.** *A restarting randomized algorithm can simulate a p-biased coin flip* exactly *in time $O(\langle p \rangle)$, and can simulate a uniformly random draw from $[n]$* exactly *in time $O(\log n)$.*

Proof. We give the simplest example, leaving the general case for the reader. Suppose we wish to draw $\boldsymbol{r} \sim [3]$ uniformly at random. We toss two probability-$\frac{1}{2}$ coins, forming a 2-bit integer $0 \leq \boldsymbol{r} \leq 3$. Then if $\boldsymbol{r} = 0$, we restart. ◀

On the other hand, it's also important to remember that time $\Omega(\langle p \rangle)$ is also *required* to flip a p-biased coin; for example, in algorithm (1) the step "reject with probability 2^{-n^2} else restart" takes n^2 time steps.

---

[2] Note that the new algorithm obtained in this manner will in general have unbounded runtime, even if C is a time-bounded class.

4 Proof of the main theorem

We now give the proof of our main theorem, that $\mathsf{BPP}_{\mathsf{CTC}[O(\log n)]} \subseteq \mathsf{BPP}_{\mathsf{path}}$.

Proof. Let $L \in \mathsf{BPP}_{\mathsf{CTC}[O(\log n)]}$; say L is defined by algorithm $\mathcal{A}$ as in Definition 2. Thus there are constants $c_1, c_2, c_3 \in \mathbb{N}$ such that on inputs $x \in \{0,1\}^n$, algorithm $\mathcal{A}$ outputs state-transition circuits $\mathcal{M}_x$ of size $O(n^{c_1})$ defining Markov chains M_x on $S = O(n^{c_2})$ states, as well as decision circuits $\mathcal{D}_x$ of size $O(n^{c_3})$. Further, we have that for each x, if π is *any* stationary distribution for M_x and $\boldsymbol{i} \sim \pi$, then

$$\mathbf{Pr}[\mathcal{D}_x(\boldsymbol{i}) = 1_{\{x \in L\}}] \geq \tfrac{2}{3}. \tag{2}$$

Our goal will be to define a polynomial-time randomized restarting algorithm $\mathcal{R}$ that has

$$\mathbf{Pr}[\mathcal{R}(x) = 1_{\{x \in L\}}] \geq 0.65 \tag{3}$$

for all x. As discussed in Section 3, this will show that $L \in \mathsf{BPP}_{\mathsf{path}}$, as required.

On input $x \in \{0,1\}^n$, the first step of algorithm $\mathcal{R}$ involves invoking a technical lemma to convert to an irreducible Markov chain. This lemma (whose proof is omitted from this extended abstract) involves replacing the transition matrix K of the chain with $K' = (1-\epsilon)K + \epsilon\frac{1}{S}J$, where S is the number of states in the chain and J is the all-1's matrix. Here the exact value $\epsilon = 2^{-\mathrm{poly}(n)}$ will be described later. More precisely, $\mathcal{R}$ first simulates $\mathcal{A}$ to get state-transition oracle circuit $\mathcal{M}_x$ for Markov chain M_x. It then constructs a state-transition oracle circuit $\mathcal{M}'_x$ for the irreducible perturbed chain M'_x, using the description of $\mathcal{M}_x$ in a black-box fashion. Let π' denote the stationary distribution for M'_x and let K denote the transition matrix for M_x. By definition, K is square matrix of dimension $S \leq O(n^{c_2})$ in which each entry is an integer multiple of $2^{-\mathrm{size}(M_x)} = 2^{-O(n^{c_1})}$. It follows that $\langle K \rangle \leq O(n^{c_4})$ for some constant $c_4 \in \mathbb{N}$. We will now specify that $\epsilon = 2^{-bn^b}$ for a sufficiently large constant b depending on c_1, c_2, and a constant from the aforementioned technical lemma. Then that lemma implies

$$\|\pi - \pi'\|_1 \leq .01 \tag{4}$$

for *some* stationary distribution π of M_x. It is easy to see that with this choice of ϵ, algorithm $\mathcal{R}$ can construct $\mathcal{M}'_x$ from $\mathcal{M}_x$ in $\mathrm{poly}(n)$ time. (Here we use the fact that ϵ is "only" exponentially small; cf. the last paragraph in Section 3.1.)

The remainder of the proof is devoted to showing that $\mathcal{R}$ can obtain an *exact* sample $\boldsymbol{r} \sim \pi'$. Having shown this, we only need to let $\mathcal{R}$ simulate $\mathcal{A}$ to get $\mathcal{D}_x$, and then output $\mathcal{D}_x(\boldsymbol{r})$. Then combining (4) with (2) shows that (3) holds for all x, as required.

We now exhibit the subroutine which the restarting-algorithm $\mathcal{R}$ will use to obtain an exact sample $\boldsymbol{r} \sim \pi'$ (the unique stationary distribution of irreducible chain M'_x):

- Choose a uniformly random labeled, rooted, undirected tree $\boldsymbol{T}$ on vertex set $[S]$. This can be done exactly (i.e., with each $\boldsymbol{T}$ occurring with probability $1/S^{S-1}$) in $\mathrm{poly}(S) = \mathrm{poly}(n)$ time, by choosing a uniformly random Prüfer Code [30, 37] in $[S]^{S-2}$, converting it to a tree $\boldsymbol{T}$, and then choosing a random vertex $\boldsymbol{r}$ of $\boldsymbol{T}$ to be the root.[3]
- Make $\boldsymbol{T}$ into a rooted arborescence $\vec{\boldsymbol{T}}$ by directing all edges toward the root $\boldsymbol{r}$.
- For each directed edge $(i,j) \in \vec{\boldsymbol{T}}$, simulate $\mathcal{M}'_x(i)$ and "check" if the output is j. If the check fails, restart.
- If all $S-1$ checks pass, halt with output $\boldsymbol{r}$.

[3] Here we may use restarting to get exactly random samples from $[S]$; see Lemma 6.

The fact that this subroutine restarts with probability strictly less than 1 follows from the fact that M'_x is irreducible; indeed, its underlying digraph $G = G_{M'_x}$ is the complete digraph. The probability P_r that this subroutine outputs $\boldsymbol{r} = r$ without encountering any restarts is precisely

$$P_r = \sum_{r\text{-rooted arborescences } T} \frac{1}{S^{S-1}} \prod_{(i,j)\in T} p_{ij},$$

where p_{ij} denotes the transition probability from i to j in the Markov chain M'_x. It follows that the probability of $\mathcal{R}$ finally outputting r (when restarts are taken into account) is

$$\frac{P_r}{\sum_{r\in[S]} P_r} = \frac{\sum_{r\text{-rooted arborescences } T} \prod_{(i,j)\in T} p_{ij}}{\sum_{\text{arborescences } T} \prod_{(i,j)\in T} p_{ij}}.$$

By the Markov Chain Tree Theorem, this is indeed precisely the probability $\pi'(r)$ of r under the stationary distribution of M'_x. ◀

We conclude this section by observing that besides the power of restarting, algorithm $\mathcal{R}$ only really needed the power to simulate the state-transition circuits $\mathcal{M}_x$ and the decision circuits $\mathcal{D}_x$. For example, if these were standard quantum circuits, it would suffice for $\mathcal{R}$ to be a quantum algorithm. Thus we may also conclude $\mathsf{BQP}_{\mathsf{CTC}[O(\log n)]} \subseteq \mathsf{PostBQP} = \mathsf{PP}$.

We also add that Say and Yakaryılmaz [31] studied various models of *finite automata* augmented with 1-bit CTCs; they showed that this augmentation causes both probabilistic and quantum finite automata to become as powerful as their respective postselected versions. The technique used in the proof of our main result can be simplified easily to show that no additional gain arises when these machines are augmented with larger constant-width CTCs.

5 Conclusion

A very interesting open question left by our work is one also raised at the end of [31]:

What is the computational power conferred by 1 time-traveling *qubit*?

Answering this question precisely would seem to require a good understanding of stationary density matrices for 1-qubit quantum channels. As mentioned in Section 1.1, we are not aware of any work showing that time-traveling qubits confer more computational power than time-traveling bits.

Acknowledgment. The first author would like to thank the Boğaziçi University Computer Engineering Department for their hospitality.

References

1 Scott Aaronson. Is quantum mechanics an island in Theoryspace? Technical Report quant-ph/0401062, arXiv, 2004.

2 Scott Aaronson. *Limits on Efficient Computation in the Physical World.* PhD thesis, University of California, Berkeley, 2004.

3 Scott Aaronson. NP-complete problems and physical reality. *ACM SIGACT News*, 36(1):30–52, 2005.

4 Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A*, 461(2063):3473–3482, 2005.

5 Scott Aaronson and John Watrous. Closed timelike curves make quantum and classical computing equivalent. *Proceedings of the Royal Society A*, 465(2102):631–647, 2009.

6 David Aldous. Reversible Markov chains and random walks on graphs, 2002. `http://www.stat.berkeley.edu/~aldous/RWG/book.pdf`.

7 Venkat Anantharam and Pantelis Tsoucas. A proof of the Markov chain tree theorem. *Statistics & Probability Letters*, 8(2):189–192, 1989.

8 Søren Asmussen, Peter Glynn, and Hermann Thorisson. Stationarity detection in the initial transient problem. *ACM Transactions on Modeling and Computer Simulation*, 2(2):130–157, 1992.

9 James Aspnes, David Fischer, Michael Fischer, Ming-Yang Kao, and Alok Kumar. Towards understanding the predictability of stock markets from the perspective of computational complexity. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2001.

10 James Aspnes, David Fischer, Michael Fischer, Ming-Yang Kao, and Alok Kumar. Towards understanding the predictability of stock markets from the perspective of computational complexity. In *New Directions in Statistical Physics*, pages 129–151. Springer Berlin Heidelberg, 2004.

11 Dave Bacon. Quantum computational complexity in the presence of closed timelike curves. *Physical Review A*, 70(3):032309, 2004.

12 Elmar Böhler, Christian Glaßer, and Daniel Meister. Small bounded-error computations and completeness. Technical Report 69, Electronic Colloquium on Computational Complexity, 2003.

13 Elmar Böhler, Christian Glaßer, and Daniel Meister. Error-bounded probabilistic computations between MA and AM. *Journal of Computer and System Sciences*, 72(6):1043–1076, 2006.

14 Andrei Broder. Generating random spanning trees. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 442–447, 1989.

15 Todd Brun. Computers with closed timelike curves can solve hard problems efficiently. *Foundations of Physics Letters*, 16(3):245–253, 2003.

16 David Deutsch. Quantum mechanics near closed timelike lines. *Physical Review D*, 44(10):3197–3217, 1991.

17 Lance Fortnow. Complexity class of the week: $\mathrm{BPP}_{\mathrm{path}}$, February 2003. `http://blog.computationalcomplexity.org/2003/02/complexity-class-of-week-bpppath.html`.

18 John Friedman, Michael Morris, Igor Novikov, Fernando Echeverria, Gunnar Klinkhammer, Kip Thorne, and Ulvi Yurtsever. Cauchy problem in spacetimes with closed timelike curves. *Physical Review D*, 42(6):1915–1930, 1990.

19 John Gill. Computational complexity of probabilistic Turing machines. In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing*, pages 91–95, 1974.

20 Kurt Gödel. An example of a new type of cosmological solutions of Einstein's field equations of gravitation. *Reviews of Modern Physics*, 21(3):447–450, 1949.

21 Yenjo Han, Lane Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, 26(1):59–78, 1997.

22 Yenjo Hem, Lane Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. In *Proceedings of the 4th Annual International Symposium on Algorithms and Computation*, pages 230–239, 1993.

23 Terrell Hill. Studies in irreversible thermodynamics IV: diagrammatic representation of steady state fluxes for unimolecular systems. *Journal of Theoretical Biology*, 10(3):442–459, 1966.

24 Hans-Helmut Kohler and Eva Vollmerhaus. The frequency of cyclic processes in biological multistate systems. *Journal of Mathematical Biology*, 9(3):275–290, 1980.

25 F. Thomson Leighton and Ronald Rivest. The Markov Chain Tree Theorem. Technical Report TM-249, Massachusetts Institute of Technology, 1983.

26 Seth Lloyd, Lorenzo Maccone, Raul Garcia-Patron, Vittorio Giovannetti, and Yutaka Shikano. The quantum mechanics of time travel through post-selected teleportation. Technical Report 1007.2615, arXiv, 2010.

27 László Lovász and Peter Winkler. Exact mixing in an unknown Markov chain. *Electronic Journal of Combinatorics*, 2:Research Paper 15, 1995.

28 Piotr Pokarowski. Directed forests with application to algorithms related to Markov chains. *Applicationes Mathematicae*, 26(4):395–414, 1999.

29 James Propp and David Wilson. How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms*, 27(2):170–217, 1998.

30 Heinz Prüfer. Neuer Beweis eines Satzes über Permutationen. *Archiv der Mathematik und Physik*, 27:742–744, 1918.

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

32 Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006.

33 Yaoyun Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computing. *Quantum Information & Computation*, 3(1):84–92, 2003.

34 Bruno Shubert. A flow-graph formula for the stationary distribution of a Markov chain. *Institute of Electrical and Electronics Engineers. Transactions on Systems, Man, and Cybernetics*, SMC-5(5):565–566, 1975.

35 Janos Simon. *On some central problems in computational complexity*. PhD thesis, Cornell University, 1975. TR 75-224.

36 Alexander Wentzell and Mark Freidlin. On small random perturbations of dynamical systems. *Russian Mathematical Surveys*, 25(1):1–55, 1970.

37 Wikipedia. Prüfer sequence, July 2014. `http://en.wikipedia.org/wiki/Prufer_sequence`.

38 David Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 296–303, 1996.

39 Michael Wolf. Quantum channels & operations: guided tour, 2012. `http://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MichaelWolf/QChannelLecture.pdf`.

40 Abuzer Yakaryılmaz and A. C. Cem Say. Proving the power of postselection. *Fundamenta Informaticae*, 123(1):107–134, 2013.