THEORY OF COMPUTING, Volume 9 (26), 2013, pp. 809–843
www.theoryofcomputing.org

On Beating the Hybrid Argument*

Bill Fefferman† Ronen Shaltiel‡ Christopher Umans§
Emanuele Viola¶

Received February 23, 2012; Revised July 31, 2013; Published November 14, 2013

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: ε-distinguishability implies ε/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. Our main results are:

1. We give an instantiation of the Nisan-Wigderson generator (JCSS 1994) that can be broken by quantum computers, and that is $o(1)$-unpredictable against AC^0. We conjecture that this generator indeed fools AC^0. 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

*An extended abstract of this paper appeared in the Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS) 2012.

†Supported by IQI.

‡Supported by BSF grant 2010120, ISF grants 686/07,864/11 and ERC starting grant 279559.

§Supported by NSF CCF-0846991, CCF-1116111 and BSF grant 2010120.

¶Supported by NSF grant CCF-0845003.

ACM Classification: F.1.3, F.2.3

AMS Classification: 81P68, 68Q15

Key words and phrases: complexity theory, pseudorandom generator, BQP, hybrid argument, circuits, branching programs

 DOI: 10.4086/toc.2013.v009a026

against poly-logarithmic width (general) read-once oblivious branching programs. (This was also observed by other researchers.) Obtaining such generators where the output is indistinguishable from uniform is a longstanding open problem.

3. We identify a property of functions f, "resamplability," that allows us to beat the hybrid argument when arguing indistinguishability of

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

from uniform. This gives new pseudorandom generators for classes such as $\mathsf{AC}^0[p]$ with a stretch that, despite being sub-linear, is the largest known. We view this as a first step towards beating the hybrid argument in the analysis of the Nisan-Wigderson generator (which applies $G_f^{\otimes k}$ on correlated $x_1,\ldots,x_k$) and proving the conjecture in the first item.

1 Introduction

1.1 The hybrid argument

The *hybrid argument* [20] (see also [10, 48] and [15] for an exposition) is a powerful proof technique that has widespread applications in cryptography and complexity theory. Suppose we have a random variable $Z = (Z_1, Z_2, \ldots, Z_k) \in \{0,1\}^k$ that can be distinguished from the uniform distribution on k bits, U, by a function D, i. e.,

$$\left|\Pr[D(Z) = 1] - \Pr[D(U) = 1]\right| \geq \varepsilon\,.$$

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

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

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

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

which means that D distinguishes two adjacent hybrids, H_{i-1} and H_i, with gap at least ε/k. From here, it is easy to convert D into a closely related function that predicts Z_i from the prefix with advantage ε/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 [10, 48, 16, 23, 31, 33]. The hybrid argument also plays an important role in the inductive arguments underlying pseudorandom generators against space-bounded computation [32, 25].

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 is much less than $1/\varepsilon$ 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, $1/\varepsilon$. 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 [37]).

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."[1]

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 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 [33] that can be broken by quantum computers in the first setting, and we modify the standard analysis of the Impagliazzo-Nisan-Wigderson (INW) generator [25] in the second. These two settings are discussed in more detail in Section 1.2.

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 $\mathsf{AC}^0[p]$ 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.

1.2 Two consequences of beating the hybrid argument

We now describe two longstanding open problems in complexity theory 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]Barak, Shaltiel, and Wigderson [7] 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.

1.2.1 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 [9] in the early 1990's. Currently, oracles are known relative to which BQP is not in MA [45], 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 [2] 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.[2]

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 Section 2 for the formal definition). As in [2], 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 Section 2.6, such a distribution can be easily converted to a proper oracle O for which $\mathsf{BQP}^O \not\subseteq \mathsf{PH}^O$. To obtain such a distribution, we prove two main statements:

1. we generalize the setting of [2] to a simple framework in which any efficiently quantum-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 Section 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 Section 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.

1.2.2 Pseudorandom generators for branching programs of small width

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 $\mathsf{RL} = \mathsf{L}$, settling a major open problem in complexity theory. Existing constructions of pseudorandom generators [32, 25] fail to reach this goal because they use seeds of length $O(\log^2 n)$. Despite significant effort, no improvement in the

[2] Aaronson [2] 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 [1]. The viability of our approach is unaffected by this development.

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 [11, 29, 12].

In this work we show that the INW generator [25] 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. (This was also observed by other researchers.) Thus, bypassing the loss of the hybrid argument would yield a breakthrough in pseudorandom generators for small-width branching programs. In Section 3 we elaborate on this approach.

1.3 New pseudorandom generators by beating the hybrid argument

Following the seminal work of [31] 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 *ε-pseudorandom* for a class of circuits $\mathcal{C}$ if for every circuit C in $\mathcal{C}$,

$$\left|\Pr[C(Z)=1]-\Pr[C(U_t)=1]\right| \leq \varepsilon\,.$$

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

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 $f:\{0,1\}^n \to \{0,1\}$ be a function with *hardness* δ 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 δ-pseudorandom generator for $\mathcal{C}$. Its stretch (which we measure additively) is 1. One way to improve it is by *repeated sampling*, namely:

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

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.[3] 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 there is no gain from using the NW generator). Repeated sampling extends the seed by k bits, and by the previous discussion, using the hybrid argument k is bounded by $1/\delta$. In Section 4 we observe that this loss is inherent in black-box proofs.

Our main technical contribution in this direction is developing new proof techniques that break this barrier and allow us to show that $G_f^{\otimes k}$ is pseudorandom even for $k > 1/\delta$. As a consequence we obtain

[3] Note that not only does the hybrid argument loss yield no useful bound 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.

Seed length of generators fooling $\text{poly}(n)$**-size circuits on** n **bits.**		
Type of circuits	**Hybrid argument**	**This work**
$\mathsf{AC}^0[p]$, p prime	$n-n^{1/3}$	$n-n/\text{poly}\log n$ (Cor. 4.8,4.18)
AC^0 with $n^{o(1)}$ majority gates	$n-n^{1/3}$	$n-n^{\beta}, \forall\beta<1$ (Cor. 4.10)
$\mathsf{AC}^0[6]$	$n-\text{poly}\log n$ (under $\mathsf{L}\not\subseteq\mathsf{AC}^0[6]$)	$n-n^{\beta}, \forall\beta<1$ (Cor. 4.20) (under $\mathsf{L}\not\subseteq\mathsf{AC}^0[6]$)

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

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 functions and circuit classes we consider are:

Majority. The n-bit majority function has hardness $\tilde{O}(1/\sqrt{n})$ for AC^0 [22] (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 $G^{\otimes k}_{\text{majority}}$ does not decay with k: 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 Section 1.2.1.

Parity. The n-bit parity function is known to have hardness $\leq\tilde{O}(1/\sqrt{n})$ for the class $\mathsf{AC}^0[p]$ for every prime $p\neq 2$ [36, 38, 39]. Here $\mathsf{AC}^0[p]$ stands for 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 $G^{\otimes k}_{\text{parity}}$ 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 a similar result for the class of AC^0 circuits with few (a small polynomial number of) majority gates using the fact that parity is hard for this class [6, 8], and for $\mathsf{AC}^0[2]$ using the determinant function over GF(2) for a certain distribution M of input matrices due to Ishai and Kushilevitz [26, 27]. The determinant function also yields conditional results for the class $\mathsf{ACC}^0 := \cup_m \mathsf{AC}^0[m]$. A recent breakthrough by Williams [47, 46] shows that $\text{NEXP}\not\subseteq\mathsf{AC}^0[m]$, but does not seem to imply pseudorandom generators. Here our contribution is to show that, under a hardness assumption, one can get generators with n output bits and seed length $n-n^{\Omega(1)}$, whereas previous techniques would only give $n-\text{poly}\lg n$.

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.

1.4 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 Section 1.3, 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 (fixed) input (x,b) produces a distribution $R(x)$ over pairs (x',b') such that x' is uniformly distributed over the domain of f, and the event $\{b = f(x) \Leftrightarrow b' = f(x')\}$ holds with probability one. We stress that we are interested in procedures R that use less resources than those required to compute f. This notion of resamplability is thus a special type of random self-reducibility, a well-studied concept in complexity theory (see, e. g., [13] 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 δ-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 Section 4 are obtained by identifying hard functions that have (known) efficient resamplers, and then applying arguments that rely on them being resamplable. We comment that in Section 4 we use more general notions of resamplability than the one described here (and some of our results make use of these generalizations).

1.5 Organization of this paper

In Section 2 we state our conjecture about a certain instantiation of the Nisan-Wigderson generator fooling AC^0, and we prove that this conjecture would yield an oracle separating BQP from PH. A construction that is related to this proof, but not needed for it, is presented in Appendix A. In Section 3 we discuss our results about pseudorandom generators for space-bounded computation. In Section 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.

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 form an orthonormal basis, and the same holds for the columns. We name the standard basis vectors of the $N = 2^n$-dimensional vector space 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. A local unitary can be applied in a single step of a quantum computer. A *local decomposition* of a unitary matrix is a factorization of the matrix into a product of local unitaries. We say an $N \times N$ unitary is *efficiently quantum computable* if this factorization has at most $\mathrm{poly}(n)$ 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 quantum-computable unitary matrix can be realized (i. e., approximated), up to exponentially small error, by a $\mathrm{poly}(n)$-size quantum circuit [28] over the chosen universal gate set.

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 $\mathrm{poly}\log(N)$ 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. The overall map takes $|i\rangle|0\rangle$ to $(-1)^{x_i}|i\rangle|0\rangle$. 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 needed in Section 2.4.2. It states (essentially) that a block-diagonal matrix, all of whose blocks are efficiently quantum computable, is itself efficiently quantum 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 $U_1, U_2, \ldots, U_M$, where each U_i is a $N/M \times N/M$ matrix that has a* $\mathrm{poly}(n)$*-size quantum circuit, a description of which is generated by a uniform* $\mathrm{poly}(n)$ *time procedure, given input i. Then given three registers of m qubits, $n-m$ qubits, and* $\mathrm{poly}(n)$ *qubits, respectively, with the third register initialized to $|000\cdots 0\rangle$, there is a* $\mathrm{poly}(n)$ *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 $\mathrm{poly}(n)$ cycles through this list: when we are on step $(a_1, a_2, \ldots, a_b) \in [n]^b$ in one of these cycles, we operate on qubits $a_1, a_2, \ldots, a_b$. 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

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

with each $j_\ell^{(i)} \in [d]$ specifying which gate to apply at step ℓ 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 ℓ). Let $f : [M] \to [d]^{n^k}$ 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 $\ell = 1, 2, 3, \ldots, n^k$: apply the "controlled unitary" that consults the ℓ-th component of the third register, and applies the specified gate to qubits $(a_1, a_2, \ldots, a_b)$ of the second register (again, $(a_1, a_2, \ldots, a_b)$ are easily determined from ℓ 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 quantum-computable unitary into a distribution that can be distinguished from uniform by a BQLOGTIME machine. Our framework generalizes the setup in [2].

Let A be any $N \times N$ matrix with entries[4] 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.

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 $z_i = \operatorname{sgn}((Ax)_i) = \operatorname{sgn}((\overline{A}x)_i)$ for $i \le M$ 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 $\varepsilon > 0$ for which*

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

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 $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} |i\rangle$. Multiply x into the phase to obtain $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} x_i |i\rangle$.

[4] We could extend this framework to matrices with general entries, but we choose to present this restriction since it is all we need.

2. Apply $\overline{A}$ to obtain $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} (\overline{A}x)_i |i\rangle$.

3. Multiply z into the phase to obtain $\frac{1}{\sqrt{N}} \sum_{i \in \{0,1\}^n} z_i (\overline{A}x)_i |i\rangle$.

4. Define vector w by
$$w_i = \frac{1}{\sqrt{N}} z_i (\overline{A}x)_i \,.$$
Apply the $N \times N$ Hadamard[5] 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 $i \le M$,
$$w_i = \frac{1}{\sqrt{N}} \left| (\overline{A}x)_i \right| \quad \text{and hence} \quad E[w_i] = \frac{1}{\sqrt{N \cdot |S(A,i)|}} \Omega(\sqrt{|S(A,i)|}) = \Omega(1/\sqrt{N})$$
(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 $E[\sum_i w_i] = M \cdot \Omega(1/\sqrt{N}) = \Omega(\sqrt{N})$, 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 ε (and in such cases the acceptance probability is at least ε^2). Overall, the acceptance probability is $\Omega(1)$. □

The BQLOGTIME algorithm for what Aaronson calls FOURIER CHECKING in [2] is recovered from the above framework by taking A to be a DFT matrix (and $M = N$).

2.3 Unitary matrix 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 quantum-computable unitary. We start by reviewing the Nisan–Wigderson pseudorandom generator.

2.4 The Nisan–Wigderson generator

Definition 2.4 ([33])**.** A set family $\mathcal{D} = \{S_1, S_2, \ldots, S_m\}$ is an (ℓ, p) design if every set in the family has cardinality ℓ, and for all $i \ne j$, $|S_i \cap S_j| \le p$.

[5] This is the matrix H whose rows and columns are indexed by $\{0,1\}^n$, with entry (i,j) equal to $-1^{\langle i,j \rangle}/\sqrt{N}$.

Definition 2.5 ([33]). Given a function $f:\{0,1\}^\ell \to \{0,1\}$ and an (ℓ,p) design $\mathcal{D}=\{S_1,S_2,\dots,S_m\}$ in a universe of size t, the function $NW_{\mathcal{D}}^f:\{0,1\}^t \to \{0,1\}^m$ is given by

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

where each f_i is the function f with a fixed set of its inputs negated,[6] 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 quantum 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 ℓ 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.

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

$$\left|\Pr[C(U_{t+m})=1] - \Pr[C(U_t, NW_{\mathcal{D}}^{\text{MAJORITY}}(U_t))=1]\right| \le o(1)\,.$$

Using the hybrid argument, a distinguishing circuit C with gap ε can be converted to a predictor with advantage ε/m and then (via the standard arguments in [31, 33]) into a slightly larger circuit that computes MAJORITY with success rate $1/2+\varepsilon/m$. Thus the above statement is true for $m \le o(\sqrt{\ell})$; if the $1/m$ loss from the hybrid argument can be avoided (or reduced), it would be true for m as large as $\mathrm{poly}(\ell)$ (and even larger) as we conjecture is true.

2.4.1 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 $\phi: B_1 \to B_2$ 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 $\mathbb{F}_q \times \mathbb{F}_q$:

$$v_{a,b}[x,y] = \begin{cases} -1 & \text{if } y = ax+b\,, \\ +1 & \text{if } y = ax+\phi(b)\,. \end{cases} \tag{2.1}$$

[6]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.

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 equation* (2.1) *are pairwise orthogonal, and their supports form a* $(2q,4)$ *design.*

Proof of Lemma 2.7. 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. □

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.

2.4.2 A local decomposition

We new describe a $q^2 \times q^2$ unitary matrix that is efficiently quantum computable and has the (normalized) vectors $v(a,b)$ from equation (2.1) 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 quantum 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,

$$\frac{1}{\sqrt{8}}(I_{q/8}|-I_{q/8}|I_{q/8}|-I_{q/8}|I_{q/8}|-I_{q/8}|I_{q/8}|-I_{q/8})$$

as its next $q/8$ rows, etc., and whose last row is $\frac{1}{\sqrt{N}}(1,1,1,\dots,1)$.

Proof of Proposition 2.8. The DFT matrices are well-known to be efficiently quantum 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 quantum 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 α 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*

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

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

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

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 quantum computable.

Proof of Lemma 2.9. Consider the $q^2 \times q^2$ block-diagonal matrix that has as its (k,k) block the matrix whose (i,j) entry is $(-1)^{\operatorname{Tr}(ij\alpha^k)}$ 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 matrix F is efficiently quantum computable and the map $j \mapsto j\alpha^k$ is classically and reversibly (and thus quantum) efficiently computable. Thus each $q \times q$ block on the diagonal is efficiently quantum computable. By Lemma 2.2 the entire matrix is efficiently quantum 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 quantum computable, has the vectors $v(a,b)$ from equation (2.1) as $q^2/2$ of its rows.*[7]

[7] To be precise, these are the $v(a,b)$ with respect to *some* equipartition B_1, B_2 and *some* bijection ϕ.

Proof of Theorem 2.10. 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

$$FD_cF^{-1} = \frac{1}{\sqrt{q}}S_c - \frac{\sqrt{q}-1}{q}J, \quad \text{and that} \quad FD'_cF^{-1} = \frac{1}{\sqrt{q}}S_c - \frac{1}{\sqrt{q}}J.$$

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 ϕ 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 equation (2.1) as the "top-halves" of each successive set of q rows of the large matrix. □

2.5 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 $\mathsf{BQLOGTIME}$ algorithm that distinguishes D_N from the uniform distribution on N bits with probability $\geq \Omega(1)$; and (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 of Theorem 2.11. 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 $\mathsf{BQLOGTIME}$ 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. □

Section 2.6 below describes how to convert a "distributional" oracle as obtained in Theorem 2.11 into a standard oracle; the next corollary then follows.

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

2.6 Distributional vs. standard oracles

For completeness we include this argument; a similar proof[8] appears in [2].

Let $D_1 = \{D_{1,n}\}, D_2 = \{D_{2,n}\}$ be ensembles of random variables over $2^{g(n)}$-bit strings (and assume $g(n) \le \mathrm{poly}(n)$ is injective and easily computable) for which $\mathsf{BQLOGTIME}$ can distinguish the two distributions but 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

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

while for every PH oracle machine M,

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

and we have $\varepsilon > \delta$ for sufficiently large $n \ge n_0$.

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

We will show that *conditioned on* $A^O(1^n) = L(1^n)$ *for all* $n \ge n_0$, we still have $L \notin \mathsf{PH}^O$ with probability 1 over the choice of L and O. Let $T(n)$ be the event that $A^O(1^n) = L(1^n)$, and for each PH machine M and let $S_M(n)$ be the event that $M^O(1^n) = L(1^n)$. Note that $T(n), S_M(n)$ are each independent of $T(n'), S_M(n')$ for $n' \ne n$. Then we have for $n \ge n_0$:

$$\Pr[T(n)] = (1/2)\cdot\Pr[A^{D_1}(1^n)=1] + (1/2)\cdot\Pr[A^{D_2}(1^n)=0] \ge 1/2+\varepsilon/2$$

and

$$\Pr[S_M(n)] = (1/2)\cdot\Pr[M^{D_1}(1^n)=1] + (1/2)\cdot\Pr[M^{D_2}(1^n)=0] \le 1/2+\delta/2$$

and thus

$$\Pr_{L,O}[S_M(n)|T(n)] = \frac{\Pr[S_M(n)\wedge T(n)]}{\Pr[T(n)]} \le \frac{\Pr[S_M(n)]}{\Pr[T(n)]} \le \frac{1+\delta}{1+\varepsilon} < 1\,.$$

So by independence of different input lengths:

$$\Pr_{L,O}\left[S_M(n_0)\wedge S_M(n_0+1)\wedge S_M(n_0+2)\wedge\cdots \,\middle|\, T(n_0)\wedge T(n_0+1)\wedge T(n_0+2)\wedge\cdots\right] = 0\,.$$

The number of possible PH machines is countably infinite, so by a union bound,

$$\Pr_{L,O}\left[\exists M\;\; S_M(n_0)\wedge S_M(n_0+1)\wedge S_M(n_0+2)\wedge\cdots \,\middle|\, T(n_0)\wedge T(n_0+1)\wedge T(n_0+2)\wedge\cdots\right] = 0\,.$$

So conditioned on $A^O(1^n) = L(1^n)$ for all $n \ge n_0$, we have $L \notin \mathsf{PH}^O$ with probability 1 over the choice of L and O. Thus (by hardwiring $L(n)$ for $n < n_0$ in the BQP machine), there exists an oracle O for which $\mathsf{BQP}^O \not\subset \mathsf{PH}^O$.

[8] Our proof differs in one respect: the conditioning on $T(n)$, which allows us to handle any pair of ε, δ with some separation.

3 Toward pseudorandom generators with short seed for small space

In this section we prove that the pseudorandom generator construction of [25] with seed length $O(\log n \log\log n)$ yields distributions that are unpredictable by $\operatorname{poly}\log n$-width branching programs. This result was also observed by other researchers. However, to the best of our knowledge, it has not been published before. We start by reviewing the construction [25] as presented in [35].

3.1 The Impagliazzo–Nisan–Wigderson pseudorandom generator

We require the standard notions of "width S read-once oblivious branching programs" (abbreviated ROBPs) and "(k,η)-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 node at the end of the path.

(k,η)-extractors. These were introduced in [34] and are functions $E:\{0,1\}^r\times\{0,1\}^d\to\{0,1\}^m$ with the property that for every distribution X over $\{0,1\}^r$ that is uniform over a set of size $\ge 2^k$, the distribution $E(X,U_d)$ is η-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 $C\ge 1$ is a constant to be determined later. The construction will rely on $(r-s-\log(1/\eta),\eta)$-extractors $E:\{0,1\}^r\times\{0,1\}^d\to\{0,1\}^r$ for $r\ge r_0$ and $d=O(s+\log(1/\eta))$. We stress that there are explicit constructions with these parameters for a sufficiently large universal constant C [17], and that the dependence of d on both s and η is optimal up to constant factors.

Definition 3.1. For $0\le j\le\log n$ we define functions $G_j:\{0,1\}^{r_0+jd}\to\{0,1\}^{2^j}$ 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 [25] shows that for every j, if G_j is a pseudorandom generator with error ε_j then G_{j+1} is a pseudorandom generator with error $2\varepsilon_j+\eta$. 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 [11, 29, 12] 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. As explained in Section 3.2 below, the error loss in this connection may be seen as arising from the hybrid argument, so in this sense these works are "beating the hybrid argument." However, [12] 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 η 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 $\tilde{O}(\log n)$.

3.2 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 δ. This is stated in the theorem below.

Theorem 3.2. *Fix $\eta > 0$ and let $Z = (Z_1, \dots, Z_n)$ denote the output distribution of G in Definition 3.1 above on a uniformly chosen seed. Then, for every width $S = 2^s$ ROBP P and every $0 \le i \le n$ we have*

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

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

The high level idea of the proof is to show that if G_j is unpredictable with advantage δ 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 in the proof [25]. 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 3.2. Let P be a width $S = 2^s$ ROBP. We say that P predicts G_j with advantage δ if there exists an i such that $\Pr[P(Z_1, \dots, Z_{i-1}) = Z_i] > 1/2 + \delta$ where $Z_1, \dots, Z_n$ are sampled by applying G_j on a uniformly chosen seed. We show that:

Claim 3.3. *For $j > 1$ if P predicts G_j with advantage δ then there exists a width S ROBP P' that predicts G_{j-1} with advantage $\delta - 2\eta$.*

Theorem 3.2 follows from Claim 3.3 by noting that if P predicts $G = G_{\log n}$ with advantage δ then by iteratively applying Claim 3.3 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 3.3. 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

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

Note that:

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

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

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

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 $\ge 2^{r-(s-\log(1/\eta))}$ which by the properties of extractors gives that $E(X,Y)$ is η-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 η-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. □

Following Theorem 3.2, 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 3.2 instantiated with $\eta = 1/\log^{\Theta(1)} n$ and let $E' : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^m$ be a $(k, 1/n)$-extractor for $k, m = n^{\Theta(1)}$ [49, 21]; 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 [7] for a study on using extractors to produce pseudorandom distributions.) This approach is inspired by a pseudorandom generator construction of [40] in the setup of small circuits. More precisely, [40] instantiate the NW generator with a function that is only mildly hard on average giving a distribution which is unpredictable, but for δ 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.[9]

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.

4.1 Breaking the repeated sampling generator

For a distribution D, we denote by $D^{\otimes k}$ the concatenation of k independent samples of D.

Fact 4.1. *There is $c > 0$ such that for any n and $\varepsilon \geq 1/2^{n/c}$ such that $\log 1/\varepsilon$ is an integer: there exists a (non-explicit) function $f : \{0,1\}^n \to \{0,1\}$ such that (1) for any circuit C of size $s \leq 2^{n/c}$,*

$$\Pr_{x \in \{0,1\}^n}[C(x) \neq f(x)] \geq 1/2 - \varepsilon\,,$$

and (2) there is a $\operatorname{poly}(n/\varepsilon)$-size DNF distinguishing $(X, f(X))^{\otimes c/\varepsilon}$ from uniform with probability ≥ 0.9.

Proof. Let $x = (y,z)$ where $|y| = \log(1/\varepsilon) + 1$, and $|z| = n - |y| \geq n/2$ (for c large enough). Let $h : \{0,1\}^{|z|} \to \{0,1\}$ be a function such that for a universal constant d, any circuit D of size $\leq 2^{n/d}$ satisfies

$$\Pr_{x \in \{0,1\}^n}[D(x) \neq f(x)] \geq \frac{1}{2} - \frac{1}{2^{n/d}}\,.$$

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 $\Pr_{x \in \{0,1\}^n}[C(x) = f(x)] \geq 1/2 + \varepsilon$ then

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

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,

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

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

[9] For context, we remark that this result in [40] is not known to hold for restricted circuit classes such as $\mathsf{AC}^0[p]$. The specific proof in [40] fails because at its heart lies hardness amplification (specifically the hard-core set lemma [24]) which in these restricted classes is either not known to hold or false [37, 30].

To see (2), consider the distributions

$$\otimes_{i \le c/\varepsilon}(y^i, z^i, f(y^i, z^i)) \tag{$\star$}$$

and

$$\otimes_{i \le c/\varepsilon}(y^i, z^i, b^i) \tag{U}$$

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

In either distribution, we expect $c/\varepsilon \cdot \varepsilon/2 = c/2$ 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. □

4.2 Using resamplability

In this section we identify a property of functions that allows us to avoid the loss in the hybrid argument. We call this property *resamplability*. For exposition, it is convenient to work with "problems" rather than functions. So we start by discussing the connection between functions and problems. There are two natural ways to define a problem given a function. The first corresponds to the notion of resamplability described in the introduction:

Definition 4.2 ($\Pi(f)$). Let $f : \{0,1\}^n \to \{0,1\}$ be a function. We define the problem $\Pi(f)$ as

$$\Pi(f)_Y := \{(x, f(x)) : x \in \{0,1\}^n\} ,$$
$$\Pi(f)_N := \{(x, 1 - f(x)) : x \in \{0,1\}^n\} .$$

The second way is:

Definition 4.3 ($\Pi'(f)$). Let $f : \{0,1\}^n \to \{0,1\}$ be a function. We define the problem $\Pi'(f)$ as

$$\Pi'(f)_Y := \{x \in \{0,1\}^n : f(x) = 1\} ,$$
$$\Pi'(f)_N := \{x \in \{0,1\}^n : f(x) = 0\} .$$

For most of our results one can work with either $\Pi(f)$ or $\Pi'(f)$; our default is to work with $\Pi(f)$. However we only know how to obtain the result in Section 4.5 working with $\Pi'(f)$.

We now define resamplability.

Definition 4.4. A problem $\Pi = \Pi_Y \bigcup \Pi_N$ 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 Π_Y (resp., Π_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.

Lemma 4.5. *Suppose a problem $\Pi = \Pi_Y \bigcup \Pi_N$ ($\Pi_Y \bigcap \Pi_N = \emptyset$) has a resampler $R_r(\cdot)$. If a function C distinguishes k independent samples of Π_Y from k independent samples of Π with probability ε, i. e.,*

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

then there is a function function C' of the form $C'(x) := C(\bar{R}^1(x), \ldots, \bar{R}^k(x))$ 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 Π_Y from Π_N with the same probability ε, i. e.,

$$\left|\Pr[C'(\Pi_Y) = 1] - \Pr[C'(\Pi_N) = 1]\right| \geq \varepsilon\,.$$

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 Π_Y from Π_N; the innovation in the proof below is that it is able to replace $\Pi_N^{\otimes k}$ with $\Pi^{\otimes k}$.

Proof of Lemma 4.5. Let $B^1, \ldots, B^k \in \{Y, N\}$ be independent bits coming up Y with probability $|\Pi_Y|/|\Pi|$. Note that the distribution $\Pi^1, \Pi^2, \ldots, \Pi^k$ equals the distribution $\Pi^1_{B^1}, \Pi^2_{B^2}, \ldots, \Pi^k_{B^k}$. By averaging, there exists a way to fix each variable B^i to a value b^i such that

$$\left|\Pr[C\left(\Pi_Y^1, \Pi_Y^2, \ldots, \Pi_Y^k\right) = 1] - \Pr[C\left(\Pi^1_{b^1}, \Pi^2_{b^2}, \ldots, \Pi^k_{b^k}\right) = 1]\right| \geq \varepsilon\,.$$

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 $F(x) := (R^1(x), \ldots, R^k(x))$ where $R^i(x)$ is a uniform element of Π_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

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

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

To demonstrate the usefulness of Lemma 4.5 let us elaborate on its consequences for the promise problem $\Pi(f)$ defined above: For $\Pi = \Pi(f)$ we get that $\Pi_Y^{\otimes k}$ is the $k \cdot (n+1)$ bit long output of the repeated sampling generator $G_f^{\otimes k}$ while $\Pi^{\otimes k}$ is the uniform distribution on $k \cdot (n+1)$ bit strings. Consequently, Lemma 4.5 establishes the pseudorandomness of $G_f^{\otimes k}$ assuming f is hard on average (with no quantitative loss in the distinguishing parameter).

In the next sections we discuss cases in which resamplability yields new results. We start with the simplest setting, that of the parity function. Then we move to majority, which requires some extra tricks. Then we discuss the use of a problem introduced by Ishai and Kushilevitz [26, 27]. We also mention a possible alternative approach to get some of our generators. Finally we observe that resamplability implies a worst-case vs. average-case connection.

4.3 Generators based on parity

First we note the efficient resamplability of parity.

Fact 4.6. *The problem* $\Pi(parity)$ *is resamplable in (*poly-*size)* NC^0.

Proof. The resampler $R_r(x,b)$ uses the first bit of r to select a bit c and the remaining bits to select a string y of length $|x|$ with parity c. It outputs the pair $(x \oplus y, b \oplus c)$. For fixed r, this amounts to complementing some input bits, which can be done in NC^0. □

Combining this fact with Lemma 4.5 we obtain new pseudorandom distributions for low-level circuit classes. We start with the class of AC^0 circuits with mod p gates—denoted $\mathsf{AC}^0[p]$, for an odd prime p. The strongest known hardness result for this class is the following well-known result by Smolensky [38, 39] (cf. [14]).

Lemma 4.7 ([39])**.** *For every d and prime $p > 2$, there is a constant $\alpha > 0$ such that the n-bit parity function is ε-hard, with $\varepsilon = n^{-1/2+o(1)}$, for $AC^0[p]$ circuits of size $\leq 2^{n^\alpha}$.*

Equivalently if X is a random variable uniformly distributed on $\{0,1\}^n$, then $(X, \mathrm{parity}(X))$ is ε-pseudorandom for such circuits. (Cf. Section 1.3 for the definition of hard and pseudorandom.) The following corollary shows that this pseudorandomness does not decay with the number of repeated experiments.

Corollary 4.8. *Fix a prime $p \neq 2$ and $d \geq 1$. For every $k \leq 2^{n^{o(1)}}$, every* $\mathrm{poly}(n,k)$*-size $AC^0[p]$ circuit C of depth d satisfies*

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

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 $G : \{0,1\}^{n(1-1/\mathrm{poly}\lg n)} \to \{0,1\}^n$ that is $o(1)$-pseudorandom for $AC^0[p]$ circuits C of depth d and size $2^{\lg^d n}$.

Proof. This proof follows from the combination of Lemma 4.5, Fact 4.6, and Smolensky's Lemma 4.7. □

For the interesting case of $p = 2$, this proof does not work. In Section 4.5 we obtain similar generators using the machinery of [27].

The distribution induced by the generator in Corollary 4.8 has the appealing feature that it can be equivalently generated by an 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 [44].

We now consider the class 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 [41]. 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 [8, Corollary 4.4] building on the seminal lower bound by Aspnes, Beigel, Furst, and Rudich [6].

Lemma 4.9 ([6, 8]). *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^α majority gates,*

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

Actually [8, Corollary 4.4] has $1/4$ instead of $o(1)$, but the same techniques give $o(1)$.

Combining Lemmas 4.9, 4.6, and 4.5, 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 4.10. *For every $d \geq 1, \delta \in (0,1)$ there is $\varepsilon > 0$ such that for large enough n there are explicit generators $G: \{0,1\}^{n(1-1/n^\delta)} \to \{0,1\}^n$ such that for any And-Or-Majority-Not circuit C of depth d, size $\leq 2^{n^\varepsilon}$, with $\leq n^\varepsilon$ majority gates,*

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

4.4 A generator based on majority

We begin by remarking that we do *not* know of a resampler for $\Pi(\text{majority})$ (nor $\Pi'(\text{majority})$), so this setup is a bit more complicated. We require a generalization of Definition 4.4 and Lemma 4.5, in which the "resampler" $R_r(\cdot)$ maps a *source* problem $W_Y \bigcup W_N$ to a *target* problem $\Pi_Y \bigcup \Pi_N$. We furthermore relax the requirement on the output of the resampler $R_r(\cdot)$, and allow the output distribution to be only η-close to the target distribution (for some parameter $\eta > 0$).

More precisely, we consider a revised notion of Definition 4.4 in which part (1) of the definition is replaced with: for any $x \in W_Y$ (resp. W_N), the distribution $R_r(x)$ for uniform r is η-close to the uniform distribution over Π_Y (resp. Π_N). The argument of Lemma 4.5 can be used in exactly the same way to reduce an ε-distinguisher for $\Pi_Y^{\otimes k}$ vs. $\Pi^{\otimes k}$ to an $(\varepsilon - k \cdot \eta)$-distinguisher for W_Y vs. W_N. In the application below, $\eta = \exp(-n^{\Omega(1)})$ is very small so that the loss of $k \cdot \eta$ is insignificant for polynomial k.

We will now implement this plan for the majority function. Our target problem is $\Pi = \Pi(\text{majority})$ on odd n. Namely, for odd n, let

$$\Pi_Y = \{(y, \text{majority}(y)) : y \in \{0,1\}^n\} \quad \text{and} \quad \Pi_N = \{(y, 1 - \text{majority}(y)) : y \in \{0,1\}^n\}\,.$$

This is done so that $\Pi_Y^{\otimes k}$ is the output of the repeated sampling generator and $\Pi^{\otimes k}$ is the uniform distribution on strings of length $k \cdot (n+1)$.

Our source problem is $W = W_Y \bigcup W_N$ defined as follows: for odd ℓ, W_Y is the set of ℓ-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 4.11 ([22]). *For any constants $d \geq 1, \varepsilon > 0$, $\text{poly}(\ell)$-size AC^0 circuits of depth d cannot distinguish W_Y from W_N with gap greater than ε.*

Only a worst-case lower bound is stated in [22], but the stated average-case result follows using standard techniques [2, 37].[10] The next step is to show a resampler from W to Π.

Lemma 4.12. *There is a function $t = \mathrm{poly}(\ell)$ and a distribution $R_r(\cdot)$ on AC^0 circuits of size $\mathrm{poly}(\ell)$ mapping ℓ 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 Π_Y, and*
- *for any $x \in W_N$, $R_r(x)$ has statistical distance $\exp(-n^{\Omega(1)})$ from uniform in Π_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 4.13. *For any constant $d \geq 1$ and any function $k = \mathrm{poly}(n)$, every $\mathrm{poly}(n)$-size AC^0 circuit C of depth d satisfies*

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

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 4.13 follows by combining Lemma 4.11 with the version of Lemma 4.5 discussed above instantiated with the resampler of Lemma 4.12. We now present the resampler required for Lemma 4.12.

Proof of Lemma 4.12. Let $t = t(\ell)$ be odd. Let D_t be a probability distribution over $\{0,\ldots,t\}$ that we specify later. On input $x \in \{0,1\}^\ell$ the resampler R will use the randomness r to do the following: Pick i at random according to the distribution D_t. Concatenate $2i+1$ copies of x with a balanced string on $(t-(2i+1))\ell$ bits, and let y denote a random permutation of the obtained string. Note that if $x \in W_Y$, the hamming weight of y is $(n+1)/2+i$ and $\mathrm{majority}(y) = 1$, while if $x \in W_N$, the hamming weight of y is $(n-1)/2-i$ and $\mathrm{majority}(y) = 0$. The final output of the resampler is obtained by flipping a coin and outputting $(y,1)$ or $(y \oplus 1^n, 0)$ depending on the coin flip. Note that for every setting of random coins r of the resampler we indeed have that (1) if $x \in W_y$ then $R_r(x) \in \Pi_Y$, and (2) if $x \in W_n$ then $R_r(x) \in \Pi_N$.

For every $x \in W_Y \cup W_N$ the distribution of the first argument of $R_r(x)$ is the same (and does not depend on x). Let us denote this random variable by z. To conclude the proof it is sufficient to choose a distribution D_t over i so that z is $\exp(-n^{\Omega(1)})$-close to uniform.

We now describe the distribution D_t for choosing i. We select $i \in \{0,\ldots,t\}$ with the probability given by the uniform distribution to strings of weight $(n+1)/2+i$, normalized to give a probability distribution. Since $n = \ell t$, by letting t be a sufficiently large polynomial in ℓ and using a Chernoff bound, the statistical distance between z and the uniform distribution is $\exp(-n^{\Omega(1)})$. □

[10]Specifically, one can use the fact that the problem is resamplable (just permute input bits) and the fact that approximate majority is in AC^0 [3, 4] (cf. [42]) to show that any small AC^0 circuit distinguishing W_Y from W_N with gap $\varepsilon \geq \Omega(1)$ can be transformed into a small AC^0 circuit solving W in the worst case.

4.5 Generators based on L-hardness

We consider the problem, introduced by Ishai and Kushilevitz [26, 27], of distinguishing certain matrices with full rank from rank full -1. This problem is used to great effect in several works, e. g., [5, 18, 19], and we use the name CMD (for connectivity matrix determinant) from [18]. For a self-contained exposition of this problem and the properties we shall need, see [43, Chapter 4].

Definition 4.14 ([26, 27])**.** An input to the problem CMD 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 $n(n+1)/2$ entries on and above the main diagonal. Each such matrix has rank $\geq n-1$. CMD_Y are matrices with full rank n over GF(2), CMD_N are matrices with rank $n-1$.

Note that the above definition corresponds to $\Pi'(f)$ where f is the boolean function that given a matrix as above outputs 1 iff the matrix has full rank, cf. Definition 4.3.

In [27] various useful properties are established. First, note CMD is balanced, i. e., $|\mathrm{CMD}_Y| = |\mathrm{CMD}_N|$. 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 NC^0 reductions, i. e., maps such that each output bits depends on just a constant number of input bits.

Lemma 4.15 ([27])**.** CMD *is $\oplus L$-complete under NC^0 reductions.*

Finally, the techniques in [27] also show that CMD is resamplable in $\mathsf{AC}^0[2]$.

Lemma 4.16 ([27])**.** CMD *is resamplable in* poly-*size* $\mathsf{AC}^0[2]$.

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

We use another result by Smolensky, that majority is hard for $\mathsf{AC}^0[2]$. See [14] for an exposition.

Lemma 4.17 ([39])**.** *For any $\mathsf{AC}^0[2]$ circuit C of size s and depth d we have*

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

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

Corollary 4.18. *For every d there is c such that for large enough n there is an explicit generator $G : \{0,1\}^{n(1-1/\lg^c n)} \to \{0,1\}^n$ such for any $\mathsf{AC}^0[2]$ of depth d and size $2^{\lg^d n}$:*

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

Towards the proof of the corollary we record the following standard fact.

Fact 4.19. *Let $f : \{0,1\}^n \to \{0,1\}$ be a balanced function, and let $C : \{0,1\}^n \to \{0,1\}$ be any function. Then*

$$\Pr_x[C(x) = f(x)] = \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr_{x:f(x)=1}[C(x) = 1] - \Pr_{x:f(x)=0}[C(x) = 1] \right).$$

Proof of Corollary 4.18. By Lemma 4.17 and Fact 4.19, circuits of the given resources satisfy

$$\left| \Pr_{\substack{x \in \{0,1\}^{\lg^a n} \\ \text{majority}(x)=1}} [C(x) = 1] - \Pr_{\substack{x \in \{0,1\}^{\lg^a n} \\ \text{majority}(x)=0}} [C(x) = 1] \right| \le o(1), \qquad (\star)$$

where the probability is over 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 NC^0 reductions (Lemma 4.15). In addition, CMD is resamplable (Lemma 4.16). The combination of these facts implies that circuits of the given resources satisfy

$$\left| \Pr[C(\text{CMD}_Y) = 1] - \Pr[C(\text{CMD}_N) = 1] \right| \le o(1),$$

where the probability is over inputs of length $\lg^b n$, for a constant b depending only on p, d. This holds because if some circuit C violates the above, on input a majority instance we can apply the reduction to CMD, and then the resampler, to violate equation $(\star)$.

The output of the generator G is a k-tuple of strings representing CMD_Y instances. Recall that 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 4.5. □

Note that the proof in Section 4.4 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 $\mathsf{AC}^0[2]$ circuits for every $k = \text{poly}(n)$.

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

Corollary 4.20. *Suppose that $L \not\subseteq \mathsf{AC}^0[6]$. Then for every $d > 1$ and any $\delta \in (0,1)$, for large enough n there is an explicit generator $G : \{0,1\}^{n(1-1/n^\delta)} \to \{0,1\}^n$ such for any $\mathsf{AC}^0[6]$ circuit of depth d and size n^d:*

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

Proof. Since CMD is hard for L, the assumption implies that $\text{poly}(n)$-size $\mathsf{AC}^0[6]$ circuits fail to compute CMD on instances of length n^δ. Since CMD is resamplable (Lemma 4.16) we can apply Proposition 4.21 to argue that $\text{poly}(n)$-size $\mathsf{AC}^0[6]$ circuits C satisfy

$$|\Pr[C(\text{CMD}_Y) = 1] - \Pr[C(\text{CMD}_N) = 1]| \le o(1),$$

where the probabilities are over instances of length n^δ.

The output of the generator G is a k-tuple of strings representing CMD_Y instances of length n^δ. Recall that CMD is balanced, i. e., $|\text{CMD}_Y| = |\text{CMD}_N|$, so the seed length is $(n^\delta - 1)n/n^\delta = n(1 - 1/n^\delta)$. The correctness follows from Lemma 4.5. □

4.6 On a possible alternative way to get generators for $\mathsf{AC}^0[p]$

In this section we sketch a possible alternative approach to get generators with seed length $n(1-1/\operatorname{poly}\log n)$ that fool $\mathsf{AC}^0[p]$ circuits, p prime, on n bits. As stated in Lemma 4.17, these circuits cannot compute majority on instances of an appropriate length $\operatorname{poly}\log n$ with probability $\geq 1/2-o(1)$. From this, it follows via techniques by Shaltiel and Viola [37] that there exists some integer $c \leq \operatorname{poly}\log n$ 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 $H(1/2+1/c)n \leq n(1-1/\operatorname{poly}\log n)$, where H is the binary entropy function. Up to the $\operatorname{poly}\log n$, 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.

4.7 Resamplability and worst-case to average-case reductions

We now observe that resamplability implies a worst-case to average-case connection. The proposition below first states a connection for general promise problems and then draws a corollary for problems of the form $\Pi(f)$ and $\Pi'(f)$ for a function f. Recall the problems were defined in Section 4.2.

Proposition 4.21. *Consider unbounded fan-in circuits on any set of gates including And, Or, and Not. For every $\varepsilon > 0$ and sufficiently large n the following holds: Let $\Pi = \Pi_Y \bigcup \Pi_N \subseteq \{0,1\}^n$ be a problem that is resamplable by circuits of size s_R and depth d_R. If there is a circuit C of size s and depth d such that*

$$|\Pr[C(\Pi_Y)=1]-\Pr[C(\Pi_N)=1]| \geq \varepsilon$$

then there is a circuit C' of size $\operatorname{poly}(n,s_R,s)$ and depth $d_R+d+O(1)$ that solves Π on every input.

In particular, let $f:\{0,1\}^n \to \{0,1\}$ be a function that is balanced and such that the problem $\Pi(f)$ (resp. $\Pi'(f)$) is resamplable by circuits of size s_R and depth d_R. If there is a circuit C of size s and depth d such that

$$\Pr_x[C(x)=f(x)] \geq 1/2+\varepsilon$$

then there is a circuit C' of size $\operatorname{poly}(n,s_R,s)$ and depth $d_R+d+O(1)$ such that $C'(x)=f(x)$ for every x.

Proof of Proposition 4.21. Let R be the resampler and consider the circuit $C(R(\cdot))$. By definition of resampler, on any input $x \in \Pi_Y$,

$$\Pr_R[C(R(x))=1] = \Pr[C(\Pi_Y)=1] =: p_Y\,,$$

while on any input $x \in \Pi_N$,

$$\Pr_R[C(R(x))=1] = \Pr[C(\Pi_N)=1] =: p_N\,.$$

By assumption, these two probabilities are bounded away by a constant $\varepsilon > 0$. Assume without loss of generality that $p_Y > p_N$.

Now consider, on any input x, repeating the computation $C(R(x))$ $O(n/\varepsilon^2)$ times in parallel with independent choices for the resampler. By a Chernoff bound, if $x \in \Pi_Y$ then $> p_Y - \varepsilon/2$ fraction of outputs will be 1 with probability $> 1 - 2^{-n}$, while if $x \in \Pi_N$ then $< p_N + \varepsilon/2$ fraction of outputs will be 1 with probability $> 1 - 2^{-n}$.

By a union bound over all $\leq 2^n$ inputs, we can fix the random choices for the resamplers so that this holds on any input.

To conclude, we need to distinguish bit strings of length $O(n/\varepsilon^2)$ with $> p_Y - \varepsilon/2$ fraction of ones from those with $< p_N + \varepsilon/2$ fraction of ones. Since $\varepsilon > 0$ is a constant, this can be done by circuits of polynomial size and depth 3 [3, 4].

The size and depth bounds are immediate by construction.

The "in particular" part follows for $\Pi(f)$ by noting that the circuit C can be used to distinguish $\Pi(f)_Y$ from $\Pi(f)_N$ with advantage 2ε: on input (x,b) output $C(x)+b+1 \mod 2$.

The "in particular" part follows for $\Pi'(f)$ using Fact 4.19. □

Acknowledgments. We thank Scott Aaronson, Yi-Kai Liu, and Mike Saks for helpful discussions. We also thank the anonymous referees for helpful feedback.

A A unitary matrix 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 Section 2.4.1 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 $b \in Q$

$$v_{a,b}[x,y] = \begin{cases} +1 & x = 0, y = b\,, \\ +1 & x \in Q, y = ax+b\,, \\ -1 & x \in Q, y = ax-b\,, \\ 0 & \text{otherwise,} \end{cases} \tag{A.1}$$

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

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

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

$$u_c[x,y] = \begin{cases} +1 & x = c, y \in \mathbb{F}_q\,, \\ 0 & \text{otherwise.} \end{cases} \tag{A.3}$$

Lemma A.1. *The vectors defined in equations* (A.1)*,* (A.2) *and* (A.3) *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 equation (A.1) and the other from equation (A.2) then the supports are disjoint. So we assume both come from equation (A.1) or both come from equation (A.2).

   (a) Both come from equation (A.1) 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:

   $$(x = -2b/(a-a'), ax+b = a'x-b) \quad \text{or} \quad (x = 2b/(a-a'), ax-b = a'x+b)\,,$$

   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 equation (A.1) and $b \neq b'$: we have exactly one of the following two intersection points:

   $$(x = (b'-b)/(a-a'), ax+b = a'x+b') \quad \text{or} \quad (x = (-b'+b)/(a-a'), ax-b = a'x-b')\,,$$

   which contributes $+1$ to the inner product, and exactly one of the following two intersection points:

   $$(x = (b'+b)/(a-a'), ax-b = a'x+b') \quad \text{or} \quad (x = (-b'-b)/(a-a'), ax+b = a'x-b')\,,$$

   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 equation (A.2) and $b = b'$: identical to case (1a) above, with $-Q$ in place of Q.

(d) Both come from equation (A.2) 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 equation (A.1), 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 equation (A.2), 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. □

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?

References

[1] SCOTT AARONSON: A counterexample to the Generalized Linial-Nisan Conjecture. *Electron. Colloq. on Comput. Complexity (ECCC)*, 17:109, 2010. ECCC. 812

[2] SCOTT AARONSON: BQP and the polynomial hierarchy. In *Proc. 42nd STOC*, pp. 141–150. ACM Press, 2010. See also in ECCC. [doi:10.1145/1806689.1806711] 812, 817, 818, 823, 832

[3] MIKLÓS AJTAI: Σ_1^1-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983. [doi:10.1016/0168-0072(83)90038-6] 832, 836

[4] MIKLÓS AJTAI AND MICHAEL BEN-OR: A theorem on probabilistic constant depth computations. In *Proc. 16th STOC*, pp. 471–474. ACM Press, 1984. [doi:10.1145/800057.808715] 832, 836

[5] BENNY APPLEBAUM, YUVAL ISHAI, AND EYAL KUSHILEVITZ: Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. Preliminary version in FOCS'04. [doi:10.1137/S0097539705446950] 833

[6] JAMES ASPNES, RICHARD BEIGEL, MERRICK L. FURST, AND STEVEN RUDICH: The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994. Preliminary version in STOC'91. [doi:10.1007/BF01215346] 814, 830, 831

[7] BOAZ BARAK, RONEN SHALTIEL, AND AVI WIGDERSON: Computational analogues of entropy. In *Proc. 7th Internat. Workshop on Randomization and Computation (RANDOM'03)*, pp. 200–215. Springer, 2003. [doi:10.1007/978-3-540-45198-3_18] 811, 826

[8] RICHARD BEIGEL: When do extra majority gates help? Polylog(N) majority gates are equivalent to one. *Comput. Complexity*, 4(4):314–324, 1994. Preliminary version in STOC'92. [doi:10.1007/BF01263420] 814, 830, 831

[9] ETHAN BERNSTEIN AND UMESH V. VAZIRANI: Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. Preliminary version in STOC'93. [doi:10.1137/S0097539796300921] 812

[10] MANUEL BLUM AND SILVIO MICALI: How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. Preliminary version in FOCS'82. [doi:10.1137/0213053] 810

[11] MARK BRAVERMAN, ANUP RAO, RAN RAZ, AND AMIR YEHUDAYOFF: Pseudorandom generators for regular branching programs. In *Proc. 51st FOCS*, pp. 40–47. IEEE Comp. Soc. Press, 2010. See also at ECCC. [doi:10.1109/FOCS.2010.11] 813, 824

[12] JOSHUA BRODY AND ELAD VERBIN: The coin problem, and pseudorandomness for branching programs. In *Proc. 51st FOCS*, pp. 30–39. IEEE Comp. Soc. Press, 2010. [doi:10.1109/FOCS.2010.10] 813, 824, 825

[13] JOAN FEIGENBAUM AND LANCE FORTNOW: Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. Preliminary version in SCT'91. [doi:10.1137/0222061] 815

[14] YUVAL FILMUS: Smolensky's polynomial method, 2010. See author's home page. 830, 833

[15] ODED GOLDREICH: *Foundations of Cryptography, Volume 1: Basic Tools*. Cambridge Univ. Press, 2001. See also author's home page. 810

[16] ODED GOLDREICH AND LEONID A. LEVIN: A hard-core predicate for all one-way functions. In *Proc. 21st STOC*, pp. 25–32. ACM Press, 1989. [doi:10.1145/73007.73010] 810

[17] ODED GOLDREICH AND AVI WIGDERSON: Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures & Algorithms*, 11(4):315–343, 1997. Preliminary version in STOC'94. See also at ECCC. [doi:10.1002/(SICI)1098-2418(199712)11:4<315::AID-RSA3>3.0.CO;2-1] 824

[18] SHAFI GOLDWASSER, DAN GUTFREUND, ALEXANDER HEALY, TALI KAUFMAN, AND GUY N. ROTHBLUM: Verifying and decoding in constant depth. In *Proc. 39th STOC*, pp. 440–449. ACM Press, 2007. [doi:10.1145/1250790.1250855] 833

[19] SHAFI GOLDWASSER, DAN GUTFREUND, ALEXANDER HEALY, TALI KAUFMAN, AND GUY N. ROTHBLUM: A (de)constructive approach to program checking. In *Proc. 40th STOC*, pp. 143–152. ACM Press, 2008. See also at ECCC. [doi:10.1145/1374376.1374399] 833

[20] SHAFI GOLDWASSER AND SILVIO MICALI: Probabilistic encryption. *J. Comput. System Sci.*, 28(2):270–299, 1984. [doi:10.1016/0022-0000(84)90070-9] 810

[21] VENKATESAN GURUSWAMI, CHRISTOPHER UMANS, AND SALIL P. VADHAN: Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *J. ACM*, 56(4), 2009. Preliminary version in CCC'07. [doi:10.1145/1538902.1538904] 826

[22] JOHAN HÅSTAD: *Computational limitations of small-depth circuits*. MIT Press, 1987. [ACM:SERIES9056.27031] 814, 831, 832

[23] JOHAN HÅSTAD, RUSSELL IMPAGLIAZZO, LEONID A. LEVIN, AND MICHAEL LUBY: A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. Preliminary versions in STOC'89 and STOC'90. [doi:10.1137/S0097539793244708] 810

[24] RUSSELL IMPAGLIAZZO: Hard-core distributions for somewhat hard problems. In *Proc. 36th FOCS*, pp. 538–545. IEEE Comp. Soc. Press, 1995. [doi:10.1109/SFCS.1995.492584] 827

[25] RUSSELL IMPAGLIAZZO, NOAM NISAN, AND AVI WIGDERSON: Pseudorandomness for network algorithms. In *Proc. 26th STOC*, pp. 356–364. ACM Press, 1994. [doi:10.1145/195058.195190] 810, 811, 812, 813, 824, 825

[26] YUVAL ISHAI AND EYAL KUSHILEVITZ: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pp. 294–304. IEEE Comp. Soc. Press, 2000. [doi:10.1109/SFCS.2000.892118] 814, 829, 833

[27] YUVAL ISHAI AND EYAL KUSHILEVITZ: Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th Internat. Colloq. on Automata, Languages and Programming (ICALP'02)*, pp. 244–256. Springer, 2002. [doi:10.1007/3-540-45465-9_22] 814, 829, 830, 833

[28] ALEXEI KITAEV, ALEXANDER SHEN, AND MIKHAIL VYALYI: *Classical and Quantum Computation*. Amer. Math. Soc., 2002. [ACM:863284] 816

[29] MICHAL KOUCKÝ, PRAJAKTA NIMBHORKAR, AND PAVEL PUDLÁK: Pseudorandom generators for group products: extended abstract. In *Proc. 43rd STOC*, pp. 263–272. ACM Press, 2011. [doi:10.1145/1993636.1993672] 813, 824

[30] CHI-JEN LU, SHI-CHUN TSAI, AND HSIN-LUNG WU: Complexity of hard-core set proofs. *Comput. Complexity*, 20(1):145–171, 2011. Preliminary version in ICALP'07. [doi:10.1007/s00037-011-0003-7] 827

[31] NOAM NISAN: Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991. [doi:10.1007/BF01375474] 810, 813, 819

[32] NOAM NISAN: Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. Preliminary version in STOC'90. [doi:10.1007/BF01305237] 810, 812

[33] NOAM NISAN AND AVI WIGDERSON: Hardness vs randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994. Preliminary version in FOCS'88. [doi:10.1016/S0022-0000(05)80043-1] 810, 811, 818, 819

[34] NOAM NISAN AND DAVID ZUCKERMAN: Randomness is linear in space. *J. Comput. System Sci.*, 52(1):43–52, 1996. Preliminary version in STOC'93. [doi:10.1006/jcss.1996.0004] 824

[35] RAN RAZ AND OMER REINGOLD: On recycling the randomness of states in space bounded computation. In *Proc. 31st STOC*, pp. 159–168. ACM Press, 1999. [doi:10.1145/301250.301294] 824

[36] ALEXANDER RAZBOROV: Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *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. [doi:10.1007/BF01137685] 814

[37] RONEN SHALTIEL AND EMANUELE VIOLA: Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. Preliminary version in STOC'08. See also at ECCC. [doi:10.1137/080735096] 811, 827, 832, 835

[38] ROMAN SMOLENSKY: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. 19th STOC*, pp. 77–82. ACM Press, 1987. [doi:10.1145/28395.28404] 814, 830

[39] ROMAN SMOLENSKY: On representations by low-degree polynomials. In *Proc. 34th FOCS*, pp. 130–138. IEEE Comp. Soc. Press, 1993. [doi:10.1109/SFCS.1993.366874] 814, 830, 833

[40] MADHU SUDAN, LUCA TREVISAN, AND SALIL P. VADHAN: Pseudorandom generators without the XOR lemma. *J. Comput. System Sci.*, 62(2):236–266, 2001. Preliminary version in STOC'99. See also at ECCC. [doi:10.1006/jcss.2000.1730] 826, 827

[41] EMANUELE VIOLA: Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM J. Comput.*, 36(5):1387–1403, 2007. Preliminary version in CCC'05. See also at ECCC. [doi:10.1137/050640941] 830

[42] EMANUELE VIOLA: On approximate majority and probabilistic time. *Comput. Complexity*, 18(3):337–375, 2009. Preliminary version in CCC'07. [doi:10.1007/s00037-009-0267-3] 832

[43] EMANUELE VIOLA: On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009. [doi:10.1561/0400000033] 833

[44] EMANUELE VIOLA: The complexity of distributions. *SIAM J. Comput.*, 41(1):191–218, 2012. Preliminary version in FOCS'10. [doi:10.1137/100814998] 830

[45] JOHN WATROUS: Succinct quantum proofs for properties of finite groups. In *Proc. 41st FOCS*, pp. 537–546. IEEE Comp. Soc. Press, 2000. [doi:10.1109/SFCS.2000.892141] 812

[46] RYAN WILLIAMS: Non-uniform ACC circuit lower bounds. In *Proc. 26th IEEE Conf. on Computational Complexity (CCC'11)*, pp. 115–125. IEEE Comp. Soc. Press, 2011. [doi:10.1109/CCC.2011.36] 814

[47] RYAN WILLIAMS: Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. Preliminary version in STOC'10. [doi:10.1137/10080703X] 814

[48] ANDREW CHI-CHIH YAO: Theory and applications of trapdoor functions (extended abstract). In *Proc. 23rd FOCS*, pp. 80–91. IEEE Comp. Soc. Press, 1982. [doi:10.1109/SFCS.1982.45] 810

[49] DAVID ZUCKERMAN: Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997. Preliminary version in STOC'96. [doi:10.1002/(SICI)1098-2418(199712)11:4<345::AID-RSA4>3.0.CO;2-Z] 826

AUTHORS

Bill Fefferman
California Institute of Technology
wjf@caltech.edu
http://www.its.caltech.edu/~wjf/

Ronen Shaltiel
Professor
University of Haifa
ronen@cs.haifa.ac.il
http://www.cs.haifa.ac.il/~ronen

Christopher Umans
Professor
California Institute of Technology
umans@caltech.edu
http://users.cms.caltech.edu/~umans/

Emanuele Viola
Professor
Northeastern University
viola@ccs.neu.edu
http://www.ccs.neu.edu/home/viola/

ABOUT THE AUTHORS

BILL FEFFERMAN is a Ph. D. student at Caltech, in the Department of Computer Science and the Institute for Quantum Information and Matter, advised by Alexei Kitaev and Chris Umans. His research focus is quantum complexity theory. This is his second article in Theory of Computing.

RONEN SHALTIEL graduated from the Hebrew University in 2001; his advisor was Avi Wigderson. His thesis focused on pseudorandom generators and extractors which remain his main focus until today. He spent time at the Institute for Advanced Study and did a postdoc at the Weizmann Institute with Oded Goldreich and Moni Naor. He is reluctant to provide personal details on an academic platform.

CHRISTOPHER UMANS graduated from U.C. Berkeley in 2000; his advisor was Christos Papadimitriou. After a postdoc in the Theory Group at Microsoft Research, he joined Caltech where he is now a professor of Computer Science. He is interested in derandomization, explicit constructions, algebraic complexity and algorithms, and hardness of approximation. Much of his time outside work is spent with his young children, Kira and Daniel.

EMANUELE VIOLA has been at Northeastern University, Boston, for five years. The picture that appears on the website was taken by his sister Alessandra in 2012 at a trattoria in Garbatella, a quaint neighborhood of Rome where Emanuele has spent countless hours wandering.