

# On beating the hybrid argument\*

Bill Fefferman<sup>†</sup>

California Institute of Technology

Christopher Umans<sup>§</sup>

California Institute of Technology

Ronen Shaltiel<sup>‡</sup>

University of Haifa

Emanuele Viola<sup>¶</sup>

Northeastern University

## Abstract

The *hybrid argument* allows one to relate the *distinguishability* of a distribution (from uniform) to the *predictability* of individual bits given a prefix. The argument incurs a loss of a factor  $k$  equal to the bit-length of the distributions:  $\epsilon$ -distinguishability implies only  $\epsilon/k$ -predictability. This paper studies the consequences of avoiding this loss – what we call “beating the hybrid argument” – and develops new proof techniques that circumvent the loss in certain natural settings. Specifically, we obtain the following results:

1. We give an instantiation of the Nisan-Wigderson generator (JCSS '94) that can be broken by quantum computers, and that is  $o(1)$ -unpredictable against  $AC^0$ . This is not enough to imply indistinguishability via the hybrid argument because of the hybrid-argument loss; nevertheless, we conjecture that this generator indeed fools  $AC^0$ , and we prove this statement for a simplified version of the problem. Our conjecture implies the existence of an oracle relative to which BQP is not in the PH, a longstanding open problem.
2. We show that the “INW” generator by Impagliazzo, Nisan, and Wigderson (STOC '94) with seed length  $O(\log n \log \log n)$  produces a distribution that is  $1/\log n$ -unpredictable against poly-logarithmic width (general) read-once oblivious branching programs. Thus avoiding the hybrid-argument loss would lead to a breakthrough in generators against small space.
3. We study pseudorandom generators obtained from a hard function by *repeated sampling*. We identify a property of functions, “resamplability,” that allows us to beat the hybrid argument, leading to new pseudorandom generators for  $AC^0[p]$  and similar classes. Although the generators have sub-linear stretch, they represent the best-known generators for these classes.

Thus we establish that “beating” or bypassing the hybrid argument would have two significant consequences in complexity, and we take steps toward that goal by developing techniques that indeed beat the hybrid argument in related (but simpler) settings, leading to best-known PRGs for certain complexity classes.

---

\*Parts of this work first appeared in the manuscript [FU10].

<sup>†</sup>wjf@cs.caltech.edu. Supported by IQI.

<sup>‡</sup>ronen@cs.haifa.ac.il. Supported by BSF grant 2004329 and ISF grant 686/07.

<sup>§</sup>umans@cs.caltech.edu. Supported by NSF CCF-0846991.

<sup>¶</sup>viola@ccs.neu.edu. Supported by NSF grant CCF-0845003.

# 1 Introduction

The *hybrid argument* [GM84] (see also [Yao82, BM84] and [Gol01] for an exposition) is a powerful proof technique that has widespread applications in cryptography and complexity theory. Suppose we have a random variable  $Z = (Z_1, Z_2, \dots, Z_k) \in \{0, 1\}^k$  that can be distinguished from the uniform distribution on  $k$  bits,  $U$ , by a function  $D$ , i.e.,

$$|\Pr[D(Z) = 1] - \Pr[D(U) = 1]| \geq \epsilon.$$

We are interested in predicting  $Z_i$  from a prefix  $Z_{1, \dots, i-1}$  with some advantage over random guessing. The hybrid argument (in its most basic form) reasons about this via *hybrid distributions*

$$H_i \stackrel{\text{def}}{=} (Z_1, Z_2, \dots, Z_i, U_{i+1}, U_{i+2}, \dots, U_k).$$

Since  $H_0 = U$  and  $H_k = Z$ , using the triangle inequality we get

$$\epsilon \leq |\Pr[D(Z) = 1] - \Pr[D(U) = 1]| \leq \sum_{i=1}^k |\Pr[D(H_{i-1}) = 1] - \Pr[D(H_i) = 1]|,$$

which means that  $D$  distinguishes two adjacent hybrids,  $H_{i-1}$  and  $H_i$ , with gap at least  $\epsilon/k$ . From here, it is easy to convert  $D$  into a closely related function that predicts  $Z_i$  from the prefix with advantage  $\epsilon/k$  over random guessing. The contrapositive is that *unpredictability* implies *indistinguishability*. The canonical application of this argument is to the construction of pseudorandom generators, where it is often easier to design an unpredictable  $Z$  (e.g., from a hard or one-way function) than to argue directly about indistinguishability [BM84, Yao82, GL89, HILL99, Nis91, NW94]. The hybrid argument is also crucially used in the inductive arguments underlying pseudorandom generators against space-bounded computation [Nis92, INW94].

The power of the hybrid argument lies in its generality: it is a generic tool, making no assumptions about the random variable  $Z$  or the complexity of  $D$ . But this generality comes with a price: the factor  $k$  multiplicative loss in passing from the distinguishability of  $Z$  from  $U$ , to the distinguishability of two adjacent hybrids. This loss is negligible when  $k \ll 1/\epsilon$ , which is a common setting for constructions of pseudorandom generators under super-polynomial hardness assumptions. But the loss is a major stumbling block when  $k$  is comparable to, or much larger than,  $1/\epsilon$ . In this case (for example) the loss prevents us from obtaining small-seed generators against various low-level circuit classes for which known lower bounds are not strong enough to withstand the loss (see [SV10]).

It is reasonable to guess that if one imposes restrictions on the type of  $Z$ 's distribution, or on the complexity of  $D$ , this loss might be lessened or avoided entirely – what we call “beating the hybrid argument.”<sup>1</sup> *In this paper we show that two longstanding open problems in complexity would be resolved by beating the hybrid argument.* The first concerns the problem

---

<sup>1</sup>Barak, Shaltiel, and Wigderson [BSW03] were the first to show that this is possible, if  $D$  is a small *PH*-circuit or oblivious bounded-width branching program, and additionally  $D$  is a particular strong, “nearly one-sided” distinguisher.

of constructing an oracle relative to which BQP is not in the Polynomial-time Hierarchy (PH), and the second concerns the problem of constructing pseudorandom generators for space. In each setting, the fact that the hybrid argument is the bottleneck is not obvious; to show that it is a bottleneck, we construct a non-standard instantiation of the Nisan-Wigderson (NW) generator [NW94] that can be broken by quantum computers in the first setting, and we modify the standard analysis of the Impagliazzo-Nisan-Wigderson (INW) generator [INW94] in the second. These two settings are discussed in more detail in §1.1.

We then pursue a program of determining when the hybrid loss can be avoided, by imposing natural restrictions on the distribution  $Z$ , and on the complexity of the distinguisher  $D$ . The complexity classes we consider are certain “low” complexity classes between  $AC^0$  and  $L$ . We show that the hybrid loss can indeed be avoided entirely when  $Z$  is obtained by *repeated sampling* from the distribution  $(U, f(U))$ , for hard functions  $f$  that enjoy a special type of random self-reducibility we dub *resamplability*. We then show that a variety of natural functions  $f$  are resamplable, such as parity and majority, the latter corresponding to a special case of our central conjecture (Conjecture 2.6) concerning the BQP vs. PH problem.

Even though we are only studying a relatively simple class of distributions  $Z$ , *our techniques are already powerful enough to obtain new, best-known pseudorandom generators for  $AC^0[p]$  and other classes*. Although our generators have sublinear stretch, they improve on the folklore generators one can obtain by using known hardness results and applying the hybrid argument – as we later summarize in Table 1. These results demonstrate that the hybrid argument can be beaten in settings close to what is needed for the two applications, and develop techniques that may be useful in tackling the distributions that arise in those applications.

## 1.1 Two consequences of beating the hybrid argument

We now describe two longstanding open problems in complexity that would be resolved by beating the hybrid argument. We also outline the main ideas in the technical development needed to establish the hybrid argument as the bottleneck.

### 1.1.1 An oracle relative to which BQP is not in the PH

The quest for an oracle relative to which BQP is not in the PH dates to the foundational papers of the field; the question was first asked by Bernstein and Vazirani [BV97] in the early 1990’s. Currently, oracles are known relative to which BQP is not in MA [Wat00], but no relativized worlds are known in which BQP is not in AM. Obtaining an oracle relative to which BQP is not in the PH thus represents a stubborn, longstanding and fundamental problem whose resolution would help clarify the relationship between BQP and classical complexity classes. In recent progress, Aaronson [Aar10b] devised a *relation* oracle problem that lies in the function version of BQP but not in the function version of the PH, but this still leaves the original problem open.<sup>2</sup>

---

<sup>2</sup>Aaronson [Aar10b] also proposed the “Generalized Linial-Nisan Conjecture” as a possible route to obtaining the desired oracle; this conjecture turned out to be false in general [Aar10a]. The viability of our approach is unaffected by this development.

In this paper we will speak almost exclusively about the “scaled down” version of the problem, which is equivalent via the well-known connection between PH and  $AC^0$ . In it, the goal is to design a promise problem (rather than an oracle) that lies in (promise)-BQLOGTIME but not (promise)- $AC^0$ . The class BQLOGTIME is the class of languages decidable by quantum computers that have random access to an  $N$ -bit input, and use only  $O(\log N)$  steps (see §2 for the formal definition). As in [Aar10b], our goal will be to design, for each input length  $N$ , a *distribution* on  $N$ -bit strings that can be distinguished from the uniform distribution by a BQLOGTIME predicate, but not by a (quasipolynomial-size)  $AC^0$  circuit. As described in Appendix B, such a distribution can be easily converted to a proper oracle  $O$  for which  $BQP^O \not\subseteq PH^O$ . To obtain such a distribution, we prove two main statements:

1. we generalize the setting of [Aar10b] to a simple framework in which any efficiently quantumly computable unitary  $U$  gives rise to a distribution that can be distinguished from uniform by a quantum computer (Aaronson’s setup is recovered by choosing  $U$  to be a DFT matrix), and
2. we give an explicit construction of unitary matrices whose row-supports form a Nisan-Wigderson design, and we show how to realize these matrices with small quantum circuits in §2.3. This is the technical core of the quantum section.

In our framework, these unitaries give rise to a distribution that is an instantiation of the NW PRG, with MAJORITY as its hard function, and we conjecture (Conjecture 2.6) that this distribution is indeed pseudorandom for  $AC^0$ . The quantitative loss in the hybrid argument is the only thing standing in the way of proving this conjecture, and thus resolving the oracle BQP vs. PH problem. In §4 we make a step towards resolving our conjecture, by showing that it is true for the simpler case in which the sets in the design for the NW generator are disjoint.

### 1.1.2 Pseudorandom generators for small-width branching programs

A longstanding open problem is to design log-space pseudorandom generators that stretch a seed of  $O(\log n)$  bits into  $n$  pseudorandom bits that are indistinguishable from uniform by polynomial-width (read-once oblivious) branching programs. Such generators would yield  $RL = L$ , settling a major open problem in complexity theory. Existing constructions of pseudorandom generators [Nis92, INW94] fail to reach this goal because they use seeds of length  $O(\log^2 n)$ . Despite significant effort, no improvement in the seed length has been achieved even when restricting attention to constant-width branching programs, although a recent, exciting line of works makes progress if the branching programs are constrained further [BRRY10, KNP10, BV10].

In this work we show that the INW generator [INW94] can be adapted to use seed length  $O(\log n \cdot \log \log n)$  and produce an  $n$  bit distribution in which each position cannot be predicted with advantage  $1/\log n$  (given the previous positions) by poly-logarithmic width branching programs. Thus, bypassing the loss of the hybrid argument would yield a breakthrough in pseudorandom generators for small-width branching programs. In §3 we elaborate on this approach.

## 1.2 New pseudorandom generators by beating the hybrid argument

Following the seminal work of [Nis91] a long line of research is concerned with pseudorandom generators against various classes of circuits. We say that a distribution  $Z$  on  $t$  bits is  $\epsilon$ -pseudorandom for a class of circuits  $\mathcal{C}$  if for every circuit  $C$  in  $\mathcal{C}$ ,

$$|\Pr[C(Z) = 1] - \Pr[C(U_t) = 1]| \leq \epsilon.$$

A function  $G : \{0, 1\}^d \rightarrow \{0, 1\}^t$  is  $\epsilon$ -pseudorandom for  $\mathcal{C}$  if  $G(U_d)$  is  $\epsilon$ -pseudorandom.

We will be mostly interested in classes of constant-depth circuits for various choices of allowed gates. For many of these classes there are known circuit lower bounds which can be used to construct pseudorandom generators. More precisely, let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a function with *hardness*  $\delta$  against some class  $\mathcal{C}$  (meaning that every circuit in the class errs on at least  $(1/2 - \delta) \cdot 2^n$  inputs). It is immediate that the function  $G_f(x) = (x, f(x))$  is a  $\delta$ -pseudorandom generator for  $\mathcal{C}$ . One way to improve the stretch of this generator is by *repeated sampling*, namely:

$$G_f^{\otimes k}(x_1, \dots, x_k) := ((x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_k, f(x_k))).$$

The pseudorandomness of  $G_f^{\otimes k}$  follows by the hybrid argument as long as  $k \leq 1/\delta$  and it stretches a seed of length  $nk$  into  $nk + k$  bits.<sup>3</sup> The repeated sampling generator can be viewed as a special case of the NW generator in which the sets of the design are all disjoint. The NW generator reduces the seed length of the generator from  $nk$  to  $\approx n^2$  which is beneficial whenever  $k \gg n$ . However, for  $k \gg n$  the analysis of the NW generator relies on the hybrid argument on  $n$  bits, which in turn requires hardness  $\delta \leq 1/n$  to get a meaningful result.

For some constant-depth circuit classes (that we mention below) the best-known lower bounds only achieve hardness  $\delta \geq \sqrt{1/n}$ . In such cases repeated sampling produces the best-known generators and their stretch  $k$  is bounded by  $1/\delta$ . In §4 we observe that this loss is inherent in proofs that use the hard function  $f$  in a black-box fashion.

Our main technical contribution in this direction is developing new proof techniques that breaks this barrier and allows us to show that  $G_f^{\otimes k}$  is pseudorandom even for  $k > 1/\delta$ . As a consequence we obtain improved pseudorandom generators for several circuit classes. These are summarized in Table 1 which also includes a comparison to the best previous results. Let us make a couple of remarks. First, as we mentioned before, the best previous results are obtained by analyzing the repeated-sampling generator via the hybrid argument. Second, in our pseudorandom generators we also exploit the fact that the hardness results hold for circuits of almost exponential size. This allows choosing  $k$  to be almost exponential in the input length of the function, maximizing the stretch obtained by repeated sampling.

The hard functions and circuit classes we consider are:

---

<sup>3</sup>Note that not only does the hybrid argument loss overwhelm when  $k \gg 1/\delta$ , but a complexity class powerful enough to compute majority can break the “repeated sampling” generator by aggregating (weak) predictions of  $f(x_i)$  from  $x_i$  over all  $i$ . This demonstrates that we must critically use limitations on the power of the class, which we do.

Table 1: Pseudorandom generators fooling circuits of size  $\text{poly}(n)$  on  $n$  bits.

Seed length of generators fooling $\text{poly}(n)$ -size circuits on $n$ bits.		
Type of circuits	Hybrid argument	This work
$\text{AC}^0[p]$ , $p$ prime	$n - n^{1/3}$	$n - n/\text{poly log } n$ (Cor. 4.6,4.17)
$\text{AC}^0$ with $n^{o(1)}$ majority gates	$n - n^{1/3}$	$n - n^\beta, \forall \beta < 1$ (Cor. 4.8)
$\text{AC}^0[6]$	$n - \text{poly log } n$ (under $L \not\subseteq \text{AC}^0[6]$ )	$n - n^\beta, \forall \beta < 1$ (Cor. 4.18) (under $L \not\subseteq \text{AC}^0[6]$ )

**Majority.** The  $n$ -bit majority function has hardness  $\tilde{O}(1/\sqrt{n})$  for  $\text{AC}^0$  [Hås87] (the notation  $\tilde{O}$  hides polylogarithmic factors), and this is tight as shown by the simple circuit that just outputs a bit of the input. We prove that, in fact, the pseudorandomness of  $G_{\text{majority}}^{\otimes k}$  does not decay with  $k$ :  $\text{AC}^0$  circuits cannot distinguish  $k$  independent copies of  $(U_n, \text{majority}(U_n))$  from uniform with any constant advantage, for any  $k = \text{poly}(n)$ . This is the special case of Conjecture 2.6 we mentioned in §1.1.1.

**Parity.** The  $n$ -bit parity function is known to have hardness  $\leq \tilde{O}(1/\sqrt{n})$  for the class  $\text{AC}^0[p]$  for every prime  $p \neq 2$  [Raz87, Smo87, Smo93]. Here  $\text{AC}^0[p]$  stands for  $\text{AC}^0$  circuits augmented with mod  $p$  gates. Whether this bound can be improved is a major, twenty-year-old open problem. Using the hybrid argument, one can only stretch  $\sqrt{n} \cdot n$  bits to  $\sqrt{n} \cdot (n + 1)$  bits, corresponding to a seed length  $n - n^{1/3}$  for  $n$  output bits. We are not aware of any better results. Similarly to the case of MAJORITY, we prove that  $G_{\text{parity}}^{\otimes k}$  remains  $\tilde{O}(1/\sqrt{n})$ -pseudorandom for any  $k \leq 2^{n^{o(1)}}$ . This translates into an improved seed length of  $n - n/\text{poly log } n$  for  $n$  output bits.

We obtain similar result for the class of  $\text{AC}^0$  circuits with few (a small polynomial number of) majority gates. Here the hardness of parity is proved in [ABFR94, Bei94], and it is known to be tight (because of the correlation between the functions majority and parity).

**Determinant.** We consider the determinant function over  $\text{GF}(2)$  for a certain distribution  $M$  of input matrices due to Ishai and Kushilevitz [IK00, IK02]. They show this problem to be hard for  $L$  (logarithmic space), and in fact complete for the complexity for the class parity  $- L$ . Using their beautiful machinery we show that the pseudorandomness of  $k$  independent copies of  $(M, \text{determinant}(M))$  does not decay with  $k$  for the class  $\text{AC}^0[2]$ . We use this to get PRGs for  $\text{AC}^0[2]$  (which could not be obtained via parity, since parity is in  $\text{AC}^0[2]$ ) and also some conditional results for  $\text{ACC}^0 := \cup_m \text{AC}^0[m]$ . We elaborate on the latter, focusing on  $\text{AC}^0[6]$  for simplicity. For  $\text{AC}^0[6]$ , essentially no lower bounds are known. Using standard techniques, under the assumption that  $L \not\subseteq \text{AC}^0[6]$ , one can prove the existence of functions in  $L$  that have hardness  $\Omega(1/\text{poly log } n)$ . This leads to PRGs with seed length  $n - \text{poly log } n$  and  $n$  output bits, and again our PRGs via repeated sampling lead to an improvement.<sup>4</sup>

<sup>4</sup>Recently, a breakthrough result by Williams [Wil10b] shows that  $\text{NEXP}$  is not contained in  $\text{ACC}^0$ . This result however does not seem to imply efficient generators.

Although our generators all have sublinear stretch, they do show that counting approximately the number of satisfying assignments to various types of circuits on  $n$  bits can be solved somewhat more efficiently than was known before, in time  $2^{n-m}$  for various  $m = m(n) = \omega(\log n)$ . A recent, exciting work by [Wil10a] highlights the importance of this type of savings, by showing that it implies new circuit lower bounds in certain settings. We also remark that our generators are all *seed extending* (meaning that the output includes the seed), and it was shown in [KvMS09] that such generators which stretch  $n$  bits into  $n+k$  bits give a polynomial time deterministic simulation that errs on few inputs when simulating randomized algorithms that on input length  $n$  use only  $k$  random bits.

An important open problem is whether our approach can be strengthened to handle the NW generator with even slightly non-disjoint sets. This would reduce the seed length and give improved stretch. Also, as mentioned earlier, achieving this goal in the case of MAJORITY and quasipolynomial-size  $AC^0$  suffices to obtain an oracle relative to which BQP is not in PH.

**The role of resamplability.** We are interested in analyzing the pseudorandomness of the repeated sampling generator while avoiding the loss of the hybrid argument. As mentioned briefly in §1.2, we cannot hope to beat the hybrid argument unless we use specific (non-black-box) properties of the function  $f$ . In this work we identify one such property, the ability to *resample* the function. Informally, we say that a function  $f$  is resamplable if there is a (randomized) procedure  $R$  that on input  $x$  produces a distribution  $R(x)$  that is uniformly distributed over  $\{x' : f(x') = f(x)\}$ . (We stress that we are interested in procedures  $R$  that can be implemented in low circuit classes and in particular cannot compute  $f$ .)

Resamplability is thus a special type of random self-reducibility, a well-studied concept in complexity theory (see, e.g., [FF93] and the references therein). In particular, it is easy to see that resamplability allows us to relate the average-case hardness of  $f$  to its worst-case hardness.

Our approach using resamplability yields the following : Let  $f$  be a function that is resamplable in a class  $\mathcal{C}$  and has hardness  $\delta > 0$  against  $\mathcal{C}$ . We show that the repeated sampling generator  $G_f^{\otimes k}$  remains  $\delta$ -pseudorandom for  $\mathcal{C}$  as long as  $k$  is smaller than the size bound of circuits in  $\mathcal{C}$ . This analysis beats the hybrid argument as it allows choosing  $k \gg 1/\delta$ . The results in §4 are obtained by identifying hard functions that have (known) efficient resamplers, and then applying arguments that rely on them being resamplable.

**Organization of this paper.** In §2 we state our conjecture about a certain instantiation of the Nisan-Wigderson generator fooling quasipolynomial-size  $AC^0$ , and we prove that this conjecture would yield an oracle separating BQP from PH. In §3 we discuss our results about pseudorandom generators for space-bounded computation. In §4 we show how to beat the hybrid argument for certain repeated sampling generators, proving along the way a special case of the conjecture mentioned above. The appendix contains some auxiliary results for the quantum section which are discussed in §2.

## 2 Toward an oracle relative to which BQP is not in the PH

In this section we discuss our results regarding the BQP vs. PH problem. We start with some standard preliminaries.

### 2.1 Preliminaries

A *unitary* matrix is a square matrix  $U$  with complex entries such that  $UU^* = I$ , where  $U^*$  is the conjugate transpose. Equivalently, its rows (and columns) form an orthonormal basis. We name the standard basis vectors of the  $N = 2^n$ -dimensional vectorspace underlying an  $n$ -qubit system by  $|v\rangle$  for  $v \in \{0, 1\}^n$ . A *local* unitary is a unitary that operates only on  $b = O(1)$  qubits; i.e. after a suitable renaming of the standard basis by reordering qubits, it is the matrix  $U \otimes I_{2^{n-b}}$ , where  $U$  is a  $2^b \times 2^b$  unitary  $U$ . A local unitary can be applied in a single step of a quantum computer. A *local decomposition* of a unitary is a factorization into local unitaries. We say an  $N \times N$  unitary is *efficiently quantumly computable* if this factorization has at most  $\text{poly}(n)$  factors.

A *quantum circuit* applies a sequence of local unitaries (“gates”) where each gate is drawn from a fixed, finite set of gates. There are universal finite gate sets for which any efficiently quantumly computable unitary can be realized (up to exponentially small error) by a  $\text{poly}(n)$ -size quantum circuit [KSV02].

**Definition 2.1** (BQLOGTIME). *A language  $L$  is in BQLOGTIME if it can be decided by a LOGTIME-uniform family of circuits  $\{C_n\}$ , where each  $C_n$  is a quantum circuit on  $n$  qubits. On an  $(N = 2^n)$ -bit input  $x$ , circuit  $C_n$  applies  $O(\log N)$  gates, with each gate being either a query gate which applies the map  $|i\rangle|z\rangle \mapsto |i\rangle|z \oplus x_i\rangle$ , or a standard quantum gate (from a fixed, finite basis). It is equivalent, by polynomially padding the number of qubits, to allow  $\text{poly} \log(N)$  gates.*

In this paper, the only manner in which our BQLOGTIME algorithm will access the input string  $x$  is the following operation, which “multiplies  $x$  into the phases”. There are three steps: (1) query with the query register clean, which applies the map  $|i\rangle|0\rangle \mapsto |i\rangle|0 \oplus x_i\rangle$  (note each  $x_i$  is in  $\{0, 1\}$ ); (2) apply to the last qubit the map  $|0\rangle \mapsto -|0\rangle, |1\rangle \mapsto |1\rangle$ ; (3) query again to uncompute the last qubit. When we speak of “multiplying  $x$  into the phase” it will be linguistically convenient to speak about  $x$  as a vector with entries from  $\{+1, -1\}$ , even though one can see from this procedure that the actual input is a 0/1 vector.

The following lemma will be useful repeatedly. It states (essentially) that a block diagonal matrix, all of whose blocks are efficiently quantumly computable, is itself efficiently quantumly computable. This is trivial when all of the blocks are identical, but not entirely obvious in general.

**Lemma 2.2.** *Fix  $N = 2^n$  and  $M = 2^m$ . Let  $U$  be an  $N \times N$  block diagonal matrix composed of the blocks  $U_1, U_2, \dots, U_M$ , where each  $U_i$  is a  $N/M \times N/M$  matrix that has a  $\text{poly}(n)$ -size quantum circuit, a description of which is generated by a uniform  $\text{poly}(n)$  time procedure, given*

input  $i$ . Then given three registers of  $m$  qubits,  $n - m$  qubits, and  $\text{poly}(n)$  qubits, respectively, with the third register initialized to  $|000 \dots 0\rangle$ , there is a  $\text{poly}(n)$  size uniform quantum circuit that applies  $U$  to the first two registers and leaves the third unchanged.

*Proof.* Fix a finite universal set of quantum gates, of cardinality  $d$ , each of which operates on at most  $b$  qubits. A convenient notion will be that of an *oblivious* circuit, in which we fix an ordering (say, lexicographic) on  $[n]^b$ , and the steps of the circuit are identified with  $\text{poly}(n)$  cycles through this list: when we are on step  $(a_1, a_2, \dots, a_b) \in [n]^b$  in one of these cycles, we operate on qubits  $a_1, a_2, \dots, a_b$ . Clearly, any (uniform) quantum circuit can be converted to a (uniform) “oblivious” circuit with at most an  $n^b$  blowup by inserting dummy identity gates.

Let  $n^k$  be an upper bound on the size of the oblivious circuits obtained in this way for the various  $U_i$ . The circuit for each  $U_i$  is now a sequence

$$j^{(i)} = \left( j_1^{(i)}, j_2^{(i)}, j_3^{(i)}, \dots, j_{n^k}^{(i)} \right),$$

with each  $j_\ell^{(i)} \in [d]$  specifying which gate to apply at step  $\ell$  in the oblivious circuit for  $U_i$  (and because the circuit is oblivious, the qubits to which this gate should be applied are easily determined from  $\ell$ ). Let  $f : [M] \rightarrow [d]^{n^k}$  be the function that maps  $i$  to the vector  $j^{(i)}$ .

Now we describe the promised efficient quantum procedure:

1. Apply the map derived from  $f$  that takes  $|i\rangle|z\rangle$  to  $|i\rangle|z \oplus f(i)\rangle$ , to the first and third register. We view the contents of the third register as a vector in  $[d]^{n^k}$ .
2. Repeat for  $\ell = 1, 2, 3, \dots, n^k$ : apply the “controlled unitary” that consults the  $\ell$ -th component of the third register, and applies the specified gate to qubits  $(a_1, a_2, \dots, a_b)$  of the second register (again,  $(a_1, a_2, \dots, a_b)$  are easily determined from  $\ell$  because the circuit is oblivious). The important observation is that this “controlled unitary” operates on only constantly many qubits.
3. Repeat step 1 to uncompute the auxiliary information in the third register.

□

## 2.2 The quantum algorithm

We give a general framework allowing one to turn any efficiently quantumly computable unitary into a distribution that can be distinguished from uniform by a BQLOGTIME machine. Our framework generalizes the setup in [Aar10b].

Let  $A$  be any  $N \times N$  matrix with entries<sup>5</sup> in  $\{0, 1, -1\}$  and pairwise orthogonal rows, and define  $S(A, i)$  to be the support of the  $i$ -th row of matrix  $A$ . Define  $\overline{A}$  to be the matrix  $A$  with entries in row  $i$  scaled by  $1/\sqrt{|S(A, i)|}$ , and observe that  $\overline{A}$  is a unitary matrix.

---

<sup>5</sup>We could extend this framework to matrices with general entries, but we choose to present this restriction since it is all we need.

Define the random variable  $D_{A,M} = (x, z)$  distributed on  $\{+1, -1\}^{2N}$  by picking  $x \in \{+1, -1\}^N$  uniformly, and setting the next  $N$  bits to be  $z \in \{+1, -1\}^N$  defined by  $z_i = \text{sgn}((Ax)_i) = \text{sgn}((\bar{A}x)_i)$  for  $i \leq M$  and  $z_i$  independently and uniformly random in  $\{+1, -1\}$  for  $i > M$ .

It will be convenient to think of  $M = N$  initially; we analyze the general case because we will eventually need to handle  $M = N/2$ . Below, we use  $U_{2N}$  to denote the random variable uniformly distributed on  $\{+1, -1\}^{2N}$ .

**Theorem 2.3.** *Let  $N = 2^n$  for an integer  $n > 0$ , and let  $M = \Omega(N)$ . For every matrix  $A \in \{0, 1, -1\}^{N \times N}$  with pairwise orthogonal rows, there is a BQLOGTIME algorithm  $Q_A$  that distinguishes  $D_{A,M}$  from  $U_{2N}$ ; i.e., there is some constant  $\epsilon > 0$  for which*

$$|\Pr[Q_A(D_{A,M}) = 1] - \Pr[Q_A(U_{2N}) = 1]| > \epsilon.$$

*The algorithm is uniform if  $A$  comes from a uniform family of matrices.*

*Proof.* The input to the algorithm is a pair of strings  $x, z \in \{+1, -1\}^N$ .

The algorithm performs the following steps:

1. Enter a uniform superposition  $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} |i\rangle$  and multiply  $x$  into the phase to obtain  $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} x_i |i\rangle$ .
2. Apply  $\bar{A}$  to obtain  $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} (\bar{A}x)_i |i\rangle$ .
3. Multiply  $z$  into the phase to obtain  $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} z_i (\bar{A}x)_i |i\rangle$ .
4. Define vector  $w$  by  $w_i = \frac{1}{\sqrt{N}} z_i (\bar{A}x)_i$ . Apply the  $N \times N$  Hadamard<sup>6</sup>  $H$  to obtain  $\sum_{i \in \{0,1\}^n} (Hw)_i |i\rangle$ , and measure in the computational basis. Accept iff the outcome is  $0^n$ .

We first argue that the acceptance probability is small in case  $(x, z)$  is distributed as  $U_{2N}$ . This follows from a symmetry argument: for fixed  $x$ , and  $w$  as defined in Step 4 above, the vector  $Hw$  above has every entry identically distributed, because  $z$  is independently chosen uniformly from  $\{-1, +1\}^N$  and every row of  $H$  is a vector in  $\{-1, +1\}^N$ . In particular this implies that the random variable  $(Hw)_i^2$  is identically distributed for all  $i$ . Together with the fact that  $\sum_i (Hw)_i^2 = 1$ , we conclude that  $E[(Hw)_i^2] = 1/N$ . Then by Markov, with probability at least  $1 - 1/\sqrt{N}$  we accept with probability at most  $\sqrt{N}/N$ , for an overall acceptance probability of at most  $2/\sqrt{N}$ .

Next, we argue that the acceptance probability is large in case  $(x, z)$  is distributed as  $D_{A,M}$ . Here we observe that for  $i \leq M$ ,  $w_i = \frac{1}{\sqrt{N}} |(\bar{A}x)_i|$  and hence  $E[w_i] = \frac{1}{\sqrt{N \cdot |S(A, i)|}} \Omega(\sqrt{|S(A, i)|}) = \Omega(1/\sqrt{N})$  (since before scaling,  $w_i$  is just the distance from the origin of a random walk on the line, with  $|S(A, i)|$  steps). For  $i > M$ , we simply have  $E[w_i] = 0$ . Then  $E[\sum_i w_i] =$

---

<sup>6</sup>This is the matrix  $H$  whose rows and columns are indexed by  $\{0, 1\}^n$ , with entry  $(i, j)$  equal to  $-1^{(i, j)}/\sqrt{N}$ .

$M \cdot \Omega(1/\sqrt{N}) = \Omega(\sqrt{N})$ , so  $E[(Hw)_{0^n}] = \Omega(1)$ . Since the random variable  $(Hw)_{0^n}$  is always bounded above by 1, we can apply Markov to its negation to conclude that with constant probability, it is *at least* a constant  $\epsilon$  (and in such cases the acceptance probability is at least  $\epsilon^2$ ). Overall, the acceptance probability is  $\Omega(1)$ .  $\square$

The BQLOGTIME algorithm for what Aaronson calls FOURIER CHECKING in [Aar10b] is recovered from the above framework by taking  $A$  to be a DFT matrix (and  $M = N$ ).

## 2.3 Unitary matrices with large, nearly-disjoint row supports

In light of Theorem 2.3, our task is now to construct a unitary  $A$  for which the associated distribution fools  $AC^0$ . A natural source for distributions that fool  $AC^0$  is the NW pseudorandom generator. In this section, we show how to “realize” an instantiation of the NW generator as an efficiently quantumly computable unitary. We need the following standard definitions:

**Definition 2.4** ([NW94]). *A set family  $\mathcal{D} = \{S_1, S_2, \dots, S_m\}$  is an  $(\ell, p)$  design if every set in the family has cardinality  $\ell$ , and for all  $i \neq j$ ,  $|S_i \cap S_j| \leq p$ .*

**Definition 2.5** ([NW94]). *Given a function  $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$  and an  $(\ell, p)$  design  $\mathcal{D} = \{S_1, S_2, \dots, S_m\}$  in a universe of size  $t$ , the function  $NW_{\mathcal{D}}^f : \{0, 1\}^t \rightarrow \{0, 1\}^m$  is given by*

$$NW_{\mathcal{D}}^f(x) = (f_1(x|_{S_1}), f_2(x|_{S_2}), f_3(x|_{S_3}), \dots, f_m(x|_{S_m})),$$

where each  $f_i$  is the function  $f$  with a fixed set of its inputs negated<sup>7</sup>, and  $x|_S$  denotes the projection of  $x$  to the coordinates in the set  $S$ .

Generally speaking, the function  $NW_{\mathcal{D}}^f$  is a PRG against a class of distinguishers as long as  $f$  is hard on average for that class of distinguishers.

Below we construct unitary matrices  $A$  with the property that all or “almost all” of the row supports  $S(A, i)$  are large and have bounded intersections. We also show that these unitaries are efficiently quantumly computable; this is the technical core of this section. The distribution  $D_{A, M}$  (it will turn out that  $M$  will be half the underlying dimension) can then easily be seen to be the distribution  $(U_N, NW_{\mathcal{D}}^{\text{MAJORITY}})$ , and we would like to argue that this distribution fools quasipolynomial-size  $AC^0$ . MAJORITY is indeed hard for (exponential-size)  $AC^0$ , but the quantitative loss in the hybrid argument stands in the way of proving such a statement by known techniques. This is because majority on  $\ell$  bits is only  $\tilde{O}(1/\sqrt{\ell})$  hard, and we output many more than  $\sqrt{\ell}$  bits.

Nevertheless, we conjecture that the distribution  $D_{A, M}$  fools constant-depth circuits. Since we aim for an oracle separation, and there is a quasi-polynomial relationship between oracle PH machines and  $AC^0$  circuits, we consider  $AC^0$  circuits of quasipolynomial size.

---

<sup>7</sup>The standard setup has each  $f_i = f$ ; we need the additional freedom in this paper for technical reasons. We know of no settings in which this alteration affects the analysis of the NW generator.

**Conjecture 2.6.** *Let  $\mathcal{D} = \{S_1, S_2, \dots, S_m\}$  be an  $(\ell, O(1))$ -design in a universe of size  $t \leq \text{poly}(\ell)$ , with  $m \leq \text{poly}(\ell)$ . Then for every constant-depth circuit of size at most  $\exp(\text{poly} \log m)$ ,*

$$|\Pr[C(U_{t+m}) = 1] - \Pr[C(U_t, \text{NW}_{\mathcal{D}}^{\text{MAJORITY}}(U_t)) = 1]| \leq o(1).$$

Using the hybrid argument, a distinguishing circuit  $C$  with gap  $\epsilon$  can be converted to a predictor with advantage  $\epsilon/m$  and then (via the standard arguments in [Nis91, NW94]) into a slightly larger circuit that computes MAJORITY with success rate  $1/2 + \epsilon/m$ . Thus the above statement is true for  $m \leq o(\sqrt{\ell})$ ; if the  $1/m$  loss from the hybrid argument can be avoided (or reduced), it would be true for  $m$  as large as  $\text{poly}(\ell)$  (and even larger) as we conjecture is true.

### 2.3.1 The paired-lines construction

We describe a collection of  $q^2/2$  pairwise-orthogonal rows, each of which is a vector in  $\{0, +1, -1\}^{q^2}$ . We identify  $q^2$  with the affine plane  $\mathbb{F}_q \times \mathbb{F}_q$ , where  $q = 2^n$  for an integer  $n > 0$ . Let  $B_1, B_2$  be an equipartition of  $\mathbb{F}_q$ , and let  $\phi : B_1 \rightarrow B_2$  be an arbitrary bijection. Our vectors are indexed by a pair  $(a, b) \in \mathbb{F}_q \times B_1$ , and their coordinates are naturally identified with  $\mathbb{F}_q \times \mathbb{F}_q$ :

$$v_{a,b}[x, y] = \begin{cases} -1 & y = ax + b \\ +1 & y = ax + \phi(b) \end{cases} \quad (1)$$

Notice that  $v(a, b)$  is  $-1$  on exactly the points of  $\mathbb{F}_q \times \mathbb{F}_q$  corresponding to the line with slope  $a$  and  $y$ -intercept  $b$ , and  $+1$  on exactly the points of  $\mathbb{F}_q \times \mathbb{F}_q$  corresponding to the line with slope  $a$  and  $y$ -intercept  $\phi(b)$ . So each  $v(a, b)$  is supported on exactly a pair of parallel lines. Orthogonality will follow from the fact that every two non-parallel line-pairs intersect in exactly one point, as argued in the proof of the next lemma.

**Lemma 2.7.** *The vectors defined in Eq. (1) are pairwise orthogonal, and their supports form a  $(2q, 4)$  design.*

*Proof.* Consider  $(a, b) \neq (a', b')$ . If  $a = a'$  then the supports of  $v(a, b)$  and  $v(a, b')$  are disjoint. Otherwise  $a \neq a'$  and there are exactly four intersection points (obtained by solving linear equations over  $\mathbb{F}_q$ ):

- $(x = (b' - b)/(a - a'), y = ax + b) = (x = (b' - b)/(a - a'), y = a'x + b')$ , which contributes  $(-1) \cdot (-1) = 1$  to the inner product, and
- $(x = (b' - \phi(b))/(a - a'), y = ax + \phi(b)) = (x = (b' - \phi(b))/(a - a'), y = a'x + b')$ , which contributes  $(+1) \cdot (-1) = -1$  to the inner product, and
- $(x = (\phi(b') - b)/(a - a'), y = ax + b) = (x = (\phi(b') - b)/(a - a'), y = a'x + \phi(b'))$ , which contributes  $(-1) \cdot (+1) = -1$  to the inner product, and
- $(x = (\phi(b') - \phi(b))/(a - a'), y = ax + \phi(b)) = (x = (\phi(b') - \phi(b))/(a - a'), y = a'x + \phi(b'))$ , which contributes  $(+1) \cdot (+1) = 1$  to the inner product.

The sum of the contributions to the inner product from these four points is zero. The computation of the support size is straightforward.  $\square$

In Appendix A, we give another construction (which is not needed for our main result) in which the number of vectors is exactly equal to the dimension of the underlying space (giving rise to a unitary in which “all rows participate” instead of only half of the rows). However, we leave as an open problem obtaining a local decomposition of the associated unitary.

### 2.3.2 A local decomposition

We now describe an  $q^2 \times q^2$  unitary matrix that is efficiently quantumly computable and has the (normalized) vectors  $v(a, b)$  from Eq. (1) as  $q^2/2$  of its  $q^2$  rows. We recall that  $q = 2^n$  for an integer  $n > 0$ .

**Proposition 2.8.** *The following  $q \times q$  unitary matrices are efficiently quantumly computable:*

1. *the DFT matrix  $F$  with respect to the additive group of  $\mathbb{F}_q$  and its inverse, and*
2. *the  $q \times q$  unitary matrix  $B$  with  $\frac{1}{\sqrt{2}}(I_{q/2} | - I_{q/2})$  as its first  $q/2$  rows,  $\frac{1}{\sqrt{4}}(I_{q/4} | - I_{q/4} | I_{q/4} | - I_{q/4})$  as its next  $q/4$  rows,  $\frac{1}{\sqrt{8}}(I_{q/8} | - I_{q/8} | I_{q/8} | - I_{q/8} | I_{q/8} | - I_{q/8} | I_{q/8})$  as its next  $q/8$  rows, etc... and whose last row is  $\frac{1}{\sqrt{N}}(1, 1, 1, \dots, 1)$ .*

*Proof.* The DFT matrices are well-known to be efficiently quantumly computable. For the second one we make use of the Hadamard matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}.$$

Let  $B_i$  be the  $q \times q$  identity matrix with its lower right  $2^i \times 2^i$  submatrix replaced by the matrix  $H \otimes I_{2^{i-1}}$ . Each  $B_i$  is efficiently quantumly computable by Lemma 2.2. It is then easy to verify that  $B = B_1 B_2 B_3 \cdots B_n$ .  $\square$

**Lemma 2.9.** *Let  $\alpha$  be a generator of the multiplicative group of  $\mathbb{F}_q$ . For  $c \in \mathbb{F}_q$ , let  $D_c$  denote the  $q \times q$  diagonal matrix*

$$\frac{1}{\sqrt{q}} \cdot \text{diag} \left( \sqrt{q}, (-1)^{\text{Tr}(\alpha^1 \cdot c)}, (-1)^{\text{Tr}(\alpha^2 \cdot c)}, (-1)^{\text{Tr}(\alpha^3 \cdot c)}, \dots, (-1)^{\text{Tr}(\alpha^{q-1} \cdot c)} \right),$$

and let  $D'_c$  denote the  $q \times q$  diagonal matrix

$$\frac{1}{\sqrt{q}} \cdot \text{diag} \left( 0, (-1)^{\text{Tr}(\alpha^1 \cdot c)}, (-1)^{\text{Tr}(\alpha^2 \cdot c)}, (-1)^{\text{Tr}(\alpha^3 \cdot c)}, \dots, (-1)^{\text{Tr}(\alpha^{q-1} \cdot c)} \right).$$

Then the  $q^2 \times q^2$  matrix  $D$  whose  $(i, j)$  block (with  $i, j \in \mathbb{F}_q$ ) equals  $D_{ij}$  if  $i = j$  and  $D'_{ij}$  otherwise, is efficiently quantumly computable.

*Proof.* Consider the  $q^2 \times q^2$  block-diagonal matrix that has as its  $(k, k)$  block the matrix whose  $(i, j)$  entry is  $(-1)^{\text{Tr}(ij\alpha^k)}$  for  $k \in \{1, 2, \dots, q-1\}$  and whose  $(0, 0)$  block is  $I_q$ . Each such block except the  $(0, 0)$  block is the DFT matrix  $F$  with its rows (or equivalently, columns) renamed according to the map  $j \mapsto j\alpha^k$ . The  $F$  matrix is efficiently quantumly computable and the map  $j \mapsto j\alpha^k$  is classically and reversibly (and thus quantumly) efficiently computable. Thus each  $q \times q$  block on the diagonal is efficiently quantumly computable. By Lemma 2.2 the entire matrix is efficiently quantumly computable.

If we index columns by  $(i, i') \in (\mathbb{F}_q)^2$  and rows by  $(j, j') \in (\mathbb{F}_q)^2$ , then the desired matrix  $D$  is the above block-diagonal matrix with the order of the two indexing coordinates for the rows transposed, and the order of the two indexing coordinates for the columns transposed.  $\square$

Our main theorem follows:

**Theorem 2.10.** *The  $q^2 \times q^2$  matrix  $(I_q \otimes B) \cdot (I_q \otimes F) \cdot D \cdot (I_q \otimes F^{-1})$ , which is efficiently quantumly computable, has the vectors  $v(a, b)$  from Eq. (1) as  $q^2/2$  of its rows<sup>8</sup>.*

*Proof.* Let  $S_c$  be the  $q \times q$  permutation matrix  $S_c$  that (when multiplied on the right) shifts columns, identified with  $\mathbb{F}_q$ , by the map  $x \mapsto x + c$ . Let  $J$  be the all-ones matrix. The main observation is that

$$FD_cF^{-1} = \frac{1}{\sqrt{q}}S_c - \frac{\sqrt{q}-1}{q}J,$$

and that

$$FD'_cF^{-1} = \frac{1}{\sqrt{q}}S_c - \frac{1}{\sqrt{q}}J.$$

Thus the final matrix has in its  $(i, j)$  block (with  $i, j \in \mathbb{F}_q$ ) the matrix

$$B \cdot \left( \frac{1}{\sqrt{q}}S_{ij} - \frac{\sqrt{q}-1}{q}J \right)$$

if  $i = j$ , and

$$B \cdot \left( \frac{1}{\sqrt{q}}S_{ij} - \frac{1}{\sqrt{q}}J \right)$$

otherwise. Observe that  $BJ$  has all zero entries except for the last row, so in particular, the first  $q/2$  rows of the  $(i, j)$  block are  $(1/\sqrt{2q})(I_{q/2} - I_{q/2})S_{ij}$ . Therefore the  $q/2$  rows of the entire  $q^2 \times q^2$  matrix corresponding to the top halves of blocks  $(i, j)$  as  $j$  varies, give the vectors  $v(i, b)$  for  $b \in B_1$ , if we identify columns with  $\mathbb{F}_q \times \mathbb{F}_q$  as follows: columns of the  $j$ -th block are identified with  $\{j\} \times \mathbb{F}_q$ , and within the  $j$ -th block,  $B_1$  is the first  $q/2$  columns and  $B_2$  is the next  $q/2$  columns (and the bijection  $\phi$  maps the element associated with the  $b$ -th column to the element associated with the  $(b + q/2)$ -th column).

Then, as  $i$  varies over  $\mathbb{F}_q$ , we find all of the vectors from Eq. (1) as the “top-halves” of each successive set of  $q$  rows of the large matrix.  $\square$

---

<sup>8</sup>To be precise, these are the  $v(a, b)$  with respect to *some* equipartition  $B_1, B_2$  and *some* bijection  $\phi$ .

## 2.4 Putting it all together

**Theorem 2.11.** *Assuming Conjecture 2.6, for every  $N$  there is a distribution  $D_N$  on  $N$  bits such that:*

1. *there is a BQLOGTIME algorithm that distinguishes  $D_N$  from the uniform distribution on  $N$  bits with probability  $\geq \Omega(1)$ ;*
2. *for every  $d$ , any depth- $d$   $AC^0$  circuit of size  $\exp(\log^d N)$  has vanishing ( $o(1)$ ) advantage in distinguishing  $D_N$  from uniform.*

*Proof.* We only construct  $D_N$  for certain lengths – one can extend the construction to work for every length by padding.

Let  $A$  be the matrix of Theorem 2.10, and set  $N = q^2$  and  $M = N/2$ . By Theorem 2.3, there is a BQLOGTIME algorithm that distinguishes  $D_{A,M}$  from the the uniform distribution  $U_{2N}$ .

By Lemma 2.7, the first  $M$  rows of  $A$  have supports forming a  $(2\sqrt{N}, 4)$ -design  $\mathcal{D}$ . It is also clear that for  $i \leq M$ , the  $(N+i)$ -th bit of  $D_{A,M}$  computes MAJORITY (with a fixed pattern of inputs negated) on those among the first  $N$  bits that lie in  $S(A, i)$ . Thus  $D_{A,M}$  is exactly the distribution  $(U_N, NW_{\mathcal{D}}^{\text{MAJORITY}}(U_N))$  followed by  $N/2$  additional independent random bits (which can have no impact on the distinguishability of the distribution from uniform). Thus by Conjecture 2.6, no constant-depth, quasipolynomial-size circuit can distinguish  $D_{A,M}$  from  $U_{2N}$ . This concludes the proof.  $\square$

The proof of the next corollary is in Appendix B.

**Corollary 2.12.** *Assuming Conjecture 2.6, there is an oracle  $O$  such that  $BQP^O \not\subseteq PH^O$ .*

## 3 Toward PRGs with seed $o(\log^2 n)$ for small space

In this section we prove that the pseudorandom generator construction of [INW94] with seed length  $O(\log n \log \log n)$  yields distributions that are unpredictable by poly  $\log n$ -width branching programs. We start by reviewing the construction [INW94] as presented in [RR99]. This requires the standard notions of “width  $S$  read-once oblivious branching programs” (abbreviated ROBPs) and “ $(k, \eta)$ -extractors”. Both these notions are defined next.

**Width  $S$  read-once oblivious branching programs (ROBPs).** These are directed graphs where the node set  $V$  is partitioned into  $n+1$  layers  $V_0, \dots, V_n$  each of size at most  $S$ . Each node  $v$  in layer  $i < n$  has two outgoing edges (one labelled by “0” and one labelled by “1”) that go to nodes in layer  $i+1$ . On input  $x \in \{0, 1\}^n$ , such a graph defines a path from the first node in the first layer (which we think of as the starting node) to a node in layer  $n+1$  by following the edges labelled by  $x$  one by one. The output is the number of arrived node in layer  $n+1$ .

**$(k, \eta)$ -extractors.** These were introduced in [NZ96] and are functions  $E : \{0, 1\}^r \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  with the property that for every distribution  $X$  over  $\{0, 1\}^r$  that is uniform over a set of size  $\geq 2^k$ , the distribution  $E(X, U_d)$  is  $\eta$ -close to uniform.

**The INW generator.** Let  $S = 2^s$  be the width of the ROBPs that we aim to fool. Let  $\eta > 0$  be a parameter that we determine later. Let  $r_0 = C \cdot (s + \log(1/\eta))$  where  $C \geq 1$  is a constant to be determined later. The construction will rely on  $(r - s - \log(1/\eta), \eta)$ -extractors  $E : \{0, 1\}^r \times \{0, 1\}^d \rightarrow \{0, 1\}^r$  for  $r \geq r_0$  and  $d = O(s + \log(1/\eta))$ . We stress that there are explicit constructions with these parameters for a sufficiently large universal constant  $C$  [GW97], and that the dependence of  $d$  on both  $s$  and  $\eta$  is optimal up to constant factors.

For  $0 \leq j \leq \log n$  we define functions  $G_j : \{0, 1\}^{r_0+jd} \rightarrow \{0, 1\}^{2^j}$  iteratively as follows:  $G_0(x)$  is defined to be the first bit of  $x$ . For  $j > 0$ , we think of the input of  $G$  as a concatenation of two strings:  $x \in \{0, 1\}^{r_0+(j-1)d}$ ,  $y \in \{0, 1\}^d$  and set

$$G_j(x, y) := G_{j-1}(x) \circ G_{j-1}(E(x, y)).$$

The final generator is given by  $G := G_{\log n}$  and has seed length  $O(\log n(s + \log(1/\eta)))$  and output length  $n$

**The analysis and the  $\log^2 n$  barrier.** The analysis of [INW94] shows that for every  $j$ , if  $G_j$  is a pseudorandom generator with error  $\epsilon_j$  then  $G_{j+1}$  is a pseudorandom generator with error  $2\epsilon_j + \eta$ . Summing up, this gives that the error of  $G = G_{\log n}$  is bounded by  $O(n\eta)$ , which forces setting  $\eta < 1/n$  to get a meaningful result. This setting implies in turn that the seed length is at least  $\Omega(\log^2 n)$  even for constant  $s$ . Recent work by [BRRY10, BV10, KNP10] shows that for restricted classes of small width ROBPs the INW generator described above yields pseudorandom generators with seed length  $\tilde{O}(\log n)$ . The key is that for restricted classes of branching programs (like *regular* branching programs) a tighter connection between the error of  $G_j$  and  $G_{j+1}$  can be made, improving the bound on the distinguishing error of the final generator. However, [BV10] show that these constructions cannot achieve seed length  $o(\log^2 n)$  for *general* branching programs, in the sense that there are choices of extractors  $E$  for which the INW generator cannot be pseudorandom even for constant-width ROBPs if we set  $\eta$  much larger than  $1/n$  so as to obtain seed length  $o(\log^2 n)$ .

The main technical contribution of this section is to show that if the goal is unpredictability instead of indistinguishability, then INW can be shown to work with seed length  $\tilde{O}(\log n)$ .

**Shooting for an unpredictable distribution.** We now consider the goal of showing that the output of  $G$  is “unpredictable” meaning that no width  $S = 2^s$  ROBP can predict the  $i$ ’th bit with advantage larger than some parameter  $\delta$ .

**Theorem 3.1.** *Fix  $\eta > 0$  and let  $Z = (Z_1, \dots, Z_n)$  denote the output distribution of  $G$  defined above on a uniformly chosen seed. Then, for every width  $S = 2^s$  ROBP  $P$  and every  $0 \leq i \leq n$ ,*

$$\Pr[P(Z_1, \dots, Z_{i-1}) = Z_i] \leq \frac{1}{2} + \delta$$

for  $\delta = O(\eta \log n)$ .

The high level idea of the proof is to show that if  $G_j$  is unpredictable with advantage  $\delta$  then  $G_{j+1}$  is unpredictable with advantage  $\delta + O(\eta)$ . Comparing to the analysis showing

pseudorandomness, the advantage is that we don't double the error when going from level  $j$  to level  $j+1$ . Loosely speaking, this is because the analysis showing unpredictability of  $G_{j+1}$  only pays for one of the two instantiations of  $G_j$ . This allows us to get meaningful results even for relatively large  $\eta \gg 1/n$ . For example, let  $s = O(\log \log n)$  (which gives  $S = (\log n)^{O(1)}$ ) and let  $\eta = 1/\log^2 n$ . For these settings,  $G$  uses a seed of length  $O(\log n \cdot \log \log n)$  and produces a distribution which is unpredictable for  $\delta = O(1/\log n)$ .

The doubling loss mentioned above arises from a use of the hybrid argument (with  $k = 2$ ) in the proof [INW94]. Thus our result can be viewed as avoiding this loss when one imposes the restriction that the distinguisher branching program is a predictor.

*Proof of Theorem 3.1.* Let  $P$  be a width  $S = 2^s$  ROBP. We say that  $P$  predicts  $G_j$  with advantage  $\delta$  if there exists an  $i$  such that  $\Pr[P(Z_1, \dots, Z_{i-1}) = Z_i] > \frac{1}{2} + \delta$  where  $Z_1, \dots, Z_n$  are sampled by applying  $G_j$  on a uniformly chosen seed. We show that:

**Claim 3.2.** *For  $j > 1$  if  $P$  predicts  $G_j$  with advantage  $\delta$  then there exists a width  $S$  ROBP  $P'$  that predicts  $G_{j-1}$  with advantage  $\delta - 2\eta$ .*

Theorem 3.1 follows from Claim 3.2 by noting that if  $P$  predicts  $G = G_{\log n}$  with advantage  $\delta$  then by iteratively applying Claim 3.2 there exists a branching program  $P'$  which predicts  $G_0$  with advantage  $\delta - 2\eta \log n$ . This is a contradiction if the latter quantity is greater than zero.

We now proceed with the proof of Claim 3.2. We have that  $P$  predicts position  $i$  in the output of  $G_j$  from the previous  $i - 1$  positions. Recall that the output of  $G_j$  is obtained by setting  $r = r_0 + (j - 1)d$ , uniformly sampling  $X \in \{0, 1\}^r, Y \in \{0, 1\}^d$  and then

$$G_j(X, Y) = G_{j-1}(X) \circ G_{j-1}(E(X, Y)).$$

If position  $i$  appears in the first half of the output then  $P$  also predicts  $G_{j-1}$  with the same advantage and we are done.

Otherwise, let  $i' = 2^j - 1$  denote the last position in the first application of  $G_{j-1}$  and we have that  $i > i'$ . Let  $W$  denote the random variable defined by considering the node that  $P$  arrives to after reading bits  $1, \dots, i'$ . We say that a node  $w$  at layer  $i' + 1$  is light if  $\Pr[W = w] \leq 2^{-(s+\log(1/\eta))} = \eta/S$ . Note that:

$$\Pr[W \text{ is light}] = \sum_{\text{light } w \in V_{i'+1}} \Pr[W = w] \leq \sum_{w \in V_{i'+1}} \eta/S \leq \eta.$$

It follows by an averaging argument that there exists  $w' \in V_{i'+1}$  which is not light such that  $P$  predicts  $G_j(X, Y)$  with advantage  $\delta - \eta$  even conditioned on event  $\{W = w'\}$ . Note that positions  $1, \dots, i'$  in the output of  $G_j(X, Y)$  depend on  $X$  but not on  $Y$ . Thus, conditioning on  $\{W = w'\}$  amounts to conditioning  $X$  to be in some subset  $T$ . We have that

$$\Pr[W = w'] \geq 2^{-(s+\log(1/\eta))}$$

which gives that  $T \subseteq \{0, 1\}^r$  is of this weight. Therefore, conditioned on  $\{W = w'\}$ ,  $X$  is uniformly distributed in a set of size  $\geq 2^{r-(s+\log(1/\eta))}$  which by the properties of extractors gives that  $E(X, Y)$  is  $\eta$ -close to uniform conditioned on  $\{W = w'\}$ .

Let  $P'$  denote the graph obtained by taking only layers  $i', \dots, i$  from  $P$ . In  $P'$  we set  $w'$  as the starting node (by renaming the nodes in the relevant layer). Note that  $P'$  is a width  $S$  ROBP defined for inputs of length  $i - i' - 1$ . Furthermore,  $P'$  predicts  $G_{j-1}(E(X, Y))$  with advantage  $\delta - \eta$  when conditioned on  $\{W = w'\}$ . As  $E(X, Y)$  is  $\eta$ -close to uniform conditioned on  $\{W = w'\}$ , we conclude that  $P'$  predicts  $G_{j-1}$  with advantage at least  $\delta - 2\eta$  when the input to  $G_{j-1}$  is chosen at random. This concludes the proof.  $\square$

Following Theorem 3.1, an alternative route to pseudorandom generators for small width ROBPs is to convert unpredictability to indistinguishability while avoiding the cost of the hybrid argument. A concrete question is whether the following construction, which applies an extractor to the output of the INW construction, is pseudorandom: let  $G = G_{\log n}$  be the generator from Theorem 3.1 instantiated with  $\eta = 1/\log^{\Theta(1)} n$  and let

$$E' : \{0, 1\}^n \times \{0, 1\}^{O(\log n)} \rightarrow \{0, 1\}^m$$

be a  $(k, 1/n)$ -extractor for  $k, m = n^{\Theta(1)}$  [Zuc97, GUV07]; the final construction is  $G'(x, z) = E'(G(x), z)$ , which has seed length  $O(\log n \cdot \log \log n)$ . The intuition is that an unpredictable distribution has high entropy from the point of view of small width ROBPs and therefore applying an extractor may produce a pseudorandom distribution. (See [BSW03] for a study on using extractors to produce pseudorandom distributions.) This approach is inspired by a pseudorandom generator construction of [STV01] in the setup of small circuits. More precisely, [STV01] instantiate the NW generator with a function that is only mildly hard on average giving a distribution which is unpredictable, but for  $\delta$  which is too large to apply the hybrid argument. They are able to show that applying an extractor on their unpredictable distribution produces a pseudorandom distribution.<sup>9</sup>

## 4 Beating the hybrid argument

In this section we show how to beat the hybrid argument for the “repeated sampling generator” in the context of several low-level circuit classes. First we note that even for this goal, it is necessary to use non-black-box techniques. For a distribution  $D$ , we denote by  $D^{\otimes k}$  the concatenation of  $k$  independent samples of  $D$ .

**Fact 4.1** (Hybrid argument loss is inherent in black-box reductions). *There is  $c > 0$  such that for any  $n$  and  $\epsilon \geq 1/2^{n^c}$  such that  $\log 1/\epsilon$  is an integer: there exists a (non-explicit) function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  such that*

1. *for any circuit  $C$  of size  $s \leq 2^{n^c}$ ,  $\Pr_{x \in \{0, 1\}^n}[C(x) \neq f(x)] \geq 1/2 - \epsilon$ , and*
2. *there is a  $\text{poly}(n/\epsilon)$ -size DNF distinguishing  $(X, f(X))^{\otimes c/\epsilon}$  from uniform with probability  $\geq 0.9$ .*

---

<sup>9</sup>For context, we remark that this result in [STV01] is not known to hold for restricted circuit classes such as  $\text{AC}^0[p]$ . The specific proof in [STV01] fails because at its heart lies hardness amplification (specifically the hard-core set lemma [Imp95]) which in these restricted classes is either not known to hold or false [SV10, LTW07].

*Proof.* Let  $x = (y, z)$  where  $|y| = \log(1/\epsilon) + 1$ , and  $|z| = n - |y| \geq n/2$  (for  $c$  large enough). Let  $h : \{0, 1\}^{|z|} \rightarrow \{0, 1\}$  be a function such that for a universal constant  $d$ , any circuit  $D$  of size  $\leq 2^{n/d}$  satisfies  $\Pr_{x \in \{0, 1\}^n}[D(x) \neq f(x)] \geq 1/2 - 1/2^{n/d}$ . The existence of such a function  $h$  follows from a counting argument.

Now define  $f(y, z)$  as  $h(z)$  if  $y \neq 0$ , and 0 otherwise.

To see (1), note that for any circuit  $C$  of size  $\leq 2^{n/d}$ , if  $\Pr_{x \in \{0, 1\}^n}[C(x) = f(x)] \geq 1/2 + \epsilon$  then

$$\begin{aligned} 1/2 + \epsilon &\leq \Pr[C(x) = f(x)] \leq \Pr[C(y, z) = f(y, z) | y \neq 0] + \Pr[y = 0] \\ &= \Pr[C(y, z) = h(z) | y \neq 0] + \epsilon/2, \end{aligned}$$

and so there exists a fixed  $y$  so that, denoting by  $C_y$  the circuit of size  $\leq 2^{n/d}$  obtained by hardwiring  $y$  into  $C$ ,

$$\Pr[C_y(z) = h(z)] \geq 1/2 + \epsilon/2 \geq 1/2 + 1/2^{n/c+1}.$$

This contradicts the hardness of  $h$  (which is  $2^{n/d}$ ) for any  $c > d$  and  $n$  large enough.

To see (2), consider the distributions

$$\bigcirc_{i \leq c/\epsilon}(y^i, z^i, f(y^i, z^i)) \quad (\star)$$

and

$$\bigcirc_{i \leq c/\epsilon}(y^i, z^i, b^i) \quad (U)$$

where  $b^i$  is a uniform random bit, and  $\bigcirc$  denotes concatenation.

In either distribution, we expect  $c/\epsilon \cdot \epsilon/2 = c/2$  values  $y^i$  to be 0. Increasing  $c$ , we can guarantee that with probability arbitrarily close to 1 we will see an arbitrarily large number of  $y^i = 0$ . The CNF  $T$  defined as  $\forall i, y^i = 0 \Rightarrow b^i = 0$  accepts  $(\star)$  with probability 1, by definition. On the other hand,  $T$  accepts  $(U)$  with probability less than 0.01, for a sufficiently large  $c$ , because every clause where  $y^i = 0$  has only probability  $1/2$  of being true.  $\square$

We indeed identify a property of functions that allows us to avoid this loss, *resamplability*. For exposition, it is convenient to work with problems rather than functions:

**Definition 4.2.** A problem  $\Pi = \Pi_Y \cup \Pi_N$  is *resamplable with resources  $T$*  (e.g.,  $T =$  circuits of size  $n^2$ ) if there are functions  $R_r(\cdot)$  such that:

1. for any  $x \in \Pi_Y$  (resp.,  $x \in \Pi_N$ ), the distribution  $R_r(x)$  for uniform  $r$  is uniform in  $\Pi_Y$  (resp.,  $\Pi_N$ ); and
2. for any fixed  $r$ , the function  $R_r(\cdot)$  is computable with resources  $T$ .

The next lemma uses resamplability to prove that the repeated sampling generator suffers *no loss* in the distinguishing parameter. Note that resamplability naturally gives rise to a reduction strategy that would show that distinguishing  $\Pi_Y^{\otimes k}$  from  $\Pi_N^{\otimes k}$  is as hard as distinguishing  $\Pi_Y$  from  $\Pi_N$ ; the innovation in the proof below is that it is able to replace  $\Pi_N^{\otimes k}$  with  $\Pi^{\otimes k}$ .

**Lemma 4.3.** *Suppose a problem  $\Pi = \Pi_Y \cup \Pi_N$  ( $\Pi_Y \cap \Pi_N = \emptyset$ ) has a resampler  $R_r(\cdot)$ . If a function  $C$  distinguishes  $k$  independent samples of  $\Pi_Y$  from  $k$  independent samples of  $\Pi$  with probability  $\epsilon$ , i.e.*

$$|\Pr[C(\Pi_Y^{\otimes k}) = 1] - \Pr[C(\Pi^{\otimes k}) = 1]| \geq \epsilon,$$

*then there is a function  $C'$  of the form  $C'(x) := C(\bar{R}^1(x), \dots, \bar{R}^k(x))$  where each  $\bar{R}^i$  is either the resampler  $R_{r^i}(x)$  for a fixed string  $r^i$ , or is just a constant function  $\pi^i \in \Pi_Y$ , such that  $C'$  distinguishes  $\Pi_Y$  from  $\Pi_N$  with the same probability  $\epsilon$ , i.e.*

$$|\Pr[C'(\Pi_Y) = 1] - \Pr[C'(\Pi_N) = 1]| \geq \epsilon.$$

*Proof.* Let  $B^1, \dots, B^k \in \{Y, N\}$  be independent bits coming up  $Y$  with probability  $|\Pi_Y|/|\Pi|$ . Note that the distribution  $\Pi^1, \Pi^2, \dots, \Pi^k$  equals the distribution  $\Pi_{B^1}^1, \Pi_{B^2}^2, \dots, \Pi_{B^k}^k$ . By averaging, there exists a way to fix each variable  $B^i$  to a value  $b^i$  such that

$$\left| \Pr[C(\Pi_Y^1, \Pi_Y^2, \dots, \Pi_Y^k) = 1] - \Pr[C(\Pi_{b^1}^1, \Pi_{b^2}^2, \dots, \Pi_{b^k}^k) = 1] \right| \geq \epsilon.$$

In both distributions in the above equation, the coordinates where  $b^i = Y$  are the same, and the others are different. Consider the randomized map  $F(x) := (R^1(x), \dots, R^k(x))$  where  $R^i(x)$  is a uniform element of  $\Pi_Y$  if  $b^i = Y$ , and is the resampler  $R_{r^i}(x)$  for a uniform  $r^i$  if  $b^i = N$ . Then the previous equation implies

$$\left| \Pr[C(F(\Pi_Y)) = 1] - \Pr[C(F(\Pi_N)) = 1] \right| \geq \epsilon.$$

Fixing the internal randomness of  $F$  we obtain the desired conclusion for  $C'(\cdot) := C(F(\cdot))$ .  $\square$

We now discuss cases in which resamplability yields new results. We start with the simplest setting, that of the parity function, and then we move to majority and determinant.

## 4.1 A generator based on parity

First we note the efficient resamplability of parity.

**Fact 4.4.** *The problem parity is resamplable in (poly-size)  $\text{NC}^0$ .*

*Proof.* The resampler  $R_r(x)$  uses  $r$  to select a string of length  $|x|$  with parity 0, and then xor's it to  $x$ . For fixed  $r$ , this amounts to complementing some input bits, which can be done in  $\text{NC}^0$ .  $\square$

Combining this fact with Lemma 4.3 we obtain new pseudorandom distributions for low-level circuit classes. We start with the class of  $\text{AC}^0$  circuits with mod  $p$  gates – denoted  $\text{AC}^0[p]$ , for an odd prime  $p$ . The strongest known hardness result for this class is the following well-known, long-standing result by Smolensky [Smo87, Smo93] (cf. [Fil10]).

**Lemma 4.5** ([Smo93]). *For every  $d$  and prime  $p > 2$ , there is a constant  $\alpha > 0$  such that the  $n$ -bit parity function is  $\epsilon$ -hard, with  $\epsilon = n^{-1/2+o(1)}$ , for  $\text{AC}^0[p]$  circuits of size  $\leq 2^{n^\alpha}$ .*

Equivalently if  $X$  is a random variable uniformly distributed on  $\{0, 1\}^n$ , then  $(X, \text{parity}(X))$  is  $\epsilon$ -pseudorandom for such circuits. (Cf. §1.2 for the definition of hard and pseudorandom.) The following corollary shows that this pseudorandomness does not decay with the number of repeated experiments.

**Corollary 4.6.** *Fix a prime  $p \neq 2$  and  $d \geq 1$ . For every  $k \leq 2^{n^{o(1)}}$ , every  $\text{poly}(n, k)$ -size  $\text{AC}^0[p]$  circuit  $C$  of depth  $d$  satisfies*

$$|\Pr [C((X, \text{parity}(X))^{\otimes k}) = 1] - \Pr[C(U) = 1]| \leq o(1),$$

where  $X$  is uniformly distributed on  $\{0, 1\}^n$ , and  $U$  is the uniform distribution over  $k \cdot (n+1)$  bits. Moreover, there is an explicit generator  $G : \{0, 1\}^{n(1-1/\text{poly} \lg n)} \rightarrow \{0, 1\}^n$  that is  $o(1)$ -pseudorandom for  $\text{AC}^0[p]$  circuits  $C$  of depth  $d$  and size  $2^{\lg^d n}$ .

*Proof.* This proof follows from the combination of Lemma 4.3 and Smolensky's Lemma 4.5; for the second claim define  $G(x^1, \dots, x^k) = (x^1, \neg \text{parity}(x^1), \dots, x^k, \neg \text{parity}(x^k))$  (i.e., it outputs a  $k$ -tuple of strings of odd hamming weight) where each  $|x^i| = \text{poly} \lg n$ .  $\square$

For the interesting case of  $p = 2$ , this proof does not work. In §4.3 we obtain similar generators using the machinery of [IK02].

The distribution induced by the generator in Corollary 4.6 has the appealing feature that it can be equivalently generated by an  $\text{NC}^0$  circuit such that each output bit depends on just 2 input bits. This can be obtained using the corresponding “trick” for parity which is explained for example in [Vio10].

We now consider the class  $\text{AC}^0$  with a limited number of majority gates. When the number of majority gates is logarithmic in the size of the circuit, strong (approaching  $1/2$  superpolynomially fast) average-case lower bounds that allow for superpolynomial-stretch generators are known [Vio07]. But when the number of majority gates is larger, say polynomial in the circuit size, the best average-case hardness result remains the one proved by Beigel [Bei94, Corollary 4.4] building on the seminal lower bound by Aspnes, Beigel, Furst, and Rudich [ABFR94].

**Lemma 4.7** ([ABFR94, Bei94]). *For any  $d$  there is  $\alpha > 0$  such that for any And-Or-Majority-Not circuit of depth  $d$ , size  $\leq 2^{n^\alpha}$ , with at most  $n^\alpha$  majority gates,*

$$\Pr_{x \in \{0,1\}^n} [C(x) = \text{parity}(x)] \leq 1/2 + o(1).$$

Actually [Bei94, Corollary 4.4] has  $1/4$  instead of  $o(1)$ , but the same techniques give  $o(1)$ .

Combining Lemmas 4.7, 4.4, and 4.3, and using the fact that the reduction does not increase the number of majority gates one gets new generators for small-depth circuits with few majority gates. We only state the particular tradeoff where the number of majority gates is polynomial.

**Corollary 4.8** (Generators against  $\text{AC}^0$  with few majority gates). *For every  $d \geq 1, \delta \in (0, 1)$  there is  $\epsilon > 0$  such that for large enough  $n$  there are explicit generators  $G : \{0, 1\}^{n(1-1/n^\delta)} \rightarrow$*

$\{0,1\}^n$  such that for any And-Or-Majority-Not circuit  $C$  of depth  $d$ , size  $\leq 2^{n^\epsilon}$ , with  $\leq n^\epsilon$  majority gates,

$$\left| \Pr_{s \in \{0,1\}^{n(1-1/n^\delta)}} [C(G(s)) = 1] - \Pr_{x \in \{0,1\}^n} [C(x) = 1] \right| \leq o(1).$$

## 4.2 A generator based on majority

We begin by remarking that we do *not* know of a resampler for MAJORITY, so this setup is a bit more complicated. We require a generalization of Definition 4.2 and Lemma 4.3, in which the “resampler”  $R_r(\cdot)$  maps a *source* problem  $\Pi_Y \cup \Pi_N$  to a *target* problem  $\Gamma_Y \cup \Gamma_N$ . Part (1) of the definition becomes that for any  $x \in \Pi_Y$  (resp.  $\Pi_N$ ),  $R_r(x)$  is uniform in  $\Gamma_Y$  (resp.  $\Gamma_N$ ); the lemma then reduces an  $\epsilon$ -distinguisher for  $\Gamma_Y^{\otimes k}$  vs.  $\Gamma_N^{\otimes k}$  to an  $\epsilon$ -distinguisher for  $\Pi_Y$  vs.  $\Pi_N$ . It should also be clear that replacing  $R_r(\cdot)$  with a function whose induced output distributions are  $\delta$ -close to the ones in the definition can only alter the distinguishing gap in the conclusion of the lemma by  $\delta k$ .

Our target problem is  $A = A_Y \cup A_N$  defined as follows: for odd  $n$ ,  $A_Y$  is the set of  $n$ -bit strings of hamming weight  $\geq n/2$ , and  $A_N$  is the set of strings of weight  $\leq n/2$ . The following is an easy exercise (cf. [Vio10]):

**Claim 4.9.** *Distinguishing  $A_Y^{\otimes k}$  from  $A_N^{\otimes k}$  AC<sup>0</sup> reduces to distinguishing  $(U, \text{majority}(U))^{\otimes k}$  from uniform, with no loss in the distinguishing parameter.*

Our source problem is  $W = W_Y \cup W_N$  defined as follows: for odd  $\ell$ ,  $W_Y$  is the set of  $\ell$ -bit strings of hamming weight  $(\ell + 1)/2$  and  $W_N$  is the set of those strings of weight  $(\ell - 1)/2$ . Distinguishing  $W_Y$  from  $W_N$  is hard:

**Lemma 4.10** ([Hås87]). *For any constants  $d \geq 1, \epsilon > 0$ ,  $\text{poly}(\ell)$ -size AC<sup>0</sup> circuits of depth  $d$  cannot distinguish  $W_Y$  from  $W_N$  with gap greater than  $\epsilon$ .*

Only a worst-case lower bound is stated in [Hås87], but the stated average-case result follows using standard techniques [Aar10b, SV10]. Specifically, one can use the fact that the problem is resamplable (just permute input bits) and the fact that approximate majority is in AC<sup>0</sup> [Ajt83, ABO84] (cf. [Vio09a]) to show that any small AC<sup>0</sup> circuit distinguishing  $W_Y$  from  $W_N$  with gap  $\epsilon \geq \Omega(1)$  can be transformed into a small AC<sup>0</sup> circuit solving  $W$  in the worst case.

We now give a resampler from  $W$  to  $A$ :

**Lemma 4.11.** *There is a function  $t = \text{poly}(\ell)$  and a distribution  $R_r(\cdot)$  on AC<sup>0</sup> circuits of size  $\text{poly}(\ell)$  mapping  $\ell$  bits to  $n = \ell \cdot t$  bits, such that*

- for any  $x \in W_Y$ ,  $R_r(x)$  has statistical distance  $\exp(-n^{\Omega(1)})$  from uniform in  $A_Y$ , and
- for any  $x \in W_N$ ,  $R_r(x)$  has statistical distance  $\exp(-n^{\Omega(1)})$  from uniform in  $A_N$ .

As a corollary we obtain the following result (which we state for only polynomially many repetitions  $k$ , because this is all that is needed for the special case of Conjecture 2.6).

**Corollary 4.12.** *For any constant  $d \geq 1$  and any function  $k = \text{poly}(n)$ , every  $\text{poly}(n)$ -size  $\text{AC}^0$  circuit  $C$  of depth  $d$  satisfies*

$$|\Pr [C ((X, \text{majority}(X))^{\otimes k}) = 1] - \Pr[C(U) = 1]| \leq o(1),$$

where  $X$  is uniformly distributed on  $\{0, 1\}^n$ , and  $U$  is the uniform distribution over  $k \cdot (n + 1)$  bits.

The proof of Corollary 4.12 follows by combining Lemma 4.10 with the version of Lemma 4.3 discussed above, and Claim 4.9.

*Proof of Lemma 4.11.* Let  $t = t(\ell)$  be odd. For  $i$  chosen according to a distribution to be determined later, let  $R_r(\cdot)$  be defined as follows: concatenate  $2i + 1$  copies of  $x$  with a balanced string on  $(t - (2i + 1))\ell$  bits, and output a random permutation of this string. Note that if  $x \in W_Y$ , the hamming weight of the resulting string is  $(n + 1)/2 + i$ , while if  $x \in W_N$ , the hamming weight of the resulting string is  $(n - 1)/2 - i$ .

Now we give the distribution for choosing  $i$ . The only constraint on  $i$  is that  $2i + 1 \leq t$ . We select  $i$  with the probability given by  $A_Y$  to the set of strings of weight  $(n + 1)/2 + i$  (which is the same as the probability given by  $A_N$  to the set of strings of weight  $(n - 1)/2 - i$ ), normalized to give a probability distribution.

On input any string in  $W_Y$  (resp.  $W_N$ ), the weight distribution of the resampler differs from the weight distribution  $A_Y$  (resp.  $A_N$ ) only by redistributing the probability mass of hamming weights greater than  $(n + 1)/2 + \Theta(t)$  (resp. less than  $(n - 1)/2 - \Theta(t)$ ). Since  $n = \ell t$ , by letting  $t$  be a sufficiently large polynomial in  $\ell$  and using a Chernoff bound, the statistical distance between the distributions is  $\exp(-n^{\Omega(1)})$ .  $\square$

### 4.3 Generators based on L-hardness

We consider the problem, introduced by Ishai and Kushilevitz [IK00, IK02], of distinguishing certain matrixes with full rank from rank full  $-1$ . This problem is used to great effect in several works, e.g. [AIK06, GGH<sup>+</sup>07, GGH<sup>+</sup>08], and we use the name CMD (for connectivity matrix determinant) from [GGH<sup>+</sup>07]. For a self-contained exposition of this problem and the properties we shall need, see [Vio09b, Chapter 4].

**Definition 4.13** ([IK00, IK02]). *An input to the problem  $\Pi = \text{CMD}$  (for connectivity matrix determinant) is an  $n \times n$  matrix  $A$  that has 0/1 entries on the main diagonal and above it, 1 on the second diagonal (one below the main), and 0 below this diagonal. The matrix  $A$  is represented by the  $n(n + 1)/2$  0/1 entries on and above the main diagonal. Each such matrix has rank  $\geq n - 1$ .  $\Pi_Y$  are matrixes with full rank  $n$  over  $GF(2)$ ,  $\Pi_N$  are matrixes with rank  $n - 1$ .*

In [IK02] various useful properties are established. First, note CMD is balanced, i.e.  $|\text{CMD}_Y| = |\text{CMD}_N|$ . To see this, imagine choosing a random matrix in the definition of CMD by first choosing all rows except the first. This yields an  $n - 1$  dimension vector space, and the matrix will have full rank  $n$  if and only if the first row will land outside of this space, which happens with probability  $1/2$ .

Second, CMD is hard for log-space computation, and in fact is complete for the richer complexity class  $\oplus L$ , under  $\text{NC}^0$  reductions, i.e. maps such that each output bits depends on just a constant number of input bits.

**Lemma 4.14** ([IK02]). *CMD is  $\oplus L$ -complete under  $\text{NC}^0$  reductions.*

Finally, the techniques in [IK02] also show that CMD is resamplable in  $\text{AC}^0[2]$ .

**Lemma 4.15** ([IK02]). *CMD is resamplable in poly-size  $\text{AC}^0[2]$ .*

*Proof sketch.* There are two distributions  $A, B$  over  $n \times n$  matrixes such that for every  $M \in \Pi_Y$  (resp.,  $M \in \Pi_N$ ) the product  $AMB$  is uniform over  $\Pi_Y$  (resp.,  $\Pi_N$ ). The resampler is thus  $R_{A,B}(M) := AMB$ . Since the multiplication is over  $\text{GF}(2)$ , this can be computed by a poly-size  $\text{AC}^0[2]$  circuit.  $\square$

We use another result by Smolensky, that majority is hard for  $\text{AC}^0[2]$ . See [Fil10] for an exposition.

**Lemma 4.16** ([Smo93]). *For any  $\text{AC}^0[2]$  circuit  $C$  of size  $s$  and depth  $d$  we have*

$$\Pr_{x \in \{0,1\}^n} [C(x) = \text{majority}(x)] \leq 1/2 + O(\log(Sn))^d / \sqrt{n} + 1/n.$$

We can now state our generator against  $\text{AC}^0[2]$ .

**Corollary 4.17** (Generator against  $\text{AC}^0$  with mod 2 gates). *For every  $d$  there is  $c$  such that for large enough  $n$  there is an explicit generator  $G : \{0, 1\}^{n(1-1/\lg^c n)} \rightarrow \{0, 1\}^n$  such for any  $\text{AC}^0[2]$  of depth  $d$  and size  $2^{\lg^d n}$ :*

$$\left| \Pr_{s \in \{0,1\}^{n(1-1/\lg^c n)}} [C(G(s)) = 1] - \Pr_{x \in \{0,1\}^n} [C(x) = 1] \right| \leq o(1).$$

*Proof.* By Lemma 4.16, circuits of the given resources satisfy

$$\left| \Pr_{x \in \{0,1\}^{\lg^a n}} [C(x) = \text{majority}(x)] - \frac{1}{2} \right| \leq o(1),$$

where the probability is for inputs of length  $\lg^a n$ , for a constant  $a$  depending only on  $p, d$ .

Note that majority is computable in logarithmic space, and recall that CMD is hard for logarithmic space under  $\text{NC}^0$  reductions (Lemma 4.14). In addition, CMD is resamplable

(Lemma 4.15) and hence randomly self-reducible (see comment after Def. 4.2). The combination of these facts implies that circuits of the given resources satisfy

$$\left| \Pr_{x \in \{0,1\}^{\lg^b n}} [C(x) = \text{CMD}(x)] - \frac{1}{2} \right| \leq o(1),$$

where the probability is for inputs of length  $\lg^b n$ , for a constant  $b$  depending only on  $p, d$ . In other words, if there is a small  $\text{AC}^0[2]$  circuit computing  $\text{CMD}$  on a  $1/2 + \alpha$  fraction of inputs, we could compute majority on a  $1/2 + \alpha$  fraction of inputs by first reducing majority to  $\text{CMD}$  in  $\text{NC}^0 \subseteq \text{AC}^0[2]$  and then resampling  $\text{CMD}$  in  $\text{AC}^0[2]$ .

The output of the generator  $G$  is a  $k$ -tuple of strings representing  $\text{CMD}_Y$  instances of  $\text{CMD}$ . Recall that  $\text{CMD}$  is balanced, i.e.  $|\text{CMD}_Y| = |\text{CMD}_N|$ , so the seed length is  $(\lg^b n - 1)n / \lg^b n = n(1 - 1/\lg^b n)$ . The correctness follows from Lemma 4.3.  $\square$

Note that the proof in §4.2 won't work, because  $W$  is solvable just by computing the parity of the instance. It is open if  $(x, \text{majority } x)^{\otimes k}$  is pseudorandom for small  $\text{AC}^0[2]$  circuits for every  $k = \text{poly}(n)$ .

We also get the following conditional result for  $\text{AC}^0[m]$  for every even  $m$ . For simplicity we state it for  $m = 6$ .

**Corollary 4.18** (Conditional generator against  $\text{AC}^0$  with mod 6). *Suppose that  $L \not\subseteq \text{AC}^0[6]$ . Then for every  $d > 1$  and any  $\delta \in (0, 1)$ , for large enough  $n$  there is an explicit generator  $G : \{0, 1\}^{n(1-1/n^\delta)} \rightarrow \{0, 1\}^n$  such for any  $\text{AC}^0[6]$  circuit of depth  $d$  and size  $n^d$ :*

$$\left| \Pr_{s \in \{0,1\}^{n(1-1/n^\delta)}} [C(G(s)) = 1] - \Pr_{x \in \{0,1\}^n} [C(x) = 1] \right| \leq o(1).$$

*Proof.* The assumption implies that depth  $\text{poly}(n)$ -size  $\text{AC}^0[6]$  circuits fail to compute  $\text{CMD}$  with probability  $1/2 - o(1)$ . For otherwise, we could reduce an instance of any fixed log-space problem to  $\text{CMD}$ , then resample  $\text{CMD}$  and use approximate majority [Ajt83, ABO84] to show that  $L \subseteq \text{AC}^0[6]$ .

This allows to use instances of length  $n^\delta$  for an arbitrarily small  $\delta > 0$ . Again, the correctness follows from Lemma 4.3.  $\square$

#### 4.4 On a possible alternative way to get generators for $\text{AC}^0[p]$

In this section we sketch a possible alternative approach to get generators with seed length  $n(1 - 1/\text{poly } \log n)$  that fool  $\text{AC}^0[p]$  circuits,  $p$  prime, on  $n$  bits. As stated in Lemma 4.16, these circuits cannot compute majority on instances of an appropriate length  $\text{poly } \log n$  with probability  $\geq 1/2 - o(1)$ . From this, it follows via techniques by Shaltiel and Viola [SV10] that there exists some integer  $c \leq \text{poly } \log n$  so that the circuits cannot distinguish (with any constant advantage) the uniform distribution from i.i.d. bits coming up 1 with probability  $1/2 + 1/c$ .

If we know  $c$ , the generator that outputs i.i.d. bits coming up 1 with probability  $1/2 + 1/c$  has seed length  $H(1/2 + 1/c)n \leq n(1 - 1/\text{poly log } n)$ , where  $H$  is the binary entropy function. Up to the poly log  $n$ , this is of the same type we get using resamplers.

However, we have been unable to determine if  $c$  is explicitly computable, though it is even possible that most values for  $c$  will do.

**Acknowledgments.** We thank Scott Aaronson, Yi-Kai Liu, and Mike Saks for helpful discussions.

## References

- [Aar10a] S. Aaronson. A counterexample to the Generalized Linial-Nisan Conjecture. In *ECCC TR: Electronic Colloquium on Computational Complexity, technical reports*, number 109, 2010.
- [Aar10b] Scott Aaronson. BQP and the polynomial hierarchy. In *42nd Annual Symposium on Theory of Computing (STOC)*, pages 141–150. ACM, 2010.
- [ABFR94] James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994.
- [ABO84] Miklós Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computation. In *16th ACM Symposium on Theory of Computing (STOC)*, pages 471–474, 1984.
- [AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in  $NC^0$ . *SIAM J. Comput.*, 36(4):845–888, 2006.
- [Ajt83] Miklós Ajtai.  $\Sigma_1^1$ -formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983.
- [Bei94] Richard Beigel. When do extra majority gates help? polylog( $N$ ) majority gates are equivalent to one. *Comput. Complexity*, 4(4):314–324, 1994.
- [BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. on Computing*, 13(4):850–864, November 1984.
- [BRRY10] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. In *51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.
- [BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *7th International Workshop on Randomization and Computation (RANDOM)*, pages 200–215, 2003.

- [BV97] Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.
- [BV10] Joshua Brody and Elad Verbin. The coin problem, and pseudorandomness for branching programs. In *51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.
- [FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. on Computing*, 22(5):994–1005, October 1993.
- [Fil10] Yuval Filmus. Smolensky’s polynomial method, 2010. <http://www.cs.toronto.edu/~yuvalf/Smolensky.pdf>.
- [FU10] B. Fefferman and C. Umans. Pseudorandom generators and the BQP vs. PH problem. Available at <http://www.cs.caltech.edu/~umans/papers/FU10.pdf>, 2010.
- [GGH<sup>+</sup>07] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 440–449, 2007.
- [GGH<sup>+</sup>08] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy Rothblum. A (de)constructive approach to program checking. In *40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 143–152, 2008.
- [GL89] Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.
- [GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.
- [Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.
- [GUV07] Venkat Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh-varjy codes. In *CCC*, pages 96–108, 2007.
- [GW97] Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Struct. Algorithms*, 11(4):315–343, 1997.
- [Hås87] Johan Håstad. *Computational limitations of small-depth circuits*. MIT Press, 1987.
- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudo-random generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

- [IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 294–304, 2000.
- [IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium (ICALP)*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002.
- [Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545. IEEE, 1995.
- [INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *26th ACM Symposium on the Theory of Computing (STOC)*, pages 356–364, 1994.
- [KNP10] M. Koucky, P. Nimbhorkar, and P. Pudlak. Pseudorandom generators for group products. Technical Report TR10-113, ECCO, 2010.
- [KSV02] A.Y Kitaev, A.H Shen, and M.N Vyalvi. *Classical and Quantum Computation*. AMS, 2002.
- [KvMS09] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators and typically-correct derandomization. In *12th International Workshop on Approximation, Randomization, and Combinatorial Optimization (RANDOM)*, volume 5687 of *Lecture Notes in Computer Science*, pages 574–587. Springer, 2009.
- [LTW07] Chi-Jen Lu, Shi-Chun Tsai, and Hsin-Lung Wu. On the complexity of hard-core set constructions. In *34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2007.
- [Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.
- [Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.
- [NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Computer & Systems Sciences*, 49(2):149–167, 1994.
- [NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

- [Raz87] Alexander Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 1987. English translation in *Mathematical Notes of the Academy of Sci. of the USSR*, 41(4):333–338, 1987.
- [RR99] Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *31st ACM Symposium on Theory of Computing (STOC)*, pages 159–168, 1999.
- [Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *19th Annual Symposium on Theory of Computing (STOC)*, pages 77–82. ACM, 1987.
- [Smo93] Roman Smolensky. On representations by low-degree polynomials. In *34th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 130–138, 1993.
- [STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. System Sci.*, 62(2):236–266, 2001.
- [SV10] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM Journal on Computing*, 39(7):3122–3154, 2010.
- [Vio07] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM Journal on Computing*, 36(5):1387–1403, 2007.
- [Vio09a] Emanuele Viola. On approximate majority and probabilistic time. *Computational Complexity*, 18(3):337–375, 2009.
- [Vio09b] Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009.
- [Vio10] Emanuele Viola. The complexity of distributions. In *51th Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2010.
- [Wat00] John Watrous. Succinct quantum proofs for properties of finite groups. In *41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 537–546, 2000.
- [Wil10a] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *42nd ACM Symposium on Theory of Computing (STOC)*, pages 231–240, 2010.
- [Wil10b] Ryan Williams. Non-uniform ACC lower bounds. Manuscript, 2010.
- [Yao82] Andrew Yao. Theory and applications of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 80–91. IEEE, 1982.

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Struct. Algorithms*, 11(4):345–367, 1997.

## A A unitary in which all rows participate

There is a tension between the triple goals of (1) having many pairwise orthogonal vectors, (2) maintaining bounded pairwise intersections of the supports, and (3) having the supports large. It is natural to wonder whether the above construction (in which we found a number of vectors equal to  $1/2$  the dimension of the underlying space) is in some sense optimal. For example, is there some barrier to simultaneously optimizing all three goals?

Here we show that one can indeed optimize all three goals at the same time, by specifying a construction that builds on the “paired-lines” construction. Our construction will have as many pairwise orthogonal vectors as the dimension of the underlying space (which is obviously as many as is possible); it will have intersections sizes bounded above by 2 (the upper bound cannot be 0 without constraining the product of the number of rows and the support sizes to be at most the dimension of the underlying space, and no pairwise intersections can have cardinality one without violating orthogonality); the support sizes will be at least the square root of the dimension of the underlying space (and one can’t exceed that without having larger intersection sizes).

This construction is not needed for our main results, but we find it aesthetically pleasing that one can optimize all three parameters in this way. We *don’t* know of a local decomposition for this matrix, and we leave finding one as an intriguing open problem.

While the construction of §2.3.1 needed characteristic two, the present construction needs odd characteristic. We fix  $\mathbb{F}_q$  with  $q$  an odd prime power, and we choose a subset  $Q \subseteq \mathbb{F}_q^*$  of size  $(q-1)/2$  for which  $Q \cap -Q = \emptyset$ , where  $-Q = \{-x : x \in Q\}$ . Our vectors will have  $q^2 - 1$  coordinates, identified with the *punctured plane*  $P = \mathbb{F}_q \times \mathbb{F}_q \setminus \{(0, 0)\}$ .

We have three types of vectors in  $\{0, -1, +1\}^P$ : first, for all  $a \in \mathbb{F}_q$  and  $b \in Q$

$$v_{a,b}[x, y] = \begin{cases} +1 & x = 0, y = b \\ +1 & x \in Q, y = ax + b \\ -1 & x \in Q, y = ax - b \\ 0 & \text{otherwise} \end{cases}, \quad (2)$$

second, for all  $a \in \mathbb{F}_q$  and  $b \in -Q$

$$v_{a,b}[x, y] = \begin{cases} +1 & x = 0, y = b \\ +1 & x \in -Q, y = ax + b \\ -1 & x \in -Q, y = ax - b \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

and finally, for each  $c \in \mathbb{F}_q^*$

$$u_c[x, y] = \begin{cases} +1 & x = c, y \in \mathbb{F}_q \\ 0 & \text{otherwise} \end{cases}. \quad (4)$$

**Lemma A.1.** *The vectors defined in Eqs. (2), (3) and (4) are pairwise orthogonal and their supports form a  $(q, 2)$ -design.*

*Proof.* It is an easy computation to see that the support of each of the vectors has cardinality  $q$ . We now argue that they are pairwise orthogonal. There are several cases depending on the two rows under consideration:

1.  $v_{a,b}$  and  $v_{a',b'}$ : if one comes from Eq. (2) and the other from Eq. (3) then the supports are disjoint. So we assume both come from Eq. (2) or both come from Eq. (3).
  - (a) Both come from Eq. (2) and  $b = b'$ : we have one intersection  $(0, b)$  (which contributes  $+1$  to the inner product) and exactly one of the following two intersection points:  $(x = -2b/(a - a'), ax + b = a'x - b)$  or  $(x = 2b/(a - a'), ax - b = a'x + b)$ , which contributes  $-1$  to the inner product. We have exactly one because the two  $x$ -values are negations of each other, and non-zero, so exactly one is in  $Q$ .
  - (b) Both come from Eq. (2) and  $b \neq b'$ : we have exactly one of the following two intersection points:  $(x = (b' - b)/(a - a'), ax + b = a'x + b')$  or  $(x = (-b' + b)/(a - a'), ax - b = a'x - b')$ , which contributes  $+1$  to the inner product, and exactly one of the following two intersection points:  $(x = (b' + b)/(a - a'), ax - b = a'x + b')$  or  $(x = (-b' - b)/(a - a'), ax + b = a'x - b')$ , which contributes  $-1$  to the inner product. For each pair, there is exactly one of the pair of possible intersection points because the two  $x$ -values are negations of each other, and non-zero, so exactly one is in  $Q$ .
  - (c) Both come from Eq. (3) and  $b = b'$ : identical to case (1a) above, with  $-Q$  in place of  $Q$ .
  - (d) Both come from Eq. (3) and  $b \neq b'$ : identical to case (1b) above, with  $-Q$  in place of  $Q$ .
2.  $u_c$  and  $u_{c'}$ : these have disjoint supports for  $c \neq c'$ .
3.  $v_{a,b}$  and  $u_c$ : if  $c \in Q$ , then the support of  $u_c$  intersects the support of  $v_{a,b}$  only if  $v_{a,b}$  comes from Eq. (2), and then we get one intersection at point  $(x = c, ax + b)$  which contributes a  $+1$  to the inner product, and one intersection at point  $(x = c, ax - b)$  which contributes a  $-1$  to the inner product. If  $c \in -Q$ , then the support of  $u_c$  intersects the support of  $v_{a,b}$  only if  $v_{a,b}$  comes from Eq. (3), and we have an identical argument, with  $-Q$  in place of  $Q$ .

This is a complete enumeration of cases, and in no case did we have more than 2 intersection points.  $\square$

We conclude this section with a question: are these matrices related in some way to the DFT matrix over some family of non-abelian groups (e.g. the affine group  $\mathbb{F}_q^* \ltimes \mathbb{F}_q$ ), or are they indeed completely different from the unitaries seen before in quantum algorithms?

## B Converting a distributional oracle problem into a standard oracle

For completeness in this section we prove Corollary 2.12. A similar proof appears in [Aar10b].

**Corollary 2.12.** (Restated.) *Assuming Conjecture 2.6, there is an oracle  $O$  such that  $BQP^O \not\subseteq PH^O$ .*

*Proof.* Theorem 2.11 gives us two ensembles of random variables  $D_1 = \{D_{1,n}\}, D_2 = \{D_{2,n}\}$  over  $(N = 2^n)$ -bit strings for which BQLOGTIME can distinguish the two distributions but quasi-polynomial-size  $AC^0$  cannot. Then when  $D_1$  and  $D_2$  are viewed as distributions on (truth-tables of) *oracles*, there is a BQP oracle machine that distinguishes the two distributions, but no  $PH$  oracle machine can distinguish them. Specifically, we have that there exists a BQP oracle machine  $A$  for which

$$\Pr[A^{D_1}(1^n) = 1] - \Pr[A^{D_2}(1^n) = 1] \geq \epsilon$$

while for every  $PH$  oracle machine  $M$ ,

$$\Pr[M^{D_1}(1^n) = 1] - \Pr[M^{D_2}(1^n) = 1] \leq \delta < \epsilon,$$

(here we use standard techniques – see, e.g., [Hås87] – which show that on any fixed input, the output of the machine as a function of the oracle can be seen as a constant-depth quasi-polynomial-size  $AC^0$  circuit) and we have  $\epsilon > \delta$  for sufficiently large  $n \geq n_0$ .

We now convert the distributions on oracles into a single oracle  $O$  for which  $BQP^O \not\subseteq PH^O$ . Let  $L$  be a uniformly random unary language in  $\{1\}^*$ . For each  $n$ , if  $1^n \in L$ , sample a  $2^n$ -bit string  $x$  from  $D_1$  and define oracle  $O$  restricted to length  $n$  so that  $x$  is its truth table; otherwise sample a  $2^n$ -bit string  $x$  from  $D_2$  and define oracle  $O$  restricted to length  $n$  so that  $x$  is its truth table.

First, note that

$$\Pr[A^O(1^n) = L(1^n)] = (1/2) \cdot \Pr[A^{D_1}(1^n) = 1] + (1/2) \cdot \Pr[A^{D_2}(1^n) = 0] \geq 1/2 + \epsilon/2.$$

Now fix any  $PH$  machine  $M$ , and note that for sufficiently large  $n$ ,

$$\Pr[M^O(1^n) = L(1^n)] = (1/2) \cdot \Pr[M^{D_1}(1^n) = 1] + (1/2) \cdot \Pr[M^{D_2}(1^n) = 0] \leq 1/2 + \delta/2.$$

Consequently, since  $\epsilon > \delta$  there is a fixed choice for the oracle at length  $n$  such that  $A^O(1^n) \neq M^O(1^n)$ , for sufficiently large  $n$ .

Fix such a choice for the oracle, and consider another  $PH$  machine  $M'$ . By the same argument, we can find another sufficiently large input length  $n'$  where  $A^O(1^{n'}) \neq M'^O(1^{n'})$ .<sup>10</sup>

Continuing in this way, we obtain a single oracle such that for any  $PH$  machine  $M$  there exists some  $n$  for which  $A^O(1^n) \neq M^O(1^n)$ , concluding the proof.  $\square$

---

<sup>10</sup>We have assumed that our machines, on an input of length  $n$ , only query the oracle at inputs of length  $n$ . This can be ensured by working with input lengths that are sufficiently spread out (so that the machine cannot afford to formulate queries to the next largest length, and so that the oracle at shorter lengths can be hardcoded.)