Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 147 (2016)

The weakness of CTC qubits and the power of approximate counting

Ryan O'Donnell* A. C. Cem Say†

April 7, 2015

Abstract

We present two results in structural complexity theory concerned with the following interrelated topics: computation with postselection/restarting, closed timelike curves (CTCs), and approximate counting. The first result is a new characterization of the lesser known complexity class $\mathsf{BPP}_{\mathsf{path}}$ in terms of more familiar concepts. Precisely, $\mathsf{BPP}_{\mathsf{path}}$ is the class of problems that can be efficiently solved with a nonadaptive oracle for the *Approximate Counting* problem. Our second result is concerned with the computational power conferred by CTCs; or equivalently, the computational complexity of finding stationary distributions for quantum channels. We show that any poly(n)-time quantum computation using a CTC of $O(\log n)$ qubits may as well just use a CTC of 1 classical bit. This result essentially amounts to showing that one can find a stationary distribution for a poly(n)-dimensional quantum channel in PP.

*Department of Computer Science, Carnegie Mellon University. Work performed while the author was at the Boğaziçi University Computer Engineering Department, supported by Marie Curie International Incoming Fellowship project number 626373.

†Boğaziçi University Computer Engineering Department.

ISSN 1433-8092

1 Introduction

It is well known that studying "non-realistic" augmentations of computational models can shed a great deal of light on the power of more standard models. The study of nondeterminism and the study of relativization (i.e., oracle computation) are famous examples of this phenomenon. Let us describe two more recently introduced such examples.

Postselection. The first involves the operation of *postselection*, which can be applied to any probabilistic computation class C to produce a new class PostC. In PostC, we assume a C-computation ends with *two* computed bits, and the final output is the value of the second bit conditioned on the first bit being 1. This model was apparently first considered by Aspnes et al. [AFF+01], under the name of *conditional probabilistic computation*, in the context of stock market prediction. They studied the most basic postselected class $\mathsf{PostBPP}$ (which they called BCPP), and also observed that $\mathsf{PostPP} = \mathsf{PP}$. Later, Böhler et al. [BGM03] recognized that $\mathsf{PostBPP}$ is the same as the class $\mathsf{BPP}_{\mathsf{path}}$ defined earlier by Han et al. [HHT93, HHT97]; indeed, postselection is essentially equivalent to the "path" operator from [HHT93] (and $\mathsf{PostPP} = \mathsf{PP}_{\mathsf{path}} = \mathsf{PP}$ was already proved by Simon [Sim75], in one of the first ever works on randomized complexity). Postselection was independently introduced (and named) by Aaronson [Aar04a, Aar05b] in the context of quantum computation. Aaronson showed that the class $\mathsf{PostBQP}$ coincides with the classical class PP; given this, the notable theorem of [BRS95] that PP is closed under intersection is an essentially trivial consequence [Aar04b]. Aaronson [Aar04b] also independently observed that $\mathsf{PostBPP} = \mathsf{BPP}_{\mathsf{path}}$. Whereas adding postselection to ZPP, RP, and BQP yields well-known classes ($\mathsf{NP} \cap \mathsf{coNP}$ [SY12], NP [HHT97], and PP [Aar04b]), the most basic postselected class $\mathsf{PostBPP} = \mathsf{BPP}_{\mathsf{path}}$ is, intriguingly, not widely understood. Perhaps the most notable feature of $\mathsf{BPP}_{\mathsf{path}}$ is the important role it plays in proofs of "quantum supremacy" for sampling problems [BJS10, AA11].

CTC computation. Another example of a non-realistic model of computation leading to interesting questions and answers about standard models is that of computation in the presence of *closed timelike curves* (CTCs) — informally, computation with time-traveling bits.[1] Deutsch [Deu91] introduced this model, and it was formally codified in [Bac04]. Aaronson and Watrous [AW09] showed that the ability to compute with polynomially many time-traveling bits "collapses everything to PSPACE". In particular, P (or even AC^0) augmented with polynomially many classical CTC bits can compute everything in PSPACE; conversely, giving BQP (or even PSPACE) polynomially many CTC *qubits* does not afford any computational power beyond PSPACE. Since polynomially many CTC bits wipes out all computational distinctions between P and PSPACE, it's natural to look at the more refined (and theoretically "more realistic") question of the power of a small number of time-traveling bits. To this end, Say and Yakaryılmaz [SY12] showed that a single classical CTC bit exactly confers the power of postselection, boosting BQP to PP and BPP to $\mathsf{BPP}_{\mathsf{path}}$. The present authors [OS14] recently extended this statement to allow for up to logarithmically many classical CTC bits.

In the above examples we have seen applications of non-realistic computational models to the study of more standard complexity classes. On one hand, these classes — PP, $\mathsf{BPP}_{\mathsf{path}}$, $\mathsf{PostRP} = \mathsf{NP}$, and so forth — are themselves typically defined in terms of non-realistic computational models. While we don't feel obligated to further justify the study of basic classes like PP and NP, we do take

[1] While time travel may seem like the height of non-realism, we mention that the existence of CTCs *is* apparently consistent with Einstein's theory of general relativity, as was first pointed out by Kurt Gödel [Göd49].

this opportunity to recall that a complexity class is generally worthy of study if it's the smallest class known to contain some interesting/natural computational task. For example, counting the number of satisfying assignments to a formula or a circuit is a fundamental computation task, and this is a main motivation for studying PP. Similarly, we view the study of time-traveling qubits (respectively, bits) as worthwhile because it's precisely equivalent to a natural computational question: What is the complexity of determining a stationary distribution for a quantum channel (respectively, Markov chain) specified by a polynomial-size circuit? Tying together much of the above discussion, the authors' recent result [OS14] may be viewed as using reasoning about postselection to show that the complexity of finding stationary distributions of Markov chains is precisely $\mathsf{BPP}_{\mathsf{path}}$.

1.1 Our results

This paper has two main contributions.

1. Characterizing $\mathsf{BPP}_{\mathsf{path}}$. Our first contribution is concerned with the class $\mathsf{BPP}_{\mathsf{path}}$. We show that instead of a definition in terms of path operators, or postselection, or narrow CTCs carrying classical bits, the class $\mathsf{BPP}_{\mathsf{path}}$ can be precisely characterized in terms of very familiar concepts from complexity theory. Specifically:

> $\mathsf{BPP}_{\mathsf{path}}$ is equivalent to efficient computation augmented with the ability to solve the *Approximate Counting* problem, nonadaptively.

Recall that the well known Approximate Counting problem — introduced by Sipser [Sip83] and Stockmeyer [Sto83, Sto85] — is the task of 2-approximating the number of satisfying assignments to a given circuit or formula. Indeed, we show that in this characterization, any polynomial number of nonadaptive calls to an Approximate Counting oracle can efficiently be reduced to just 2 calls (and indeed, just 1 call to a related problem called **ApproxCountRatio**). Thus our first main theorem may be stated as follows:

Theorem 1.1. $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}_{\|}^{\mathbf{ApproxCount}} = \mathsf{P}_{\|}^{\mathbf{ApproxCount}[2]}$.

Given the understanding of $\mathsf{BPP}_{\mathsf{path}}$ as of 2012, the proof of Theorem 1.1 is not too challenging. Still, we find it to be a memorable and novel characterization of the class $\mathsf{BPP}_{\mathsf{path}}$.

Before moving on to discuss our second contribution, we remark that the *nonadaptive* use of the **ApproxCount** oracle in Theorem 1.1 is important. Indeed, it's unlikely that $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}^{\mathbf{ApproxCount}}$; the reason is that under a derandomization assumption (see, e.g., [SU06]) we have $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}_{\|}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[\log]}$, and it is widely believed that $\mathsf{P}^{\mathsf{NP}[\log]} \subsetneq \mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathbf{ApproxCount}}$. A similar distinction can be made in the context of PP, using two famous theorems in complexity theory: On one hand, $\mathsf{P}_{\|}^{\mathsf{PP}} = \mathsf{PP}$ (due to Fortnow and Reingold [FR96], building on the Beigel–Reingold–Spielman theorem [BRS95]); on the other hand, P^{PP} contains the (presumably) much larger class PH (Toda's theorem [Tod91]).

2. The complexity of quantum channel fixed points. The second contribution of this paper is concerned with the computational advantage conferred by a small number of "time-traveling qubits". We recall formal definitions in Section 4; for now, let's briefly say that $\mathsf{C}_{\mathsf{QCTC}[w]}$ denotes the complexity class C augmented with a CTC supporting $w = w(n)$ qubits. Roughly speaking, this is equivalent to adding the ability to write down a quantum circuit defining a 2^w-dimensional quantum channel and then freely get one sample from a stationary density matrix for the channel.

Similarly, we use notation $\mathsf{CTC}[w]$ for the case of a CTC carrying w classical bits; in this case, the channel simply becomes a 2^w-state Markov chain.

As mentioned earlier, Aaronson and Watrous [AW09] showed that CTCs supporting $w = \mathrm{poly}(n)$ (qu)bits give rather extravagant computational power, collapsing everything to PSPACE; i.e.,

$$\mathsf{P}_{\mathsf{CTC}[\mathrm{poly}]} = \mathsf{BQP}_{\mathsf{QCTC}[\mathrm{poly}]} = \mathsf{PSPACE}.$$

The main technical result in that paper is the inclusion $\mathsf{BQP}_{\mathsf{QCTC}[\mathrm{poly}]} \subseteq \mathsf{PSPACE}$, which in turn essentially boils down to showing that given a $\mathrm{poly}(n)$-qubit quantum channel defined by a $\mathrm{poly}(n)$-size quantum circuit, one can compute a stationary density matrix for the channel within PSPACE. The main concrete question left open by [AW09] was to determine the computational power conferred by "narrow (bounded-width) CTCs"; for example, they pointed out that already $\mathsf{NP} \subseteq \mathsf{BPP}_{\mathsf{CTC}[1]}$, improving on a similar earlier result of Bacon [Bac04] involving 1 CTC qubit. This question was essentially resolved in the case of CTCs carrying *classical* bits; as mentioned earlier, Say and Yakaryılmaz [SY12] showed

$$\mathsf{BQP}_{\mathsf{CTC}[1]} = \mathsf{PP}, \qquad \mathsf{BPP}_{\mathsf{CTC}[1]} = \mathsf{BPP}_{\mathsf{path}},$$

and the present authors [OS14] extended the above to allow even for $O(\log n)$ classical CTC bits.

Interestingly, to this point we do not know any setting in which a CTC supporting qubits is more powerful than one supporting the same number of classical bits. Indeed, the second contribution of this paper is to show that time-traveling qubits are *not* more powerful than classical ones in the case of narrow CTCs. In fact, something much stronger is true:

> If a problem can be solved by a quantum computer with the ability to send a $O(\log n)$-qubit message back in time, then the computer can do the same task while compressing the message to a single classical bit.

More precisely:

Theorem 1.2. $\mathsf{BQP}_{\mathsf{QCTC}[\log]} = \mathsf{BQP}_{\mathsf{CTC}[1]} = \mathsf{PP}$.

We remark that prior to this result, even the power of 1 CTC qubit was not understood; we only knew the bounds $\mathsf{PP} \subseteq \mathsf{BQP}_{\mathsf{QCTC}[1]} \subseteq \mathsf{PSPACE}$. The main task in proving Theorem 1.2 is showing that given an $O(\log n)$-qubit channel specified by a $\mathrm{poly}(n)$-size quantum circuit, one can *approximately* compute a stationary density matrix for the channel within PP. Our proof of this follows the work of Aaronson and Watrous somewhat closely, but uses "counting complexity technology" in place of "parallel linear algebra technology". In particular, we require the PP-analogue of our first result, Theorem 1.1.

1.2 Some philosophical remarks with which the reader may disagree

Let us make some brief, argumentative, remarks suggesting that a complexity class like $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}_{\parallel}^{\mathbf{ApproxCount}}$ may not be so "unrealistic" after all. Of course, a single oracle call to **ApproxCount** can decide **SAT**, which already may seem unrealistic. On the other hand, some researchers in the SAT-solving community will claim that solving **SAT** efficiently is no big deal (!). Indeed it is true that very powerful SAT-solvers now exist [SE14], with the empirical ability to efficiently decide **SAT** on many "naturally occurring" instances (whatever exactly that means).

On the other hand we know that if NP actually equals P then so does PH; yet, despite some practical efforts towards QBF-solving [PPT+10], it seems much rarer to hear claims from the

SAT-solving community that PH is easy. Perhaps the reason is related to the following classic complexity-theory puzzle: if Alice invents an efficient computer program solving **SAT** then we know she can put PH in P; but, if she merely gives Bob the ability to use her program, he can only compute P^{NP} (or $\mathsf{BPP}^{\mathsf{NP}}$). The resolution of this puzzle is of course that Alice applies her **SAT**-solving algorithm to *its own code*. Arguably, this produces rather "unnatural" **SAT** instances, and explains why SAT-solving practitioners do not seem to try the same trick.

On the other hand, SAT-solving practitioners *do* try to use their "heuristic **SAT** oracles" in a more modest way to try to solve problems beyond NP. For example, in a prizewinning paper from 2006, Gomes, Sabharwal, and Selman [GSS06] showed some success in boosting practical **SAT**-solvers to practical **ApproxCount**-solvers using the Valiant–Vazirani-based $\mathsf{BPP}^{\mathsf{NP}}$ algorithm. It is this sort of result which leads us to suggest that the study of algorithms within "unrealistically large" classes like $\mathsf{BPP}^{\mathsf{NP}}$ may in fact have practical consequences. Indeed, we feel that our characterization $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}_{\|}^{\mathbf{ApproxCount}}$ shows that $\mathsf{BPP}_{\mathsf{path}}$ is even *more* "practical", as restriction to a nonadaptive oracle precludes some of the more tricky/unnatural queries that might arise from adaptive oracle use. In fact, as we showed, $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}_{\|}^{\mathbf{ApproxCount}[2]}$; hence $\mathsf{BPP}_{\mathsf{path}}$ algorithms only require *two* queries to a "heuristic **ApproxCount** oracle", possibly further increasing their applicability in practice.

2 Preliminaries

In this section we introduce some terminology and definitions used in the remainder of the paper.

We assume knowledge of standard complexity classes such as P, BPP, NP, PP, GapP, and also BQP (discussed further below). We recall that for $k \in \mathbb{Z}^+$, the notation $\mathsf{C}_{\|,k}^{L}$ refers to languages decided by a C-machine with k "rounds" of nonadaptive queries to a computational problem L. Most frequently $k = 1$, in which case the notation is simply $\mathsf{C}_{\|}^{L}$. We also use the standard notation $\mathsf{C}_{\|}^{L[k]}$ to denote that at most k (nonadaptive) queries may be made to L. Here if $k = 1$ then there is no distinction between nonadaptive and adaptive oracle access.

Regarding quantum computation, we will write $\imath = \sqrt{-1}$ and $\delta(\rho, \sigma)$ for the trace distance between density matrices ρ and σ. Recall that the trace distance is also equal to the maximum probability with which any measurement can distinguish ρ and σ.

2.1 Definitions for approximate counting

If C is a one-output Boolean circuit we write $\#C$ to denote the number of inputs that it accepts. For $\alpha \in \mathbb{R}$ we write $\mathrm{sgn}(\alpha) \in \{-1, 0, +1\}$ for the sign of α.

Definition 2.1. Given $\alpha, \beta \in \mathbb{R}$ and $r \geq 1$, we say that β *is an r-approximation of* α, denoted $\alpha \overset{\times r}{\approx} \beta$ or $\beta \overset{\times r}{\approx} \alpha$, if and only if both $\mathrm{sgn}(\alpha) = \mathrm{sgn}(\beta)$ and

$$\frac{1}{r} \cdot |\beta| \leq |\alpha| \leq r \cdot |\beta|.$$

In the special case of $r = 2$ we will write simply $\alpha \approx \beta$. Note that regardless of r, if one of α or β is 0, the other must be too.

Definition 2.2. ApproxCountRatio is the following following computational task: Given as input the description of two Boolean circuits C, D, report (the standard encoding of) a rational number $\alpha \geq 0$ such that

$$\alpha \quad \approx \quad \frac{\#C}{\#D},$$

or report "$\#D = 0$". **ApproxCountDifference** is the same computational task except the requirement is to report $\alpha \in \mathbb{Q}$ such that

$$\alpha \quad \approx \quad \#C - \#D.$$

Finally, a slightly more standard version of **ApproxCountRatio** is what is typically called the "approximate counting problem", **ApproxCount**: given C output α such that $\alpha \approx \#C$.

Remark 2.3. **ApproxCount** and **ApproxCountRatio** are easily interreducible. One one hand, **ApproxCount** $\leq_m$ **ApproxCountRatio**, simply by setting D to be a trivial circuit with $\#D = 1$. In the other direction we need two nonadaptive queries:

$$\textbf{ApproxCountRatio} \in \mathsf{P}_{\|}^{\textbf{ApproxCount}[2]}$$

because we can simply approximate $\#C$ and $\#D$ separately and then report the ratio (or that $\#D = 0$). Actually, for this to work we need to be able to do **ApproxCount** with $\sqrt{2}$-approximation, rather than 2-approximation; but this is easy by Stockmeyer's trick, explained below.

Finally, it's clear that **ApproxCountDifference** is at least as hard these problems: specifically, **ApproxCount** $\leq_m$ **ApproxCountDifference**, simply by setting D to a trivial circuit with $\#D = 0$.

Remark 2.4. *(Stockmeyer's trick [Sto85, Theorem 3.1].)* We will also require certain generalizations of these problems. In **StrongApproxCount**, there is an additional input "precision parameter" $p \in \mathbb{Z}^+$, written in unary; the requirement then is that

$$\alpha \quad \overset{\times(1+1/p)}{\approx} \quad \#C.$$

Conversely, for each fixed constant $r \geq 2$ we define **WeakApproxCount**(r) with the requirement

$$\alpha \quad \overset{\times r}{\approx} \quad \#C.$$

However, it is well known that these problems are equivalent in the sense that

$$\textbf{StrongApproxCount} \leq_m \textbf{WeakApproxCount}(r)$$

for any r. Stockmeyer's trick for this is the following: Given an input (C, p) for **StrongApproxCount**, suppose we construct circuit C^k defined by $C^k(x^{(1)}, \ldots, x^{(k)}) = C(x^{(1)}) \wedge \cdots \wedge C(x^{(k)})$. Then $\#C^k = (\#C)^k$, and if $\alpha \overset{\times r}{\approx} (\#C)^k$ then $\alpha^{1/k} \overset{\times r^{1/k}}{\approx} \#C$. In this way the approximation factor can efficiently be reduced to $1 + 1/p$ by making $k = O(p)$ appropriately large.

Stockmeyer's trick applies equally well for **ApproxCountRatio**; i.e.,

$$\textbf{StrongApproxCountRatio} \leq_m \textbf{WeakApproxCountRatio}(r).$$

On the other hand, it's not obvious how to apply it for the **ApproxCountDifference** problem.

3 The power of approximate counting

The main goal of this section is to prove our first main result, Theorem 1.1, characterizing $\mathsf{BPP}_{\mathsf{path}}$ as the class of problems efficiently solvable with nonadaptive access to an **ApproxCount** oracle. We begin in Subsection 3.1 by recalling the definition of $\mathsf{BPP}_{\mathsf{path}}$ and the essentially equivalent notions of postselection, restarting, and the "path operator". Then in Subsection 3.2 we prove Theorem 1.1. Finally, in Subsection 3.3 we prove the analogue of Theorem 1.1 when BPP replaced by PP. This will be useful for our second main result, on the weakness of CTC qubits.

3.1 Prior definitions of $\mathsf{BPP}_{\mathsf{path}}$, postselection and restarting

Let M be a randomized (coin-flipping) Turing machine, which we take to equivalently mean a nondeterministic machine with branching factor 2 at each time step. For a given input x, let's write $\#_{\text{acc}}M(x)$ for the number of accepting computation paths, $\#_{\text{rej}}M(x)$ for the number of rejecting paths, and $\#_{\text{path}}M(x)$ for the total number of computation paths. Since we do *not* assume that each computation path has the length, it is not necessarily the case that $\mathbf{Pr}[M(x) \text{ accepts}] = (\#_{\text{acc}}M(x))/(\#_{\text{path}}M(x))$. The "path" version [HHT97] of a standard randomized complexity class is the (larger) class you get if you nevertheless use $(\#_{\text{acc}}M(x))/(\#_{\text{path}}M(x))$ in place of $\mathbf{Pr}[M(x) \text{ accepts}]$ in the class's definition. I.e.:

Definition 3.1. $\mathsf{BPP}_{\mathsf{path}}$ is the class of languages L for which there is a randomized polynomial-time Turing machine M such that

$$x \in L \quad \Rightarrow \quad \frac{\#_{\text{acc}}M(x)}{\#_{\text{path}}M(x)} \geq \frac{2}{3},$$
$$x \notin L \quad \Rightarrow \quad \frac{\#_{\text{acc}}M(x)}{\#_{\text{path}}M(x)} \leq \frac{1}{3}.$$

$\mathsf{PP}_{\mathsf{path}}$ is the class obtained if we replace the above bounds by $\geq \frac{1}{2}$ and $< \frac{1}{2}$. $\mathsf{RP}_{\mathsf{path}}$ is the class obtained if we replace the above bounds by $\geq \frac{1}{2}$ and $= 0$.

Essentially contemporaneous to Gill's definition [Gil74] of PP, Simon [Sim75] defined $\mathsf{PP}_{\mathsf{path}}$ and gave the (very easy) proof that $\mathsf{PP}_{\mathsf{path}} = \mathsf{PP}$. Han, Hemaspaandra, and Thierauf [HHT97] defined these classes more generally, and gave the (not hard) proof that $\mathsf{NP} = \mathsf{RP}_{\mathsf{path}}$. (The containment $\mathsf{RP}_{\mathsf{path}} \subseteq \mathsf{NP}$ is clear; we will explain the reverse containment shortly.) However, $\mathsf{BPP}_{\mathsf{path}}$ proved to be a "new" class, not interpretable in terms of other known classes. Han et al. showed $\mathsf{MA}, \mathsf{P}_{\|}^{\mathsf{NP}} \subseteq \mathsf{BPP}_{\mathsf{path}} \subseteq \mathsf{BPP}^{\mathsf{NP}}$, and it's clear from their proof of the last of these that in fact $\mathsf{BPP}_{\mathsf{path}} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$.

There is an alternative interpretation of these classes using Aaronson's notion of postselection. We recall the definition of $\mathsf{PostBPP}$ (with the definitions of $\mathsf{PostBQP}$, PostPP, etc. being analogous). A $\mathsf{PostBPP}$ machine M is a BPP machine with an additional ending state called "fail", in addition to the usual "accept" and "reject" states. The computation of $M(x)$ is then always *conditioned on* $M(x)$ not ending in a "fail" state. (Note: We say the computation $M(x)$ is "invalid" if it *always* ends in a "fail" state. An additional requirement on M is that $M(x)$ is never invalid.) It is not hard to show (see [Aar04b]) that $\mathsf{PostBPP} = \mathsf{BPP}_{\mathsf{path}}$, and similarly for PP and RP.

In this work we in fact prefer an equivalent perspective on postselection, called *restarting*, studied in [YS13]. The restarting model is the same as the postselection model, except that we think of the machine M as "restarting" its entire computation whenever it ends with a "fail" state. (The running time of M is treated as the maximum running time of a single start-to-(end/restart) pass, and not as the overall cumulative runtime.) It's not hard to see that this model is equivalent to postselection; nevertheless, we will henceforth write $\mathsf{RestartingBPP}$ in place of $\mathsf{PostBPP}$ (though both are equal to $\mathsf{BPP}_{\mathsf{path}}$). We prefer the restarting interpretation, as algorithm design seems to be a bit clearer in this model. For example, it is fairly clear that the following is a correct $\mathsf{RestartingRP}$ algorithm for **SAT** (thus establishing $\mathsf{RestartingRP} = \mathsf{NP}$): "Given formula ϕ on n variables, guess an assignment x; if x satisfies ϕ then accept; otherwise, restart with probability $1 - 2^{-n^2}$ and reject with probability 2^{-n^2}."

Finally, in Appendix A we describe Proposition A.1, a simple trick (appearing in [SY12]) which essentially lets one assume that $\mathsf{RestartingBPP}$ machines are never invalid.

3.2 Proof of our characterization of $\mathsf{BPP}_{\mathsf{path}}$

In this subsection we prove Theorem 1.1. In light of Remark 2.3, it is equivalent to prove the following:

Theorem 3.2. $\mathsf{BPP}_{\mathsf{path}} = \mathsf{P}^{\mathbf{ApproxCountRatio}[1]} = \mathsf{P}_{\|}^{\mathbf{ApproxCount}}$.

Remark 3.3. By Remark 2.4, the theorem also holds with either "strong" or "weak" versions of approximate counting.

Proof. Indeed, we will observe that

$$\mathsf{RestartingBPP} = \mathsf{BPP}_{\mathsf{path}} \subseteq \mathsf{P}^{\mathbf{ApproxCountRatio}[1]} \subseteq \mathsf{P}_{\|}^{\mathbf{ApproxCount}} \subseteq \mathsf{P}_{\|}^{\mathbf{ApproxProbDecision}} \subseteq \mathsf{RestartingBPP},$$

where **ApproxProbDecision** is a certain computational task defined below. Thus all the above classes are equal.

The first inclusion, $\mathsf{BPP}_{\mathsf{path}} \subseteq \mathsf{P}^{\mathbf{ApproxCountRatio}[1]}$, is straightforward from the original definition of $\mathsf{BPP}_{\mathsf{path}}$ (cf. [HHT97]'s proof that $\mathsf{BPP}_{\mathsf{path}} \subseteq \mathsf{BPP}^{\mathsf{NP}}$); it's also essentially proven in [SU06]. To be precise, suppose L is decided by a $\mathsf{BPP}_{\mathsf{path}}$ machine M; then by the parsimonious Cook–Levin Theorem, given x we can in polynomial time construct circuits C, D such that that $\#C$ equals the number of accepting paths of $M(x)$ and $\#D$ equals the total number of computations paths of $M(x)$. Then one call to **ApproxCountRatio** oracle (more precisely, one call to a **StrongApproxCountRatio** with precision parameter $p = 10$) lets us 1.1-approximate $\#C/\#D$, which is sufficient to decide whether it is at least $\frac{2}{3}$ or at most $\frac{1}{3}$.

The second inclusion is immediate from Remark 2.3, and in fact only two calls to the **ApproxCount** oracle are needed.

Next we come to the third inclusion, with **ApproxProbDecision** referring to the following computational task: Given circuits C, D, output "yes" if $\mathbf{Pr}[C] < \frac{1}{2} \cdot \mathbf{Pr}[D]$, output "no" if $\mathbf{Pr}[C] > 2 \cdot \mathbf{Pr}[D]$, and output either answer if $\mathbf{Pr}[C] \approx \mathbf{Pr}[D]$. Here $\mathbf{Pr}[C]$ denotes the *fraction* of input strings that C accepts, and similarly for $\mathbf{Pr}[D]$. To prove the third inclusion it's clearly sufficient to show that $\mathbf{ApproxCount} \in \mathsf{P}_{\|}^{\mathbf{ApproxProbDecision}}$; by Stockmeyer's trick (Remark 2.4) it's enough to show that given C, we can 4-approximate $\#C$ in $\mathsf{P}_{\|}^{\mathbf{ApproxProbDecision}}$. Of course, it suffices to 4-approximate $\mathbf{Pr}[C]$. This can be done with an essentially straightforward "nonadaptive approximate binary search". Letting n denote the number of input bits to C (which is at most its encoding length), we nonadaptively query the **ApproxProbDecision** oracle on (C, D_0), (C, D_1), $\ldots$, (C, D_{n+2}), where D_j denotes some simple circuit with

$$\mathbf{Pr}[D_j] = \frac{2^j}{2^{n+2}}.$$

Note that if $\mathbf{Pr}[C] \neq 0$ then $\mathbf{Pr}[C] \geq 2^{-n}$, in which case the **ApproxProbDecision** oracle must answer "no" on query (C, D_0). On the other hand, if $\mathbf{Pr}[C] = 0$ then the oracle must answer "yes" to all queries. Thus if all oracle answers are "yes", we may correctly output $\mathbf{Pr}[C] = 0$. Otherwise, we output $2^{j^*}/2^{n+2}$ as our approximation for $\mathbf{Pr}[C]$, where j^* is the least index for which the oracle responded "yes". (If the oracle never responds with "yes", we output 1.) It is easy to see that this output is necessarily a 4-approximation to $\mathbf{Pr}[C]$, as desired.

We now come to the last inclusion, $\mathsf{P}_{\|}^{\mathbf{ApproxProbDecision}} \subseteq \mathsf{RestartingBPP}$. The main observation here will be that

ApproxProbDecision can be solved in $\mathsf{RestartingBPP}$ with success probability at least 0.6. (1)

Once we show this, the conclusion that $\mathsf{P}_{\|}^{\mathbf{ApproxProbDecision}} \subseteq \mathsf{RestartingBPP}$ follows essentially from the fact [HHT97, Theorem 3.2] that $\mathsf{P}_{\|}^{\mathsf{BPP}_{\mathsf{path}}} = \mathsf{BPP}_{\mathsf{path}}$ (similarly, the fact [Aar05b, Proposition 3] that $\mathsf{P}_{\|}^{\mathsf{PostBQP}} = \mathsf{PostBQP}$). To give the details, suppose $L \in \mathsf{P}_{\|}^{\mathbf{ApproxProbDecision}}$, with $M^?$ being a nonadaptive oracle machine that decides L when given access to any correct oracle-implementation of **ApproxProbDecision**. On input x of length n, $M^?(x)$ computes for $\mathrm{poly}(n)$ time and deterministically generates some instances $(C_1, D_1), \ldots, (C_p, D_p)$ for **ApproxProbDecision**. Assuming it gets back p correct answers from the oracle, it computes deterministically for some more $\mathrm{poly}(n)$ time and then correctly either accepts or rejects x. Now assuming (1), and recalling that RestartingBPP has efficient success-probability amplification (just like BPP; see [HHT97, Theorem 3.1]), there is some RestartingBPP machine R that, when given any (C_j, D_j), eventually gives a correct answer except with probability at most $\frac{1}{3p}$. We now claim that the the following RestartingBPP machine S correctly decides L with probability at least $\frac{2}{3}$ (and hence that $L \in \mathsf{RestartingBPP}$ as needed): On input x, machine S simulates $M^?(x)$, then runs R sequentially on each of the instances $(C_1, D_1), \ldots, (C_p, D_p)$, obtains answers $z_1, \ldots, z_p$, and then continues to simulate $M^?(x)$ with these answers. It suffices to show that all answers z_j are simultaneously correct except with probability at most $1/3$. This follows because the p runs of R are independent; hence when we condition on all p runs not restarting, this is equivalent to separately conditioning each run $R(C_j, D_j)$ to not restart. Thus each run $R(C_j, D_j)$ returns a correct answer z_j except with probability at most $\frac{1}{3p}$, so indeed all answers are simultaneously correct except with probability at most $1/3$, by the union bound.

(At this point we would like to make the subtle point that this argument crucially uses the fact that $M^?$ is a deterministic machine. If $M^?$ were randomized, it would produce probability distributions on oracle query-sequences, and the probability that S restarts is likely to be different for different query-sequences. Hence when S finally halts, the conditional probability with which is processed each query-sequence is likely to be different from the probability with which $M^?$ generates it, likely causing the simulation to fail. This subtlety affects [Aar05b], as noted in [Aar14].)

Finally, it remains to show (1). Given an **ApproxProbDecision** instance, the following RestartingBPP algorithm "essentially" works (modulo a bug that will be fixed using Proposition A.1): Simulate $C(x)$ and $D(y)$, where x and y are randomly chosen inputs. If $C(x)$ rejects and $D(y)$ accepts, output "yes"; if $C(x)$ accepts and $D(y)$ rejects, output "no"; otherwise, if they give the same answer, restart. This algorithm ultimately outputs "yes" with probability

$$\frac{(1 - \mathbf{Pr}[C])\,\mathbf{Pr}[D]}{(1 - \mathbf{Pr}[C])\,\mathbf{Pr}[D] + \mathbf{Pr}[C](1 - \mathbf{Pr}[D])} \tag{2}$$

— except in the "edge cases" that $\mathbf{Pr}[C] = \mathbf{Pr}[D] \in \{0, 1\}$, in which it invalidly always restarts. Notice that in these edge cases $\mathbf{Pr}[C] \approx \mathbf{Pr}[D]$, so that it doesn't matter for **ApproxProbDecision** what answer the RestartingBPP algorithm returns. Thus we can safely apply Proposition A.1 to our RestartingBPP algorithm, which has a negligible effect on (2) in the "non-edge cases". To analyze these cases, suppose first that $\mathbf{Pr}[C] < \frac{1}{2} \cdot \mathbf{Pr}[D]$. Then since (2) is a decreasing function of $\mathbf{Pr}[C]$, we would have

$$(2) > \frac{(1 - \frac{1}{2}\,\mathbf{Pr}[D])\,\mathbf{Pr}[D]}{(1 - \frac{1}{2}\,\mathbf{Pr}[D])\,\mathbf{Pr}[D] + \frac{1}{2}\,\mathbf{Pr}[D](1 - \mathbf{Pr}[D])} = \frac{2 - \mathbf{Pr}[D]}{3 - 2\,\mathbf{Pr}[D]} \geq \frac{2}{3}.$$

Accounting for the use of Proposition A.1, we have correctness probability at least 0.6 in the case that $\mathbf{Pr}[C] < \frac{1}{2} \cdot \mathbf{Pr}[D]$. The case of $\mathbf{Pr}[C] > \frac{1}{2} \cdot \mathbf{Pr}[D]$ is entirely symmetrical. This completes the proof. □

3.3 The analogous theorem for PP

We will also state and prove the PP-analogue of Theorem 3.2. This result should not really be considered novel; there is an enormous literature on counting complexity and PP, and the proof of the result is more or less straightforward given this literature. We remark that we're not just recording this result for completeness; we'll actually need it for our proof Theorem 1.2 in Section 4, concerning the weakness of CTC qubits.

Theorem 3.4. $\mathsf{PP} = \mathsf{P}^{\mathbf{ApproxCountDifference}[1]} = \mathsf{P}_{\|,k}^{\mathbf{ApproxCountDifference}}$ *for any constant* k.

Proof. Similar to the previous proof, we'll observe that

$$\mathsf{P}_{\|,k}^{\mathsf{PP}} \subseteq \mathsf{PP} \subseteq \mathsf{P}^{\mathbf{ApproxCountDifference}[1]} \subseteq \mathsf{P}_{\|,k}^{\mathbf{ApproxCountDifference}} \subseteq \mathsf{P}_{\|,k}^{\mathsf{PP}}.$$

The first inclusion is the closure of PP under constant-round polynomial-time truth-table reductions, due to Fortnow and Reingold [FR96] (building on Beigel, Reingold, and Spielman [BRS95]). For the second inclusion, suppose $L \in \mathsf{PP}$ is decided with by randomized machine M (with probability exceeding $1/2$ on each input); we may assume $M(x)$ always makes some fixed polynomial $q(|x|)$ number of coin flips. By parsimonious Cook–Levin, given x we can efficiently build a circuit C such that $\#C$ is the number of accepting coin-flip sequences for $M(x)$. Now it only remains to determine $\mathrm{sgn}(\#C - \#D)$, where D is some easily constructed circuit with exactly $2^{q(|x|)+1}$ accepting inputs; this can be done with a single call to **ApproxCountDifference**. The third inclusion is trivial. It remains to show the fourth inclusion; for this, it suffices to show that $\mathbf{ApproxCountDifference} \in \mathsf{P}_{\|}^{\mathsf{PP}}$. Again, this can be done by a "nonadaptive approximate binary search". Given as input (C, D) (with total encoding length n), it suffices to correctly, nonadaptively determine the answers to the following $2n + 4$ questions:

$$\#C - \#D \overset{?}{\geq} 0, \quad \#C - \#D \overset{?}{\leq} 0, \quad \#C - \#D \overset{?}{\geq} 2^j, \quad \#C - \#D \overset{?}{\geq} -2^j, \quad \forall\, 0 \leq j \leq n.$$

But these all reduce to questions of the form $\#C' \overset{?}{\geq} \#D'$ for easily-constructed circuits C', D', and this is a PP-complete problem. □

Remark 3.5. Although it is not obvious how to apply Stockmeyer's trick to the **ApproxCountDifference** problem, nevertheless it's not hard to see that Theorem 3.4 continues to hold either with a $\mathbf{WeakApproxCountDiff}(r)$ oracle (for any r) or a **StrongApproxCountDiff** oracle. On one hand, our argument for $\mathsf{PP} \subseteq \mathsf{P}^{\mathbf{ApproxCountDifference}[1]}$ only needed to determine $\mathrm{sgn}(\#C - \#D)$, which can be done with a single call to $\mathbf{WeakApproxCountDiff}(r)$ for any r. On the other hand, to put **StrongApproxCountDiff** in $\mathsf{P}_{\|}^{\mathsf{PP}}$ we just need to do a more refined nonadaptive approximation. Specifically, suppose input (C, D, p) has encoding length n, so C and D have at most n inputs, and $p \leq n$. Since $\#C - \#D \in \mathbb{Z} \cap [-2^n, 2^n]$, to $(1 + 1/p)$-approximate $\#C - \#D$ it suffices to correctly, nonadaptively determine answers to

$$\#C - \#D \gtrless \pm k_j$$

for all integers k_j in some nondecreasing sequence satisfying

$$k_j = j \text{ for all } 0 \leq j \leq p, \qquad k_{j+1} \leq (1 + 1/p) k_j \text{ for all } j \geq p, \qquad k_N = 2^n.$$

It is straightforward to construct an entire such sequence (along with circuits C'_j, D'_j having $\#C'_j = \#C + k_j$ and $\#D'_j = \#D + k_j$) in deterministic $\mathrm{poly}(n)$ time.

4 The weakness of CTC qubits

We begin this section by recalling the (Deutschian) model of computation in the presence of closed timelike curves (CTCs). For more information, see e.g. [Deu91, Bac04, Aar05a, AW09, SY12, OS14]. Briefly and informally, $\mathsf{BQP}_{\mathsf{QCTC}[w]}$ is the class of languages decided with high probability by a quantum algorithm that can set up a w-qubit quantum channel and then freely get one sample from (one of) its stationary mixed state(s).

Definition 4.1. We describe here the $\mathsf{BQP}_{\mathsf{QCTC}[w]}$ model of computation, where $w = w(n)$ is the "width" parameter. (If any $w = \mathrm{poly}(n)$ is allowed we write simply $\mathsf{BQP}_{\mathsf{QCTC}[\mathrm{poly}]}$; if any $w = O(\log n)$ is allowed we write simply $\mathsf{BQP}_{\mathsf{QCTC}[\log]}$.) Let S be a polynomial-time deterministic Turing machine that on input $x \in \Sigma^n$ outputs the description of a quantum circuit C composed of Hadamard, Toffoli, and phase-shift gates.[2] The circuit C should have a $w(n)$-qubit "CTC register", and some additional qubits designated the "CR (causality-respecting) register". The CR-qubits begin with ancillary gates initialized to $|0\rangle$ and end with measurement gates. In this way, C defines some w-qubit quantum *channel* $\mathcal{C}$. The CTC qubits are assumed to be initialized to an arbitrary stationary mixed state ρ for $\mathcal{C}$. (Every quantum channel has at least one stationary mixed state, by Brouwer's fixed point theorem; see, e.g., [Deu91, AW09, Wol12].) We designate the last qubit of the CR register to be the "output" bit. Finally, we say that a language $L \in \Sigma^*$ is in the class $\mathsf{BQP}_{\mathsf{QCTC}[w]}$ if there is an S as described above such that for all inputs $x \in \Sigma^*$, the output of the resulting C agrees with $1[x \in L]$ with probability at least $2/3$ (regardless of what stationary ρ is chosen for the CTC register).

Definition 4.2. The above definition can be naturally restricted in several ways. If the CTC register is restricted to carry *bits* rather than qubits — thereby making $\mathcal{C}$ into simply a 2^w-state Markov chain — then we call the resulting class $\mathsf{BQP}_{\mathsf{CTC}[w]}$. If we further restrict the circuit C to be simply a randomized circuit (with AND, OR, NOT, and probability-$\frac{1}{2}$ coin flip gates), then the resulting class is called $\mathsf{BPP}_{\mathsf{CTC}[w]}$. Finally, if even the coin-flip gates are disallowed (making the Markov chain $\mathcal{C}$ deterministic) and the computation's decision must be correct with probability 1, the resulting class is called $\mathsf{P}_{\mathsf{CTC}[w]}$.

Recall from the second part of Section 1.1 that prior to our work, the following results were known:

$$\mathsf{BQP}_{\mathsf{QCTC}[\mathrm{poly}]} = \mathsf{P}_{\mathsf{CTC}[\mathrm{poly}]} = \mathsf{PSPACE}, \qquad \mathsf{BQP}_{\mathsf{CTC}[\log]} = \mathsf{BQP}_{\mathsf{CTC}[1]} = \mathsf{PP}.$$

In Subsection 4.2 below we will prove the paper's second main result, Theorem 1.2, that in fact even $\mathsf{BQP}_{\mathsf{QCTC}[\log]}$ is equal to $\mathsf{BQP}_{\mathsf{CTC}[1]} = \mathsf{PP}$. Before that, we'll need to establish a couple of technical results. One such result, concerning converting an approximate density matrix into a proper one, is rather mundane; it's completely deferred to Appendix B. The other result, concerning closure properties within counting classes, is somewhat more important and nontrivial. We discuss it next, but also defer its proof to an appendix.

4.1 Closure properties within counting classes

Our proof of $\mathsf{BQP}_{\mathsf{QCTC}[\log]} = \mathsf{PP}$ significantly relies on a certain counting-complexity result, which we state here informally:

[2] This is a standard universal gate set for traditional quantum computation, and is akin to assuming that randomized circuits have access just to probability-$\frac{1}{2}$ coin flips. See the last paragraph of [AW09, Section 6] for related discussion.

If the entries of a matrix are #P-definable univariate polynomials over the complex integers (with the fraction $\frac{1}{2}$ adjoined) then so too is the determinant of this matrix.

On one hand, several similar statements are known in the literature: e.g. [AAM03] that the characteristic polynomial of GapL-definable matrices is GapL-definable; or, that the product of polynomially many GapP-definable matrices with entries from $\mathbb{Z}$ or $\mathbb{Z}+\imath\mathbb{Z}$ is GapP-definable [FR99, Wat08]. Indeed, the last of these is the essence of the standard proof that BQP $\subseteq$ PP. In some sense, our desired statement is somewhat "routine" for those well-versed in counting complexity. Nevertheless, the exact statement we want is a bit complex, and we were unable to find in the literature even the statement that the determinant of a GapP-definable matrix of integers is GapP-definable. Hence we will give a careful formal statement, as well as a somewhat thorough proof sketch, in Appendix C.

4.2 Proof of Theorem 1.2

We are now able to give the proof that $O(\log n)$ time-traveling qubits can be compressed to a single time-traveling bit.

Proof of Theorem 1.2. We need to show that $\mathsf{BQP}_{\mathsf{QCTC}[w]} \subseteq \mathsf{PP}$, where $w = w(n) = O(\log n)$. There is no harm in assuming that in fact $w = \Theta(\log n)$; we will do so to simplify the exposition. Now suppose **Lang** $\subseteq \Sigma^*$ is in $\mathsf{BQP}_{\mathsf{QCTC}[\log]}$, with associated Turing machine S; by Theorem 3.4 and Remark 3.5, it will suffice to describe a $\mathsf{P}_{\|,2}^{\mathbf{StrongApproxCountDiff}}$ algorithm deciding **Lang**.

Our proof will follow the Aaronson–Watrous proof [AW09] of $\mathsf{BQP}_{\mathsf{QCTC}[\mathrm{poly}]} \subseteq \mathsf{PSPACE}$ somewhat closely. On input x of length n, suppose we simulate $S(x)$, producing circuit C_x. On one hand, we may view C_x as a *unitary quantum circuit* with w CTC qubits and $\mathrm{poly}(n)$ CR qubits, with the property that when the CR qubits are fixed to $|0\rangle$ at the input and are measured at the output, the result is a w-qubit quantum channel (or (super)operator) $\mathcal{C}_x$. On the other hand, we may view C_x as a *general quantum circuit* (see [Wat08]) having only the w-qubit CTC register, with ancillary gates at the CR inputs and erasure gates at the CR outputs. (We remark that in this view, the size of the circuit C_x is actually exponential in its number of inputs w, but is nevertheless $\mathrm{poly}(n)$.)

Let $M_x \in \mathbb{C}^{2^{2w}\times 2^{2w}}$ be the *natural matrix representation* for C_x (in the general quantum circuit viewpoint). Thus for all $2^w \times 2^w$ density matrices ρ we have $\mathrm{vec}(\mathcal{C}_x(\rho)) = M_x\mathrm{vec}(\rho)$, where as usual vec is the linear operator stacking matrices into vectors (i.e., $\mathrm{vec}(|y\rangle\langle z|) = |y\rangle|z\rangle$). As shown in [Wat08, Section IV.5] (essentially following the BQP $\subseteq$ PP proof from [FR99]), the $2^{4w} = \mathrm{poly}(n)$ many entries of M_x — each of which is a number in $(\mathbb{Z}+\imath\mathbb{Z})[\frac{1}{2}]$ — are "polynomial-time countable" (PC) functions of x (see Appendix C for our definition of "PC"). In the notation of Appendix C, the function $f : \Sigma^* \to (\mathbb{Z}+\imath\mathbb{Z})[\frac{1}{2}]^{\mathbb{W}\times\mathbb{W}}$ defined by

$$f(x) = [M_x]_{<2^{2w},2^{2w}}$$

is PC. As a minor technical note, in [Wat08] the entries of M_x are taken to be complex integers, with the powers of $\frac{1}{2}$ arising from Hadamard gates "implicitly remembered". We will instead handle the powers of $\frac{1}{2}$ explicitly, letting $B = b(n) = \mathrm{poly}(n)$ be a bound on the maximum such power of $\frac{1}{2}$ (say, $b(n)$ is the running time of S, which bounds the number of Hadamard gates in all circuits C_x).

Let Λ_x denote the "fixed point projection" operator associated to $\mathcal{C}_x$ as described in [AW09] (equivalently, the "Cesaro means" operator, denoted $(\mathcal{C}_x)_\infty$, described in [Wol12, Chapter 6]). Specifically,

$$\Lambda_x = \lim_{z\searrow 0^+} \Lambda_x(z), \qquad \text{where} \quad \Lambda_x(z) := z\sum_{t=0}^{\infty}(1-z)^t\mathcal{C}_x^t,$$

and Λ_x is an onto map from the set of all $2^w \times 2^w$ density matrices to the set of all density matrices that are fixed points for $\mathcal{C}_x$. As shown in [AW09], the natural matrix representation of Λ_x is

$$R_x = \lim_{z \searrow 0^+} R_x(z), \qquad \text{where} \quad R_x(z)[j,k] = \frac{(-1)^{j+k} z \cdot \det((I-(1-z)M_x)^{\setminus k,j})}{\det(I-(1-z)M_x)}, \tag{3}$$

and each entry of R_x has absolute value at most 1. (In the above, the notation $N^{\setminus k,j}$ indicates that the kth row and jth column are deleted from matrix N.) We now treat z as an indeterminate, rather than a real in $[0,1)$. The matrix $I-(1-z)M_x$ is easily seen to be a PC function of x, since M_x is. Hence Theorem C.8 (with $L = D = 2^{2w} = \mathrm{poly}(n)$) tells us that the denominator of (3) is a PC function of x (in the sense described in that theorem). Similarly, the numerator of (3) is easily seen to be a PC function of x, j, and k, in the sense of Theorem C.8.

(As additional technical notes, although the numerator of (3) involves an $(L-1)\times(L-1)$ matrix, we define its z-degree bound to be L in light of the extra factor of z in the numerator. Also, we may arrange for the resulting denominator in Theorem C.8 to be 2^{LB} rather than $2^{(L-1)B}$, as it is when we compute. Then, since we only care about the ratio of the numerator and denominator in (3), we can actually completely ignore these canceling powers of $\frac{1}{2}$.)

Thus if we write

$$R_x(z) = \Big(R_x[j,k]\Big)_{1 \le j,k \le L}, \qquad \text{where} \quad R_x[j,k] = \frac{\sum_{\ell=0}^{L}(a_\ell^{j,k} + \imath b_\ell^{j,k})z^\ell}{\sum_{\ell=0}^{L}(c_\ell^{j,k} + \imath d_\ell^{j,k})z^\ell}, \tag{4}$$

we know that the $4L^2(L+1)$ integers $(a_\ell^{j,k}), (b_\ell^{j,k}), (c_\ell^{j,k}), (d_\ell^{j,k})$ are GapP functions of x (more precisely, there is a GapP function taking as input x and a unary index from from $[4L^2(L+1)]$, and outputting the associated a_ℓ, b_ℓ, c_ℓ, or d_ℓ).

We now finally return to the $\mathsf{P}_{\|,2}^{\textbf{StrongApproxCountDiff}}$ for deciding **Lang**. Using the above deductions and the parsimonious Cook–Levin theorem, on input $x \in \Sigma^n$ we can construct in deterministic polynomial time pairs of circuits $\overline{A}_\ell^{j,k}, \underline{A}_\ell^{j,k}$ for $1 \le j,k \le L = 2^{2w(n)}$ and $0 \le \ell \le L$ with the property that $\#\overline{A}_\ell^{j,k} - \#\underline{A}_\ell^{j,k} = a_\ell^{j,k}$ for a_ℓ as in (4); and, similarly circuits for $b_\ell^{j,k}, c_\ell^{j,k}, d_\ell^{j,k}$. We now use the first round of our queries to the **StrongApproxCountDiff** oracle on all these pairs of circuits, with precision parameter $p = \mathrm{poly}(n)$ to be specified later. With the resulting information we are able to determine, for each fixed j,k, the least ℓ^* such that $c_{\ell^*}^{j,k} + \imath d_{\ell^*}^{j,k} \neq 0$; it then follows that

$$R_x[j,k] = \lim_{z \searrow 0^+} R_x(z)[j,k] = \frac{a_{\ell^*}^{j,k} + \imath b_{\ell^*}^{j,k}}{c_{\ell^*}^{j,k} + \imath d_{\ell^*}^{j,k}},$$

and we furthermore have $(1+1/p)$-approximations each of the quadruples of integers appearing on the right-hand side. Recall that R_x is the natural matrix representation of Λ_x, and that $\Lambda_x(\sigma)$ is a fixed point for $\mathcal{C}_x$ for any density matrix σ. Selecting, say, $\sigma = |0^w\rangle\langle 0^w|$, we have $\mathrm{vec}(\rho) = (1,0,0,\ldots,0)$ and hence that the first column of R_x is $\mathrm{vec}(\rho)$ for some ρ a fixed point of $\mathcal{C}_x$. Thus our algorithm can obtain an approximation $\rho' \in \mathbb{C}^{2^w \times 2^w}$ of a fixed point ρ of $\mathcal{C}_x$ with the following guarantee: for each $1 \le j,k \le 2^w$ there are integers a,b,c,d and $(1+1/p)$-approximations of them a',b',c',d' such that $\rho[j,k] = \frac{a+\imath b}{c+\imath d}$ and $\rho'[j,k] = \frac{a'+\imath b'}{c'+\imath d'}$. Applying Lemma B.1 from Appendix B (with $p = \mathrm{poly}(n)$ chosen retrospectively to to be a sufficiently large compared to 2^w), our algorithm further obtains a complex rational density matrix $\widetilde{\rho} \in \mathbb{C}^{2^w \times 2^w}$ with $\delta(\rho,\widetilde{\rho}) \le .01$.

By definition of **Lang** $\in \mathsf{BQP}_{\mathsf{QCTC}[w]}$, if the CTC bits of C_x were set according to ρ, the circuit would correctly decide whether $x \in$ **Lang** with probability at least $2/3$. By the properties of trace distance, if we use $\widetilde{\rho}$ instead, each x will still be decided correctly with probability at least

$2/3 - .01 \geq 5/8$. Therefore, to complete the proof of $\mathbf{Lang} \in \mathsf{P}_{\|,2}^{\mathbf{StrongApproxCountDiff}}$ it suffices to show that we can, say, 5/4-approximate the the acceptance probability of $C_x(\widetilde{\rho})$ with a single call to the **StrongApproxCountDiff** oracle. That this is possible essentially follows from the proof of $\mathsf{BQP} \subseteq \mathsf{PP}$. More precisely, from [Wat08, Section IV.5] (and parsimonious Cook–Levin) we could produce in $\mathrm{poly}(n)$ time produce circuits G and H such that

$$\mathbf{Pr}[\mathcal{C}_x(|0^w\rangle \langle 0^w|) \text{ accepts}] = \frac{\#G - \#H}{2^q},$$

where $q = \mathrm{poly}(n)$ is a size bound for C_x. This is not quite what we need; we want to input $\widetilde{\rho}$ to $\mathcal{C}_x$, not $|0^w\rangle \langle 0^w|$. But it's straightforward (cf. [FGHP99]) to augment [Wat08, Section IV.5] to allow for this, since our algorithm maintains the $\mathrm{poly}(n)$-size matrix $\widetilde{\rho}$ of complex rationals explicitly. Thus we can can construct the appropriate circuits G, H for $\mathcal{C}_x(\widetilde{\rho})$, and complete the decision for $x \in \mathbf{Lang}$ with one more call to the **StrongApproxCountDiff** oracle on G and H with precision parameter 1/4. □

Acknowledgments

The authors would like to thank Scott Aaronson for helpful answers to quite a number of questions, as well as Chris Umans for his expertise on $\mathsf{BPP}_{\mathsf{path}}$, and Eric Allender and Lance Fortnow for their expertise on GapP. The first author would also like to thank the Boğaziçi University Computer Engineering Department for its hospitality.

References

[AA11] Scott Aaronson and Andris Ambainis. The need for structure in quantum speedups. In *Proceedings of the 2nd Annual Innovations in Theoretical Computer Science conference*, pages 338–352, 2011. 1

[AAM03] Eric Allender, Vikraman Arvind, and Meena Mahajan. Arithmetic complexity, Kleene closure, and formal power series. *Theory of Computing Systems*, 36(4):303–328, 2003. 4.1

[Aar04a] Scott Aaronson. Is quantum mechanics an island in Theoryspace? Technical Report quant-ph/0401062, arXiv, 2004. 1

[Aar04b] Scott Aaronson. *Limits on Efficient Computation in the Physical World*. PhD thesis, University of California, Berkeley, 2004. 1, 3.1

[Aar05a] Scott Aaronson. NP-complete problems and physical reality. *ACM SIGACT News*, 36(1):30–52, 2005. 4

[Aar05b] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A*, 461(2063):3473–3482, 2005. 1, 3.2

[Aar14] Scott Aaronson. PostBQP postscripts: A confession of mathematical errors, 2014. http://www.scottaaronson.com/blog/?p=2072. 3.2

[AFF+01] 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. 1

[AW09] 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. 1, 1.1, 4, 4.1, 2, 4.2

[Bac04] Dave Bacon. Quantum computational complexity in the presence of closed timelike curves. *Physical Review A*, 70(3):032309, 2004. 1, 1.1, 4

[BGM03] Elmar Böhler, Christian Glaßer, and Daniel Meister. Error-bounded probabilistic computations between MA and AM. In *Proceedings of the 28th Annual International Symposium on Mathematical Foundations of Computer Science*, pages 249–258, 2003. 1

[BJS10] Michael Bremner, Richard Jozsa, and Dan Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. In *Proceedings of the Royal Society A*, pages 459–472, 2010. 1

[BRS95] Richard Beigel, Nick Reingold, and Daniel Spielman. PP is closed under intersection. *Journal of Computer & System Sciences*, 50(2):191–202, 1995. 1, 1.1, 3.3

[Deu91] David Deutsch. Quantum mechanics near closed timelike lines. *Physical Review D*, 44(10):3197–3217, 1991. 1, 4, 4.1

[FGHP99] Stephen Fenner, Frederic Green, Steven Homer, and Randall Pruim. Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 455(1991):3953–3966, 1999. 4.2

[FR96] Lance Fortnow and Nick Reingold. PP is closed under truth-table reductions. *Information and Computation*, 124(1):1–6, 1996. 1.1, 3.3

[FR99] Lance Fortnow and John Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999. 4.1, 4.2

[Gil74] John Gill. Computational complexity of probabilistic Turing machines. In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing*, pages 91–95, 1974. 3.1

[Göd49] 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. 1

[GSS06] Carla Gomes, Ashish Sabharwal, and Bart Selman. Model counting: A new strategy for obtaining good bounds. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 54, 2006. 1.2

[HHT93] 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. 1

[HHT97] Yenjo Han, Lane Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, 26(1):59–78, 1997. 1, 3.1, 3.1, 3.2, 3.2

[OS14] Ryan O'Donnell and A. C. Cem Say. One time-traveling bit is as good as logarithmically many. In *Proceedings of the 35th Annual IARCS Conference on Foundations of Software Technology and Theoretical Computer Science*, 2014. 1, 1.1, 4

[PPT+10] Claudia Peschiera, Luca Pulina, Armando Tacchella, Uwe Bubeck, Oliver Kullmann, and Inês Lynce. The seventh QBF solvers evaluation (QBFEVAL'10). In *Theory and Applications of Satisfiability Testing—SAT 2010*, pages 237–250. Springer, 2010. 1.2

[SE14] Carsten Sinz and Uwe Egly, editors. *Theory and Applications of Satisfiability Testing—SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*. Springer, 2014. 1.2

[Sim75] Janos Simon. *On some central problems in computational complexity*. PhD thesis, Cornell University, 1975. TR 75-224. 1, 3.1

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 330–335, 1983. 1.1

[Sto83] Larry Stockmeyer. The complexity of approximate counting. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 118–126, 1983. 1.1

[Sto85] Larry Stockmeyer. On approximation algorithms for #p. *SIAM Journal on Computing*, 14(4):849–861, 1985. 1.1, 2.4

[SU06] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006. 1.1, 3.2

[SY12] A. C. Cem Say and Abuzer Yakaryılmaz. Computation with multiple CTCs of fixed length and width. *Natural Computing*, 11(4):579–594, 2012. 1, 1, 1.1, 3.1, 4

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991. 1.1

[Wat08] John Watrous. Quantum computational complexity. Technical Report 0804.3401, arXiv, 2008. 4.1, 4.2, 4.2

[Wol12] Michael Wolf. Quantum channels & operations: guided tour, 2012. http://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MichaelWolf/QChannelLecture.pdf. 4.1, 4.2

[YS13] Abuzer Yakaryılmaz and A. C. Cem Say. Proving the power of postselection. *Fundamenta Informaticae*, 123(1):107–134, 2013. 3.1

A A simple fix for invalid RestartingBPP machines

Proposition A.1. *Let M be a* RestartingBPP *machine which is possibly* invalid *on some inputs, meaning that it might restart with probability* 1*. There is a simple transformation to a* RestartingBPP *machine M' which is valid on all inputs and which has the following properties:*

- *If $M(x)$ is invalid, then $M'(x)$ rejects with probability* 1*.*
- *If $M(x)$ is valid, then*

$$\mathbf{Pr}[M'(x) \text{ accepts}] \overset{\times(1+2^{-|x|})}{\approx} \mathbf{Pr}[M(x) \text{ accepts}], \quad \mathbf{Pr}[M'(x) \text{ rejects}] \overset{\times(1+2^{-|x|})}{\approx} \mathbf{Pr}[M(x) \text{ rejects}].$$

Proof. Let t be a polynomial time-bound for M (i.e., $M(x)$ always accepts, rejects, or restarts within $t(|x|)$ steps). Now on input x with $|x| = n$ the machine M' acts as follows: First, it immediately rejects with probability $\epsilon := 2^{-t(n)-n-1}$; otherwise it simulates $M(x)$. $M'(x)$ is clearly always valid, and further if $M(x)$ is invalid (i.e., always restarts) then $M'(x)$ will always reject. Finally, suppose $M(x)$ is valid and it accepts with probability p, rejects with probability q, where $p + q > 0$. Then

$$\mathbf{Pr}[M(x) \text{ accepts}] = \frac{p}{p+q},$$

and

$$\mathbf{Pr}[M'(x) \text{ accepts}] = \frac{(1-\epsilon)p}{(1-\epsilon)p + ((1-\epsilon)q + \epsilon)} = \frac{p}{p+q+\epsilon/(1-\epsilon)} = \frac{p}{p+q} \cdot \frac{1}{1+\frac{\epsilon/(1-\epsilon)}{p+q}}.$$

But $p+q > 0$ implies $p+q \geq 2^{-t(n)}$, and hence $\frac{\epsilon/(1-\epsilon)}{p+q} \leq 2^{-n}$, as desired. The calculation regarding rejection probability is similar. □

B A technical lemma on approximate density matrices

Lemma B.1. *There is a deterministic* poly(m)*-time algorithm with the following guarantee. Suppose the algorithm is given as input a matrix $\rho' \in \mathbb{C}^{m \times m}$ (with m a power of 2), where entry $\rho'[j,k]$ is expressed as $(a'_{jk} + \imath b'_{jk})/(c'_{jk} + \imath d'_{jk})$, and where $a'_{jk}, b'_{jk}, c'_{jk}, d'_{jk}$ are integers of encoding-length* poly(m)*. Further assume that these integers $(1+\epsilon)$-approximate some integers $a_{jk}, b_{jk}, c_{jk}, d_{jk}$ (respectively) with the property that $\rho \in \mathbb{C}^{m \times m}$ defined by $\rho[j,k] = (a_{jk}+\imath b_{jk})/(c_{jk}+\imath d_{jk})$ is a density matrix. Then the algorithm outputs a density matrix $\widetilde{\rho} \in (\mathbb{Q}+\imath\mathbb{Q})^{m \times m}$ satisfying $\delta(\rho, \widetilde{\rho}) \leq \text{poly}(m) \cdot \epsilon$.*

Proof. We begin by showing that ρ' is in fact entrywise close to ρ. Fix any entry coordinates $1 \leq j, k \leq m$ (which we will drop from the subsequent notation for clarity). We have $|a+\imath b|^2 = a^2 + b^2$ and $|a' + \imath b'|^2 = (a')^2 + (b')^2$. Since a', b' are $(1+\epsilon)$-approximations of a, b, it follows that

$$|a' + \imath b'| \overset{\times(1+O(\epsilon))}{\approx} |a + \imath b|.$$

A similar statement holds for $|c + \imath d|$, and hence

$$\left|\frac{a' + \imath b'}{c' + \imath d'}\right| \overset{\times(1+O(\epsilon))}{\approx} \left|\frac{a + \imath b}{c + \imath d}\right|.$$

Next, since a'/b' is a $(1+2\epsilon)$-approximation of a/b, it is not hard to show (using elementary properties of arctan) that $|\operatorname{Arg}(a'+\imath b') - \operatorname{Arg}(a+\imath b)| \leq O(\epsilon)$ (or if $a+\imath b = 0$ then so too is $a'+\imath b'$). The same holds true for $c' + \imath d'$, and hence

$$\left|\operatorname{Arg}\left(\frac{a' + \imath b'}{c' + \imath d'}\right) - \operatorname{Arg}\left(\frac{a + \imath b}{c + \imath d}\right)\right| \leq O(\epsilon).$$

From the last two deductions — together with the fact that $|(a+\imath b)/(c+\imath d)| \leq 1$ since $(a+\imath b)/(c+\imath d)$ is an entry in a density matrix — it is not hard to obtain the entrywise error bound

$$\left|\frac{a' + \imath b'}{c' + \imath d'} - \frac{a + \imath b}{c + \imath d}\right| \leq O(\epsilon). \tag{5}$$

The same would be true if we replaced ρ' with $\frac{1}{2}(\rho' + (\rho')^\dagger)$, since $\rho = \rho^\dagger$; we will assume the algorithm in fact makes this replacement, thereby ensuring that ρ' is Hermitian. From the above entrywise estimate, we deduce the Hilbert–Schmidt norm bound

$$\|\rho - \rho'\|_{HS} \leq O(m\epsilon).$$

This also bounds the spectral norm $\|\rho - \rho'\|$. Since all of the eigenvalues of ρ are nonnegative, we conclude that all the eigenvalues of ρ' are at least $-O(m\epsilon)$. If we now replace ρ' with $\rho' + O(m\epsilon)I$ we get that ρ' is positive semidefinite. This also only changes the entrywise bound (5) by $O(m\epsilon)$. Since $\mathrm{tr}(\rho) = 1$, it follows that $\mathrm{tr}(\rho') \leq 1 + O(m^2\epsilon)$. If finally we normalize ρ' by its trace, it becomes a valid density matrix, with

$$|\rho[j,k] - \rho'[j,k]| \leq O(m^2\epsilon)$$

for all $1 \leq j, k \leq m$. And from this we can deduce the trace distance bound $\delta(\rho, \rho') \leq O(m^{3.5}\epsilon)$. □

C Closure properties within counting classes

In this section only, we write $[n] = \{0, 1, 2, \ldots, n-1\}$ rather than $\{1, 2, \ldots, n\}$, and we index strings, vectors, matrices, etc. starting from 0 rather than from 1.

Let $\Sigma = \{a, b\}$ and let $\overline{\Sigma} = \Sigma \cup \{,, +, -, \Re, \Im, |\}$. We write **Inputs** for the set of inputs to a computational problem, and identify it with binary strings from Σ^*. We typically call a function **Inputs** $\to T$ a *sequence of T's*. We identify the integers $\mathbb{Z}$ with the set of functions $\{+, -\} \to \mathbb{N}$ in the natural way; i.e., $z : \{+, -\} \to \mathbb{N}$ stands for the integer $z(+) - z(-)$. (Note that a function corresponds to a unique integer, but an integer has infinitely many representations as functions.) Similarly, complex integers $\mathbb{Z} + \imath\mathbb{Z}$ are identified with functions $\{\Re, \Im\} \to \mathbb{Z}$; i.e., functions $\{\Re, \Im\} \to (\{+, -\} \to \mathbb{N})$. As is customary, we also write this as $\{\Re, \Im\} \to \{+, -\} \to \mathbb{N}$ and further identify it with $(\{\Re, \Im\} \times \{+, -\}) \to \mathbb{N}$.

As an example, let's consider a "sequence of complex integers f : **Inputs** $\to (\mathbb{Z} + \imath\mathbb{Z})$". This is identified with a function $f : \Sigma^* \times \{\Re, \Im\} \times \{+, -\} \to \mathbb{N}$. The ω'th element in the sequence is

$$\Big(f(\omega, \Re, +) - f(\omega, \Re, -)\Big) + \imath\Big(f(\omega, \Im, +) - f(\omega, \Im, -)\Big).$$

The input to such an f is naturally represented by a string in $(\overline{\Sigma} \setminus \{|\})^*$.

Even though we already have the notation $\mathbb{N}$, for clarity we will also write $\mathbb{W} = \{0, 1, 2, 3, \ldots\}$ ("whole numbers") for indexing purposes. We identify this set with the unary strings from $\{|\}^*$, thinking of | as a "tally" symbol. We identify an infinite-length vector of natural numbers $v \in \mathbb{N}^{\mathbb{W}}$ with a function $\mathbb{W} \to \mathbb{N}$, and use the notation $[v]_{<m}$ to denote its truncation to length m. We would similarly identify a vector of, say, integers with a function $\mathbb{W} \to \mathbb{Z} = (\mathbb{W} \times \{+, -\}) \to \mathbb{N}$. We think of (infinite) matrices A as vectors of vectors, and also use the notation $[A]_{<m,m'}$ for their $m \times m'$ truncation. We use similar notation for higher-order tensors (i.e., "k-dimensional arrays", $k > 2$).

Finally, we will identify a univariate polynomial $Q \in \mathbb{N}[x]$ having natural-number coefficients with the vector of its coefficients. More generally, we make such an identification for any univariate power series Q over a semiring; $[Q]_{<m}$ will denote its truncation to degree $m - 1$. Even more generally, we identify a bivariate power series over a semiring with its matrix of coefficients, and similarly for power series over more than two variables.

Let us give one more example of all of these definitions. A sequence of matrices with entries from $\mathbb{Z}[x]$ would be identified with a function $f : \mathbf{Inputs} \times \mathbb{W} \times \mathbb{W} \times \mathbb{W} \times \{+,-\} \to \mathbb{N}$, where, e.g.,

$$f(\omega, |, ||, |||, +) - f(\omega, |, ||, |||, -)$$

would be the (integer) coefficient on x^3 in the $(1,2)$-entry of the ω'th matrix. Finally, for any such function f whose range is $\mathbb{N}$, we say it is *polynomial-time countable*, denoted PC, if there is a polynomial-time nondeterministic Turing machine F with $f(X)$ equal to the number of accepting computation paths of $F(X)$ for all well-formed input strings $X \in \overline{\Sigma}^*$. For example, $f : \mathbf{Inputs} \to \mathbb{N}$ is PC if and only if $f \in \#\mathsf{P}$, and $f : \mathbf{Inputs} \to \mathbb{Z}$ is PC if and only if $f \in \mathsf{GapP}$.

We now turn to closure properties. As warmups, let us first review some well-known basic closure properties of $\#\mathsf{P}$ and GapP:

Proposition C.1. *$\#\mathsf{P}$ is closed under summation-marginalization over sequences ("exponential summation"), and product-marginalization over vectors ("polynomial product"). That is:*

- *If $f : \mathbf{Inputs} \times \Sigma^* \to \mathbb{N}$ is PC, and p is a (length-bounding) polynomial, then the following $g : \mathbf{Inputs} \to \mathbb{N}$ is PC:*
$$g(\omega) = \sum_{\alpha \in \Sigma^{p(|\omega|)}} f(\omega, \alpha).$$
- *If $f : \mathbf{Inputs} \times \mathbb{W} \to \mathbb{N}$ is PC, and p is a polynomial, then the following $g : \mathbf{Inputs} \to \mathbb{N}$ is PC:*
$$g(\omega) = \prod_{j < p(|\omega|)} f(\omega, j).$$

Proof. For summation-marginalization over sequences, let F be the TM for f. Then the TM G for g acts as follows: on input ω, it computes $p(|\omega|)$, nondeterministically branches over all $\alpha \in \Sigma^{p(|\omega|)}$, and then simulates $F(\omega, \alpha)$.

For product-marginalization over vectors, let F be the TM for f. Then the TM G for g acts as follows: For $j = 0, 1, 2, \ldots, p(|\omega|) - 1$, it simulates $F(\omega, |^j)$, rejecting whenever F does. If all simulations accept, so does G.

(In both cases, we trust the reader to make the easy verification that the defined G gives the correct answer and is polynomial-time.) □

Proposition C.2. *(Product-marginalization for GapP.) If $f : \mathbf{Inputs} \times \mathbb{W} \to \mathbb{Z}$ is PC, and p is a polynomial, then the following $g : \mathbf{Inputs} \to \mathbb{Z}$ is PC:*

$$g(\omega) = \prod_{j < p(|\omega|)} f(\omega, j).$$

Proof. We identify f with a function $\mathbf{Inputs} \times \mathbb{W} \times \{+,-\} \to \mathbb{N}$. Then

$$g(\omega) = \prod_{j<P} (f(\omega, j, +) - f(\omega, j, -)) = \sum_{\alpha \in \{+,-\}^P} (-1)^{\|\alpha\|} \cdot \prod_{j<P} f(\omega, j, \alpha_j),$$

where we have abbreviated $P = p(|\omega|)$ and $\|\alpha\|$ for the number of $-$'s in α. Thus, now identifying g with a function $\mathbf{Inputs} \times \{+,-\} \to \mathbb{N}$, we have

$$g(\omega, \sigma) = \sum_{\alpha \in \{+,-\}^P} 1\Big[(-1)^{\|\alpha\|} = \sigma\Big] \cdot \prod_{j<P} f(\omega, j, \alpha_j)$$

where $1[\cdot]$ denotes a 0-1 indicator function.

The function

$$g_1(\omega, \alpha, j) \coloneqq f(\omega, j, \alpha_j)$$

is easily seen to be PC, since f is. Thus so is

$$g_2(\omega, \alpha) \coloneqq \prod_{j<P} g_1(\omega, \alpha, j),$$

by product-marginalization as in Proposition C.1. From this, it's easy to see that

$$g_3(\omega, \sigma, \alpha) \coloneqq 1\Big[(-1)^{\|\alpha\|} = \sigma\Big] \cdot g_2(\omega, \alpha)$$

is PC. (We simply compute whether $\|\alpha\|$ has the same parity as σ; if not reject; otherwise, simulate the TM for g_2.) Finally, we have that

$$g(\omega, \sigma) = \sum_{\alpha \in \{+,-\}^P} g_3(\omega, \sigma, \alpha)$$

is PC, by summation-marginalization over α.

(As a small comment on this kind of proof, consider our first claim, that g_1 is PC. Really, here we are assuming g_1's input is "well-formed" in the sense that we have some string $\omega \in \Sigma^*$, then a comma, then $\alpha \in \{+,-\}^{p(|\omega|)}$, then a comma, then $|^j$ for some $1 \leq j \leq p(|\omega|)$. Further, when saying g_1 is "polynomial-time countable", we refer to polynomial-time just in the length of ω, not all input parameters. However, the reader is expected to verify that in the "final construction" of a counting TM for g in terms of the one for f, the TM for g_1 is only ever invoked with well-formed inputs.) □

It is somewhat tedious to prove all the necessary closure results on the way to our final destination (concerning closure under determinants over polynomial rings). We sketch the proof via a sequence of illustrative examples, leaving complete verification to the reader.

Proposition C.3. (Product-marginalization for univariate polynomials over $\mathbb{N}$.) *If $f : \textbf{Inputs} \times \mathbb{W} \to \mathbb{N}[x]$ is PC, and ℓ and d are polynomials (length-bounding and degree-bounding, respectively), then the following $g : \textbf{Inputs} \to \mathbb{N}[x]$ is PC:*

$$g(\omega) = \prod_{j<\ell(|\omega|)} [f(\omega, j)]_{<d(|\omega|)}.$$

Proof. We identify f with a function $\textbf{Inputs} \times \mathbb{W} \times \mathbb{W} \to \mathbb{N}$ and write $L = \ell(|\omega|)$, $D = d(|\omega|)$. Then

$$g(\omega) = \prod_{j<L} \sum_{k<D} f(\omega, j, k) x^k = \sum_{k_0,\ldots,k_{L-1}<D} \left(\prod_{j<L} f(\omega, j, k_j) \right) x^{k_0+\cdots+k_{L-1}}.$$

Thus, now identifying g with a function $\textbf{Inputs} \times \mathbb{W} \to \mathbb{N}$, we have

$$g(\omega, k) = \sum_{\substack{K=(K_0,\ldots,K_{L-1}) \\ K_0,\ldots,K_{L-1}<D}} 1[K_0 + \cdots + K_{L-1} = k] \cdot \prod_{j<L} f(\omega, j, K_j).$$

(Actually, this is only true provided $k < LD$. On one hand, we will never actually "use" $g(\omega, k)$ for $k \geq LD$; on the other hand, for safety we can easily ensure that the TM for g has 0 accepting branches on input (ω, k) with $k \geq LD$.)

We will take care of the summation above via marginalization over sequences, but this requires being slightly careful about how to encode tuples with strings. Given $\omega \in \Sigma^*$, we will identify a tuple K with a string in $\{|, +\}^{LD}$ by padding out the encoding of each $K_j \in \{|\}^*$ with $+$'s to get a length-D string, then concatenating the L encodings. Given $\omega \in \Sigma^*$ and $K \in \{|, +\}^{LD}$, let us say "K is well-formed (given ω)" if K indeed properly encodes a tuple from $[D]^L$ in this way. (Note that we need ω in these definitions so that $L = \ell(|\omega|)$ and $D = d(|\omega|)$ are defined.) We may now write the above as

$$g(\omega, k) = \sum_{K \in \{|,+\}^{LD}} 1[K \text{ is well-formed (given } \omega)] \cdot 1[K_0 + \cdots + K_{L-1} = k] \cdot \prod_{j<L} f(\omega, j, K_j).$$

Note that if K is not well-formed then expressions like "K_j" in the latter factors may not actually make sense. Since it doesn't matter though, we fix a simple convention; say, if the string K is not well-formed then each K_j is defined to be 0.

We are now in a position to finish the proof. The function $(\omega, K, j) \mapsto f(\omega, j, K_j)$ is PC since f is; hence so too is

$$g_1(\omega, K) \coloneqq \prod_{j<L} f(\omega, j, K_j)$$

by product-marginalization. A simple consequence is that

$$g_2(\omega, k, K) \coloneqq 1[K \text{ is well-formed (given } \omega)] \cdot 1[K_0 + \cdots + K_{L-1} = k] \cdot \prod_{j<L} f(\omega, j, K_j)$$

is PC, since it is easy to compute whether K is well-formed and whether $K_0 + \cdots + K_{L-1} = k$. Thus

$$g(\omega, k) = \sum_{K \in \{|,+\}^{LD}} g_2(\omega, k, K)$$

is PC by summation-marginalization over K (using the length-bounding polynomial $\ell \cdot d$). □

We remark that above we used closure under product-marginalization for $\mathbb{N}$-valued functions when multiplying together the coefficients. Since we have the same closure for $\mathbb{Z}$-valued functions, we could have similarly proved Proposition C.5 for polynomials in $\mathbb{Z}[x]$. We will use a similar observation in the following proposition:

Proposition C.4. (Closure under determinants over $\mathbb{Z}$.) *If $f : \mathbf{Inputs} \to \mathbb{Z}^{\mathbb{W} \times \mathbb{W}}$ is PC, and m is a polynomial, then the following $g : \mathbf{Inputs} \to \mathbb{Z}$ is PC:*

$$g(\omega) = \det([f(\omega)]_{<M,M}),$$

where $M = m(|\omega|)$.

Proof. Thinking of f as a function $\mathbf{Inputs} \times \mathbb{W} \times \mathbb{W} \to \mathbb{Z}$, we have

$$g(\omega) = \sum_{\pi \in S_M} \operatorname{sgn}(\pi) \prod_{j<M} f(\omega, j, \pi_j).$$

Similar to the previous proof, given ω we will encode permutations $\pi \in S_M$ as strings in $\{|,+\}^{M^2}$. Again we will use the terminology "π is well-formed", and we will make an arbitrary convention about ill-formed strings; say, that they are taken to represent the identity permutation in S_M. Thus

$$g(\omega) = \sum_{\pi \in \{|,+\}^{M^2}} 1[\pi \text{ is well-formed}] \cdot \operatorname{sgn}(\pi) \prod_{j<M} f(\omega, j, \pi_j).$$

Since f is a PC (as a function with range $\mathbb{Z}$), so too is

$$g_1(\omega, \pi) \coloneqq \prod_{j<M} f(\omega, j, \pi_j),$$

by product-marginalization for $\mathbb{Z}$-valued functions (i.e., for GapP). As sgn and well-formedness of π are computable in $\operatorname{poly}(|\omega|)$ time, it's easy to conclude that

$$g_2(\omega, \pi) = 1[\pi \text{ is well-formed}] \cdot \operatorname{sgn}(\pi) \cdot g_1(\omega, \pi)$$

is a PC function with range $\mathbb{Z}$. Thus $g\colon \mathbf{Inputs} \to \mathbb{Z}$ is indeed PC, by summation-marginalization over π. □

Proposition C.5. (Product-marginalization for $\mathbb{N}[\imath]$.) *If $f : \mathbf{Inputs} \times \mathbb{W} \to \mathbb{N}[\imath]$ is PC, and ℓ is a polynomial, then the following $g : \mathbf{Inputs} \to \mathbb{Z}[\imath]$ is PC:*

$$g(\omega) = \prod_{j<\ell(|\omega|)} f(\omega, j).$$

Proof. We can prove this by first thinking of elements of $\mathbb{N}[\imath]$ as degree-1 polynomials over an "indeterminate $\imath$" with natural coefficients. I.e., define $f' : \mathbf{Inputs} \times \mathbb{W} \to \mathbb{N}[x]$ by

$$f'(\omega, j) = (\Re f(\omega, j)) + (\Im f(\omega, j))x.$$

It's easy to see that f' is PC: the TM F' for f' will, on input (ω, j, k), simulate $F(\omega, j, \Re)$ if $k = |^0$ and simulate $F(\omega, j, \Im)$ if $k = |^1$, where F is the TM for f. By product-marginalization for $\mathbb{N}[x]$-valued functions,

$$g'(\omega) = \prod_{j<L} f'(\omega, j)$$

is PC, where $L = \ell(|\omega|)$. That is,

$$g'(\omega, j) = [x^j] \left(\prod_{j<L} f'(\omega, j) \right)$$

is an $\mathbb{N}$-valued PC function. Now note that

$$g(\omega) = \sum_{\substack{j<L \\ j \bmod 4 = 0}} g'(\omega, j) - \sum_{\substack{j<L \\ j \bmod 4 = 2}} g'(\omega, j) + \imath \sum_{\substack{j<L \\ j \bmod 4 = 1}} g'(\omega, j) - \imath \sum_{\substack{j<L \\ j \bmod 4 = 3}} g'(\omega, j).$$

Now to show that g is PC, given the TM G' for g' the TM G for g will act as follows: On input, say, $(\omega, \Im, +)$, nondeterministically branch over all $j < \ell(|\omega|)$ with $j \bmod 4 = 1$ and simulate $G'(\omega, j)$. Similarly for $(\omega, \Im, -)$ and $(\omega, \Re, \pm)$. □

It's not hard to extend the above to product-marginalization for $\mathbb{Z}[i]$, too. We now consider adjoining the fraction $\frac{1}{2}$; we take it to denote an indeterminate for polynomials, but introduce a natural associated closure rule:

Proposition C.6. (Taking a polynomial over $\frac{1}{2}$ to a common denominator.) *If $f : \mathbf{Inputs} \to \mathbb{N}[\frac{1}{2}]$ is* PC*, and d is a polynomial, then the following $g : \mathbf{Inputs} \to \mathbb{N}[\frac{1}{2}]$ is* PC*:*

$$g(\omega) = C \cdot \left(\tfrac{1}{2}\right)^{D-1},$$

where $D = d(|\omega|)$ and $C \in \mathbb{N}$ is the unique natural number such that $[f(\omega)]_{<D} = g(\omega)$ when both are viewed in the natural way as nonnegative rational numbers.

Proof. Think of $f : \mathbf{Inputs} \times \mathbb{W} \to \mathbb{N}$ and let F be the associated TM. We now define the TM G for g as follows: On input (ω, k), the$G$ first computes D. If $k \neq D-1$ then G simply rejects ("outputs the coefficient 0"). Otherwise, G will want to halt with

$$C = 2^{D-1} f(\omega, 0) + 2^{D-2} f(\omega, 1) + \cdots + 2^0 f(\omega, D-1)$$

accepting computation paths. This is straightforward: G first nondeterministically branches over all $0 \leq k < D$, then it nondeterministically branches over all strings $\alpha \in \Sigma^{D-1-k}$, then it simulates $f(\omega, k)$. □

We remark that the above result also holds analogously for $\mathbb{Z}[\frac{1}{2}]$ because "taking the positive/negative part" commutes with "putting to a common denominator"; e.g.,

$$(a^+ - a^-) + \frac{1}{2}(b^+ - b^-) = \frac{1}{2}(2a^+ + b^+) - \frac{1}{2}(2a^- + b^-).$$

Similarly for $(\mathbb{Z} + \imath\mathbb{Z})[\frac{1}{2}]$ and $(\mathbb{Z} + \imath\mathbb{Z})[\frac{1}{2}, x]$.

Proposition C.7. (Product-marginalization for polynomials over $\mathbb{N}[\frac{1}{2}][x]$.) *If $f : \mathbf{Inputs} \times \mathbb{W} \to \mathbb{N}[\frac{1}{2}][x]$ is* PC*, and ℓ, d, and b are polynomials, then the following $g : \mathbf{Inputs} \to \mathbb{N}[\frac{1}{2}][x]$ is* PC*:*

$$g(\omega) = \prod_{j<L} [f(\omega, j)]_{<D,B}.$$

Here $L = \ell(|\omega|)$ and similarly for D, B. Further, $[f(\omega, j)]_{<D,B}$ refers to the element of $\mathbb{Z}[i, \frac{1}{2}][x]$ given by truncating the polynomial $f(\omega, j)$ to degree less than D (in x) and its coefficients to degree less than B (in $\frac{1}{2}$).

Proof. We follow the proof of Proposition C.5 regarding product-marginalization for $\mathbb{N}[x]$, but now the coefficient ring is $\mathbb{N}[\frac{1}{2}]$. Identifying f with a function $\mathbf{Inputs} \times \mathbb{W} \times \mathbb{W} \to \mathbb{N}[\frac{1}{2}]$ and g with a function $\mathbf{Inputs} \times \mathbb{W} \to \mathbb{N}[x]$, we have

$$g(\omega, k) = \sum_{K \in \{|,+\}^{LD}} 1[K \text{ is well-formed (given } \omega)] \cdot 1[K_0 + \cdots + K_{L-1} = k] \cdot \prod_{j<L} [f(\omega, j, K_j)]_{<B} \quad (6)$$

(provided $k < LD$; 0 otherwise). Certainly $h : \mathbf{Inputs} \times \{|,+\}^{LD} \times \mathbb{W} \to \mathbb{N}[\frac{1}{2}]$ defined by

$$h(\omega, K, j) = f(\omega, j, K_j)$$

is PC since f is. Thus by appeal to Proposition C.5 (with $\frac{1}{2}$ as the indeterminate), we know that $h_1 : \mathbf{Inputs} \times \{|,+\}^{LD} \to \mathbb{N}[\frac{1}{2}]$ defined by

$$h_1(\omega, K) = \prod_{j<L} [h(\omega, K, j)]_{<B}$$

is PC as well. In more detail, h_1 is PC when viewed as the following function $\mathbf{Inputs} \times \{|,+\}^{LD} \times \mathbb{W} \to \mathbb{N}$:

$$h_1(\omega, K, c) = [(\tfrac{1}{2})^c] \prod_{j<L} [f(\omega, j, K_j)]_{<B}$$

(technically, with the further constraint that $h_1(\omega, K, c) = 0$ if $c \geq LB$). It's now straightforward to deduce from (6) using summation-marginalization that g is PC. □

By combining the ideas in all previous positions, we may straightforwardly (albeit tediously) deduce the following:

Theorem C.8. *Suppose f is a polynomial-time countable sequence of matrices whose entries are univariate polynomials in indeterminate z with coefficients from $(\mathbb{Z} + \imath\mathbb{Z})[\frac{1}{2}]$; i.e., $f : \mathbf{Inputs} \times (\mathbb{W} \to \mathbb{W}) \to (\mathbb{Z} + \imath\mathbb{Z})[\frac{1}{2}][z]$ is PC. Then (informally speaking) the sequence formed by taking the determinant of these matrices and clearing denominators is also polynomial-time countable. More precisely, let ℓ, d, and b be polynomials and suppose we consider each $f(\omega)$ to be truncated to an $L \times L$ matrix of polynomials of degree less than D in z, with coefficients in $(\mathbb{Z}+\imath\mathbb{Z})[\frac{1}{2}]$ having powers of $\frac{1}{2}$ up to but not including B. (Here $L = \ell(|\omega|)$ and similarly for D, B.) Then the two functions $\Re g, \Im g : \mathbf{Inputs} \times \mathbb{W} \to \mathbb{Z}$ defined by*

$$[z^k] \det(f(\omega)) = \frac{\Re g(\omega, k) + \imath \cdot \Im g(\omega, k)}{2^{LB}}$$

are in GapP.

ECCC ISSN 1433-8092
http://eccc.hpi-web.de