Pseudorandom Generators and Typically-Correct Derandomization

Jeff Kinne[1] *, Dieter van Melkebeek[1] **, and Ronen Shaltiel[2] ***

[1] Department of Computer Sciences, University of Wisconsin-Madison, USA
{jkinne,dieter}@cs.wisc.edu
[2] Department of Computer Science, University of Haifa, Israel
ronen@cs.haifa.ac.il

Abstract. The area of derandomization attempts to provide efficient deterministic simulations of randomized algorithms in various algorithmic settings. Goldreich and Wigderson introduced a notion of "typically-correct" deterministic simulations, which are allowed to err on few inputs. In this paper we further the study of typically-correct derandomization in two ways.

First, we develop a generic approach for constructing typically-correct derandomizations based on seed-extending pseudorandom generators, which are pseudorandom generators that reveal their seed. We use our approach to obtain both conditional and unconditional typically-correct derandomization results in various algorithmic settings. We show that our technique strictly generalizes an earlier approach by Shaltiel based on randomness extractors, and simplifies the proofs of some known results. We also demonstrate that our approach is applicable in algorithmic settings where earlier work did not apply. For example, we present a typically-correct polynomial-time simulation for every language in BPP based on a hardness assumption that is weaker than the ones used in earlier work.

Second, we investigate whether typically-correct derandomization of BPP implies circuit lower bounds. Extending the work of Kabanets and Impagliazzo for the zero-error case, we establish a positive answer for error rates in the range considered by Goldreich and Wigderson. In doing so, we provide a simpler proof of the zero-error result. Our proof scales better than the original one and does not rely on the result by Impagliazzo, Kabanets, and Wigderson that NEXP having polynomial-size circuits implies that NEXP coincides with EXP.

1 Introduction

Randomized Algorithms and Derandomization One of the central topics in the theory of computing deals with the power of randomness – can randomized proce-

* Partially supported by NSF award CCR-0728809 and by a Cisco Systems Distinguished Graduate Fellowship.
** Research done while visiting the University of Haifa and the Weizmann Institute of Science. Partially supported by NSF award CCR-0728809.
*** Partially supported by BSF grant 2004329 and ISF grant 686/07.

dures be efficiently simulated by deterministic ones? In some settings exponential gaps have been established between randomized and deterministic complexity; in some settings efficient derandomizations[3] are known; in others the question remains wide open. The most famous open setting is that of time-bounded computations, i.e., whether BPP=P, or more modestly, whether BPP lies in deterministic subexponential time. A long line of research gives "hardness versus randomness tradeoffs" for this problem (see [12] for an introduction). These are *conditional results* that give derandomizations assuming a hardness assumption (typically circuit lower bounds of some kind), where the efficiency of the derandomization depends on the strength of the hardness assumption. The latter is used to construct an efficient *pseudorandom generator*, which is a deterministic procedure G that stretches a short "seed" s into a longer "pseudorandom string" $G(s)$ with the property that the uniform distribution on pseudorandom strings is computationally indistinguishable from the uniform distribution on all strings. G allows us to derandomize a randomized procedure $A(x, r)$ that takes an input x and a string r of "coin tosses" as follows: We run the pseudorandom generator on all seeds to produce all pseudorandom strings of length $|r|$; for each such pseudorandom string we run A using that pseudorandom string as "coin tosses", and output the majority vote of the answers of A. Note that this derandomization procedure takes time that is exponential in the seed length of the pseudorandom generator. For example, efficient pseudorandom generators with logarithmic seed length imply that BPP=P, whereas subpolynomial seed length only yields simulations of BPP in deterministic subexponential time.

Typically-Correct Derandomization Weaker notions of derandomization have been studied, in which the deterministic simulation is allowed to err on some inputs. Impagliazzo and Wigderson were the first to consider derandomizations that succeed with high probability on any efficiently samplable distribution; related notions have subsequently been investigated in [8, 20, 4, 17]. Goldreich and Wigderson [3] introduced a weaker notion in which the deterministic simulation only needs to behave correctly on most inputs of any given length. We refer to such simulations as "typically-correct derandomizations". The hope is to construct typically-correct derandomizations that are more efficient than the best-known everywhere-correct derandomizations, or to construct them under weaker assumptions than the hypotheses needed for everywhere-correct derandomization.

Previous Work on Typically-Correct Derandomization Goldreich and Wigderson [3] had the key idea to obtain typically-correct derandomizations by "extracting randomness from the input": extract $r = E(x)$ in a deterministic way such that $B(x) = A(x, E(x))$ behaves correctly on most inputs. If this approach works (as such) and E is efficient, the resulting typically-correct derandomization B

[3] In this paper the term "derandomization" always refers to "full derandomization", i.e., obtaining equivalent deterministic procedures that do not involve randomness at all.

has essentially the same complexity as the original randomized procedure A. In principle, the approach is limited to algorithms A that use no more than $|x|$ random bits; by combining it with pseudorandom generators one can try to handle algorithms that use a larger number of random bits. Goldreich and Wigderson managed to get the approach to work *unconditionally* for logspace algorithms for undirected connectivity, a problem which has been fully derandomized by now [15]. Under a *hardness assumption* that is not known to imply BPP=P, namely that there are functions that are *mildly* hard on average for small circuits with access to an oracle for satisfiability, they showed that BPP has polynomial-time typically-correct derandomizations that err on very few inputs, namely at most a subexponential number. Their construction uses Trevisan's extractor [19].

Zimand [24] showed *unconditional* typically-correct derandomizations with polynomial overhead for sublinear-time algorithms, which can be viewed as randomized decision trees that use a sublinear number of random bits. Zimand's approach relies on a notion of randomness extractors called "exposure-resilient extractors" introduced in [23].

Shaltiel [16] described a generic approach to obtain typically-correct derandomization results. Loosely speaking he showed how to construct a typically-correct derandomization for any randomized procedure that uses a sublinear amount of randomness when given an extractor with exponentially small error that extracts randomness from distributions that are "recognizable by the procedure." We elaborate on Shaltiel's approach in Section 4. Using this approach and "off the shelf" randomness extractors, Shaltiel managed to reproduce Zimand's result for decision trees as well as realize *unconditional* typically-correct derandomizations for 2-party communication protocols and streaming algorithms.

Shaltiel also combined his approach with pseudorandom generator constructions to handle procedures that require a polynomial number of random bits. He obtained typically-correct derandomizations with a polynomially small error rate for randomized algorithms computable by polynomial-sized constant-depth circuits, based on the known hardness of parity for such circuits. He also derived a *conditional* typically-correct derandomization result for BPP under a hardness hypothesis that is incomparable to the Goldreich-Wigderson hypothesis (and is also not known to imply BPP=P), namely that there are functions that are *very* hard on average for small circuits without access to an oracle for satisfiability. The resulting error rate is exponentially small. For both results Shaltiel applies the pseudorandom generators that follow from the hardness versus randomness tradeoffs twice: once to reduce the need for random bits to sublinear, and once to construct the required randomness extractor with exponentially small error. Whereas the first pseudorandom generator application can do with functions that are *mildly* hard on average, the second one requires functions that are *very* hard on average.

Our Approach In this paper we develop an alternative generic approach for constructing typically-correct derandomizations. The approach builds on "seed-extending pseudorandom generators" rather than "extractors". A seed-extending pseudorandom generator is a generator G which outputs the seed as part of the

pseudorandom string, i.e., $G(s) = (s, E(s))$ for some function E. [4] The well-known Nisan-Wigderson pseudorandom generator construction [14] can easily be made seed-extending. We show that whenever a seed-extending pseudorandom generator passes certain statistical tests defined by the randomized procedure $A(x, r)$, the deterministic procedure $B(x) = A(x, E(x))$ forms a typically-correct derandomization of A, where the error rate depends on the error probability of the original randomized algorithm and on the error of the pseudorandom generator.

Note that this approach differs from the typical use of pseudorandom generators in derandomization, where the pseudorandom generator G is run on every seed. As the latter induces a time overhead that is exponential in the seed length, one aims for pseudorandom generators that are computable in time exponential in the seed length. A polynomial-time simulation is achieved only in the case of logarithmic seed lengths. In contrast, we run G *only once*, namely with the input x of the randomized algorithm as the seed. We use the pseudorandom generator to *select* one "coin toss sequence" $r = E(x)$ on which we run the randomized algorithm. As opposed to the traditional derandomization setting, our approach benefits from pseudorandom generators that are computable in time less than exponential in the seed length. With a pseudorandom generator computable in time polynomial in the output length, we obtain nontrivial polynomial-time typically-correct derandomizations even when the seed length is just subpolynomial.

Our approach has the advantage of being more direct than the one of [16], in the sense that it derandomizes the algorithm A in "one shot". More importantly, it obviates the second use of pseudorandom generators in Shaltiel's approach and allows us to start from the *weaker assumption* that there are functions which are *mildly* hard on average for small circuits without access to an oracle for satisfiability.

While our assumption is weaker than both the one in [3] and the one in [16], the error rate of our typically-correct derandomizations is only polynomially small. We can decrease the error rate by strengthening the hardness assumption. Under the same hardness assumption as [16] our approach matches the exponentially small error rate in that paper.

We can similarly relax the hardness assumption in a host of other settings. In some cases this allows us to establish new *unconditional* typically-correct derandomizations, namely for models where functions that are *very* hard on average are not known but functions which are only *mildly* hard on average are known unconditionally.

We also determine the precise relationship between our approach and Shaltiel's. We show that in the range of exponentially small error rates, "extractors for recognizable distributions" are equivalent to seed-extending pseudorandom gen-

[4] Borrowing from the similar notion of "strong extractors" in the extractor literature, such pseudorandom functions have been termed "strong" in earlier papers. In coding-theoretic terms, they could also be called "systematic". However, we find the term "seed-extending" more informative.

erators that pass the statistical tests we need. This means that all the aforementioned results of [16] can also be obtained using our new approach. Since we can also handle situations where [16] does not apply, our approach is more generic.

Typically-Correct Derandomization and Circuit Lower Bounds Kabanets and Impagliazzo [9] showed that subexponential-time derandomizations of BPP imply circuit lower bounds that seem beyond the scope of current techniques. We ask whether subexponential-time typically-correct derandomizations imply such lower bounds. A main contribution of our paper is an affirmative answer in the case of the error rates considered by Goldreich and Wigderson. The case of higher error rates remains open.

Our result is a strengthening of [9] from the everywhere-correct setting to the typically-correct setting. In developing it, we also obtain a simpler proof for the everywhere-correct setting. Our proof scales better than the one in [9], yields the same lower bound for a smaller class, and does not rely on the result from [6] that NEXP having polynomial-size circuits implies that NEXP coincides with EXP.

Organization We start Section 2 with the formal definitions of the notions used throughout the rest of the paper, and the key lemma that shows how seed-extending pseudorandom generators yield typically-correct derandomizations. In Section 3 we state and discuss both the conditional and unconditional results we obtain by applying our approach using the Nisan-Wigderson pseudorandom generator construction. In Section 4 we give a detailed comparison of our approach with Shaltiel's extractor-based approach. In Section 5 we describe our results on circuit lower bounds that follow from typically-correct and everywhere-correct derandomization of BPP. Due to space limitations all formal proofs are deferred to the full version of this paper.

2 Typically-Correct Derandomization and the PRG Approach

Notation and Concepts We use the following terminology throughout the paper. We view a randomized algorithm as defined by a deterministic algorithm $A(x, r)$ where x denotes the input and r the string of "coin tosses". We typically restrict our attention to one input length n, in which case A becomes a function $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ where m represents the number of random bits that A uses on inputs of length n. We say that $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ *computes* a function $L : \{0,1\}^n \to \{0,1\}$ with error ρ if for every $x \in \{0,1\}^n$, $\Pr_{R \leftarrow U_m}[A(x, R) \neq L(x)] \leq \rho$, where U_m denotes the uniform distribution over $\{0,1\}^m$. We say that the randomized algorithm A *computes* a language L with error $\rho(\cdot)$, if for every input length n, the function A computes the function L with error $\rho(n)$.

Given a randomized algorithm A for L, our goal is to construct a deterministic algorithm B of complexity comparable to A that is typically correct for L. By

the latter we mean that B and L agree on most inputs of any given length, or equivalently, that the relative Hamming distance between B and L at any given length is small.

Definition 1 (typically-correct behavior). *Let $L : \{0,1\}^n \to \{0,1\}$ be a function. We say that a function $B : \{0,1\}^n \to \{0,1\}$ is* within distance δ *of L if $\Pr_{X \leftarrow U_n}[B(X) \neq L(X)] \leq \delta$. We say that an algorithm B* computes *a language L* to within $\delta(\cdot)$ *if for every input length n, the function B is within distance $\delta(n)$ of the function L.*

In general, a function $G : \{0,1\}^n \to \{0,1\}^\ell$ is ϵ-pseudorandom for a test $T : \{0,1\}^\ell \to \{0,1\}$ if $|\Pr_{S \leftarrow U_n}[T(G(S)) = 1] - \Pr_{R \leftarrow U_\ell}[T(R) = 1]| \leq \epsilon$. In this paper we are dealing with tests $T(x,r)$ that receive two inputs, namely x of length n and r of length m, and with corresponding pseudorandom functions G of the form $G(x) = (x, E(x))$, where x is of length n and $E(x)$ of length m. We call such functions "seed-extending".

Definition 2 (seed-extending function). *A function $G : \{0,1\}^n \to \{0,1\}^{n+m}$ is* seed-extending *if it is of the form $G(x) = (x, E(x))$ for some function $E : \{0,1\}^n \to \{0,1\}^m$. We refer to the function E as the extending part of G.*

Note that a seed-extending function G with extending part E is ϵ-pseudorandom for a test $T : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ if

$$|\Pr_{X \leftarrow U_n, R \leftarrow U_m}[T(X,R) = 1] - \Pr_{X \leftarrow U_n}[T(X, E(X)) = 1]| \leq \epsilon.$$

A seed-extending $\epsilon(\cdot)$-pseudorandom generator for a family of tests $\mathcal{T}$ is a deterministic algorithm G such that for every input length n, G is a seed-extending $\epsilon(n)$-pseudorandom function for the tests in $\mathcal{T}$ corresponding to input length n.

The Seed-Extending Pseudorandom Generator Approach Our key observation is that good seed-extending pseudorandom generators G for certain simple tests based on the algorithm A yield good typically-correct derandomizations of the form $B(x) = A(x, E(x))$. The following lemma states the quantitative relationship.

Lemma 1. *Let $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ and $L : \{0,1\}^n \to \{0,1\}$ be functions such that*

$$\Pr_{X \leftarrow U_n, R \leftarrow U_m}[A(X,R) \neq L(X)] \leq \rho. \tag{1}$$

Let $G : \{0,1\}^n \to \{0,1\}^{n+m}$ be a seed-extending function with extending part E, and let $B(x) = A(x, E(x))$.

1. *If G is ϵ-pseudorandom for tests of the form $T(x,r) = A(x,r) \oplus L(x)$, then B is within distance $\rho + \epsilon$ of L*
2. *If G is ϵ-pseudorandom for tests of the form $T_{r'}(x,r) = A(x,r) \oplus A(x,r')$ where $r' \in \{0,1\}^m$ is an arbitrary string, then B is within distance $3\rho + \epsilon$ of L.*

Note that if A computes L with error ρ then condition (1) of the lemma is met. The two parts of the lemma differ in the complexity of the tests and in the error bound. The complexity of the tests plays a critical role for the existence of pseudorandom generators. In the first item the tests use the language L as an oracle, which may result in too high a complexity. In the second item we reduce the complexity of the tests at the cost of introducing non-uniformity and increasing the error bound. The increase in the error bound is often not an issue as we can easily reduce ρ by slightly amplifying the original algorithm A before applying the lemma.

The Nisan-Wigderson Construction Some of the constructions of pseudorandom generators in the literature are seed-extending or can be easily modified to become seed-extending. One such example is the Nisan-Wigderson construction [14], which builds a pseudorandom generator for a given class of randomized algorithms out of a language that is hard on average for a related class of algorithms. We use the following terminology for the latter.

Definition 3 (hardness on average). *A language L is $\delta(\cdot)$-*hard *for a class of algorithms $\mathcal{A}$ if no $A \in \mathcal{A}$ is within distance $\delta(n)$ of L for almost all input lengths n.*

We use the Nisan-Wigderson construction for all our results in the next section. Some of the results are conditioned on reasonable but unproven hypotheses regarding the existence of languages that are hard on average. Others are unconditional because languages of the required hardness have been proven to exist.

3 Applications

3.1 Conditional Results

The first setting we consider is that of BPP. We use a modest hardness assumption to show that any language in BPP has a polynomial-time deterministic algorithm that errs on a polynomially small fraction of the inputs.

Theorem 1. *Let L be a language in BPP that is computed by a randomized bounded-error polynomial-time algorithm A. For any positive constant c, there is a positive constant d depending on c and the running time of A such that the following holds. If there is a language H in P that is $\frac{1}{n^c}$-hard for circuits of size n^d, then there is a deterministic polynomial-time algorithm B that computes L to within $\frac{1}{n^c}$.*

Comparison to Previous Work We now compare Theorem 1 to previous conditional derandomization results for BPP. We first consider everywhere-correct results. Plugging our assumption into the hardness versus randomness tradeoffs of [14] gives the incomparable result that BPP is in deterministic subexponential time, i.e., in time 2^{n^ϵ} for every positive constant ϵ. We remark that to obtain

this result one can relax the assumption and allow the language H to be in deterministic linear-exponential time, i.e., E=DTIME($2^{O(n)}$).

We next compare Theorem 1 to previous conditional results on typically-correct derandomization of BPP [3, 16]. The assumption that we use is weaker than the assumptions that are used by previous work. More specifically, [3] needs H to be $\frac{1}{n^c}$-hard for circuits of size n^d with a SAT oracle, and [16] requires that H be $(\frac{1}{2} - \frac{1}{2^{n^{\Omega(1)}}})$-hard for circuits of size n^d.

Thus, the two aforementioned results do not yield any typically-correct derandomization when starting from the modest assumption that we use. Under their respective stronger assumptions, the other approaches do yield typically-correct algorithms that are closer to L. We remark that we can match the distance in [16] if we are allowed to assume the same hardness hypothesis.

Extensions to Other Algorithmic Settings [11] observed that the proof of the Nisan-Wigderson generator [14] relativizes and used this fact to give hardness versus randomness tradeoff results in a number of different algorithmic settings. This approach also works within our typically-correct derandomization framework.

Some consequences are listed in the table below for the classes AM, BP.$\oplus$P and BP.L, where the latter refers to randomized algorithms that run in logarithmic space and are allowed two-way access to their random coins [13]. We could also state similar results for the other settings considered by [11]. For each of these complexity classes we need to assume a different hardness assumption, where the difference lies in the type of circuits and in the uniform class to consider. We remark that for BP.$\oplus$P we only need a worst-case hardness assumption as in this setting worst-case hardness is known to imply average-case hardness [2].

Setting	Hardness Assumption	Conclusion
AM=BP.NP	NP $\cap$ coNP $\frac{1}{n^c}$-hard for SIZE$^{\mathrm{SAT}}(n^d)$	AM within $\frac{1}{n^c}$ of NP
BP.$\oplus$P	$\oplus$P $\not\subseteq$ SIZE$^{\oplus\mathrm{SAT}}(n^d)$	BP.$\oplus$P within $\frac{1}{n^c}$ of $\oplus$P
BP.L	L $\frac{1}{n^c}$-hard for BP-SIZE(n^d)	BP.L within $\frac{1}{n^c}$ of L

In the table, SIZE(s) refers to Boolean circuits of size s, SIZE$^O(\cdot)$ refers to Boolean circuits that have access to oracle gates for the language O, and BP-SIZE(s) refers to branching programs of size s. A class of languages is $\delta(\cdot)$-hard for $\mathcal{A}$ if it contains a language that is $\delta(\cdot)$-hard for $\mathcal{A}$.

3.2 Unconditional Results

Constant Depth Circuits Our techniques imply typically-correct derandomization results for randomized constant-depth polynomial-size circuits. This result uses the fact that the parity function is $(\frac{1}{2} - \frac{1}{2^{n^{\Omega(1)}}})$-hard on average for constant-depth circuits [5] and gives an alternative and simpler proof of a result of [16] in this setting.

Constant Depth Circuits with Few Symmetric Gates In contrast to the approach of [16], our techniques also yield results in settings where the best-known lower bounds only yield moderate hardness on average. One such model is that of constant-depth circuits that are allowed a small number of arbitrary symmetric gates, i.e., gates that compute functions which only depend on the Hamming weight of the input, such as parity and majority. In this setting Viola [21] constructed a function that is $(\frac{1}{2} - \frac{1}{n^{\Omega(\log n)}})$-hard on average. Via the Nisan-Wigderson construction, this in turn translates into a pseudorandom generator with stretch that is quasi-polynomial and error that is polynomially small in the output length, resulting in an error rate that is only quasipolynomially small. Thus, the approach of [16] does not apply, but ours can exploit these weak pseudorandom generators and gives the following result for both log-space and polynomial-time uniformity.

Theorem 2. *Let L be a language and A a uniform randomized bounded-error circuit of constant depth and polynomial size that uses $o(\log^2 n)$ symmetric gates such that A computes L with error at most ρ. Then there is a uniform deterministic circuit B of constant depth and polynomial size that uses exactly the same symmetric gates as A in addition to a polynomial number of parity gates such that B computes L to within $3\rho + \frac{1}{n^{\Omega(\log n)}}$.*

Multi-Party Communication Complexity [16] proves a typically-correct derandomization result for two-party communication protocols. The proof of [16] is tailored to the two-party case and does not extend to the general case of k-party communication in which the players have the inputs on their foreheads [1]. Using our approach we can handle $k > 2$ and show that every uniform randomized k-party communication protocol has a uniform deterministic k-party communication protocol of comparable communication complexity that is typically correct. The following statement holds for both log-space and poly-time uniformity, where we call a communication protocol uniform if whenever a player sends a message, that message can be efficiently computed as a function of the player's view.

Theorem 3. *Let L be a language and A a uniform randomized communication protocol that computes L with error at most ρ and uses k players, q bits of communication, and m bits of public randomness, with k, q, m, and $\log(1/\epsilon)$ functions computable within the uniformity bounds. Then there is a uniform deterministic communication protocol B that computes L to within $3\rho + \epsilon$ and uses k players and $O(2^k \cdot m \cdot (q + \log(m/\epsilon)))$ bits of communication.*

For $k = 2$, Theorem 3 yields a weaker result than that of [16] – which gives a deterministic protocol with communication complexity $O(q + m)$ rather than $O(q \cdot m)$ – although we can also obtain the stronger result using our approach, as explained in the next section.

4 Comparison with the Extractor-Based Approach

We have seen several settings in which seed-extending pseudorandom generators allow us to prove typically-correct derandomization results that do not follow from the extractor-based approach of [16]. We now show that the approach of [16] is essentially equivalent to having seed-extending pseudorandom generators with *exponentially small error*. This reaffirms our claim that our approach is more general since we additionally obtain meaningful results using pseudorandom generators with larger error.

Overview of the Extractor-Based Approach [16] uses a notion of "extractors for recognizable distributions" explained below. For every function $f : \{0,1\}^n \to \{0,1\}$ one can associate the distribution U_f that is *recognized* by f, which is the uniform distribution over $f^{-1}(1) = \{x : f(x) = 1\}$. A function $E : \{0,1\}^n \to \{0,1\}^m$ is a (k,ϵ)-extractor for distributions recognizable by some collection of functions $f : \{0,1\}^n \to \{0,1\}$, if for every such function f with $|f^{-1}(1)| \geq 2^k$, the distribution $E(U_f)$ has statistical distance at most ϵ from the uniform distribution on m bit strings.

[16] shows the following general approach towards typically-correct derandomization. Let $A : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a randomized algorithm that computes some function L with error ρ. Let $\Delta = 100m$ and let E be an $(n-\Delta, 2^{-\Delta})$-extractor for distributions recognizable by functions of the form $f_{r_1,r_2}(x) = A(x,r_1) \oplus A(x,r_2)$ where $r_1, r_2 \in \{0,1\}^m$ are arbitrary strings. Then setting $B(x) = A(x, E(x))$ gives an algorithm that is within $3\rho + 2^{-10m}$ of L.

Comparison The above approach requires extractors with error that is exponentially small in m, and breaks down completely when the error is larger. We now observe that an extractor with exponentially small error yields a seed-extending pseudorandom generator with exponentially small error. It follows that the extractors used in [16] can be viewed as seed-extending pseudorandom generators with exponentially small error.

Theorem 4. *Let $T : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a function. Let $\Delta = m + \log(1/\epsilon) + 1$ and let $E : \{0,1\}^n \to \{0,1\}^m$ be an $(n - \Delta, 2^{-\Delta})$-extractor for distributions recognizable by functions of the form $f_r(x) = T(x,r)$ where $r \in \{0,1\}^m$ is an arbitrary string. Then, $G(x) = (x, E(x))$ is ϵ-pseudorandom for T.*

We remark that in some algorithmic settings, namely decision trees and 2-party communication protocols, the approach of [16] yields typically-correct derandomizations that are more efficient than the ones that follow from applying our methodology directly based on known hardness results. Nevertheless, by Theorem 4 the extractors used in [16] give rise to seed-extending pseudorandom generators that yield typically-correct derandomizations matching the efficiency of the extractor-based approach.

We also observe that seed-extending pseudorandom generators with error that is exponentially small in m yield extractors for recognizable distributions.

Thus, the approach of [16] is essentially equivalent to the special case of seed-extending pseudorandom generators with error that is exponentially small.

Theorem 5. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function and let $E : \{0,1\}^n \to \{0,1\}^m$ be a function such that $G(x) = (x, E(x))$ is seed-extending ϵ-pseudorandom for test $T(x,r)$ of the form $T_z(x,r) = f(x) \wedge (r = z)$ where $z \in \{0,1\}^m$ is an arbitrary string. Assume that $\epsilon < 2^{-3m}$ and let $\Delta = (\log(1/\epsilon) - m)/2 > m$. Then E is an $(n - \Delta, 2^{-\Delta})$-extractor for the distribution recognizable by f.*

Note that seed-extending pseudorandom generators with error $\epsilon < 2^{-m}$ must have $m < n$ (as there are only 2^n seeds). This is why the approach of [16] cannot directly handle randomized algorithms with a superlinear number of random bits. In contrast, in Theorems 1 and 2 we are able to directly handle algorithms with a superlinear number of random bits using pseudorandom generators with larger error.

5 Circuit Lower Bounds

From everywhere-correct derandomization It is well-known that the existence of pseudorandom generators for polynomial-size circuits (which yields everywhere-correct derandomization of BPP) implies that EXP does not have polynomial-size circuits; this is the easy direction of the hardness versus randomness tradeoffs. Impagliazzo et al. [6] showed that everywhere-correct derandomization of promise-BPP into NSUBEXP implies that NEXP does not have polynomial-size circuits. Building on [6], Kabanets and Impagliazzo [9] showed that everywhere-correct derandomization of BPP into NSUBEXP implies that NEXP does not have Boolean circuits of polynomial size or that the permanent over $\mathbb{Z}$ does not have arithmetic circuits of polynomial size. As a byproduct of our investigations, we obtain a simpler proof of the latter result.

We use the following terminology for the statements of our lower bound results. We consider arithmetic circuits with internal nodes representing addition, subtraction, and multiplication, and leaves representing variables and the constants 0 and 1. ACZ denotes the language of all arithmetic circuits that compute the zero polynomial over $\mathbb{Z}$. Perm denotes the permanent of matrices over $\mathbb{Z}$, and 0-1-Perm its restriction to matrices with all entries in $\{0,1\}$. We measure the size of circuits by the string length of their description, and assume that the description mechanism is such that the description of a circuit of size s can easily be padded into the description of an equivalent circuit of size s' for any $s' > s$.

Our approach yields the following general statement regarding everywhere-correct derandomization of the specific BPP-language ACZ.

Theorem 6. *Let $a(n)$, $s(n)$, and $t(n)$ be functions such that $a(n)$ and $s(n)$ are constructible, $a(n)$ and $t(n)$ are monotone, and $s(n) \geq n$. The following holds as long as for every constant c and sufficiently large n,*

$$t\left((s(n))^c \cdot a((s(n)^c))\right) \leq 2^n.$$

If ACZ $\in$ NTIME($t(n)$) *then (i)* NTIME(2^n) $\cap$ coNTIME(2^n) *does not have Boolean circuits of size* $s(n)$*, or (ii) Perm does not have arithmetic circuits of size* $a(n)$*.*

We point out that part (i) states a lower bound for NEXP $\cap$ coNEXP rather than just for NEXP, and Theorem 6 does so for the entire range of the parameters; the proof in [9] only gives such a lower bound in the case where all the parameters are polynomially bounded. More importantly, due to its dependence on the result from [6] that NEXP having polynomial-size circuits implies that NEXP coincides with EXP, the proof in [9] only works when $s(n)$ is polynomially bounded; our proof gives nontrivial results for $s(n)$ ranging between linear and linear-exponential.

From typically-correct derandomization We initiate the study of whether typically-correct derandomization of BPP implies circuit lower bounds. We show that it does in the case of typically-correct derandomizations that run in NSUBEXP and are of the quality considered by Goldreich and Wigderson [3].

Theorem 7. *If for every positive constant* ϵ *there exists a nondeterministic Turing machine which runs in time* 2^{n^ϵ} *and correctly decides ACZ on all but at most* 2^{n^ϵ} *of the inputs of length* n *for almost every* n*, then (i) NEXP does not have Boolean circuits of polynomial size, or (ii) Perm does not have arithmetic circuits of polynomial size.*

Note that Theorem 7 strengthens the main result of [9], which establishes the theorem in the special case where the nondeterministic machines decide ACZ correctly on all inputs. We start with a proof sketch of this weaker result using our new approach, and then show how to adapt it to the setting of typically-correct derandomization with error rates of the order considered in [3].

The proof consists of two parts. We first show that $\mathrm{P}^{\text{0-1-Perm}[1]}$ does not have circuits of fixed polynomial size, where $\mathrm{P}^{\text{0-1-Perm}[1]}$ denotes the class of languages that can be decided in polynomial-time with one query to an oracle for 0-1-Perm. This follows because PH does not have circuits of fixed polynomial size [10], PH is contained in $\mathrm{P}^{\#\mathrm{P}[1]}$ [18], and 0-1-Perm is complete for #P under reductions that make a single query [22].

In the second step we assume that

(α) ACZ has derandomizations N_ϵ of the form described in the statement of Theorem 7 but without any errors, and
(β) Perm has polynomial-size arithmetic circuits,

and show that these hypotheses imply that $\mathrm{P}^{\text{0-1-Perm}[1]}$ is contained in NSUBEXP. The crux is the following single-valued nondeterministic algorithm to compute the permanent of a given 0-1-matrix M over $\mathbb{Z}$.

1. Guess a polynomial-sized candidate arithmetic circuit C for Perm on matrices of the same dimension as M.
2. Verify the correctness of C. Halt and reject if the test fails.

3. Use the circuit C to determine the permanent of M in deterministic polynomial time.

The circuit in step 1 exists by virtue of hypothesis (β). By the downward self-reducibility of Perm, the test in step 2 just has to check an arithmetic circuit identity based on C, which can be verified in nondeterministic subexponential time by virtue of (α).

All together, the hypotheses (α) and (β) imply that NSUBEXP does not have circuits of fixed polynomial size, and therefore neither does NE. Since NE has a complete language under linear-time reductions, the latter implies that NEXP does not have polynomial-size circuits.

Now suppose that our nondeterministic algorithms N_ϵ for ACZ can err on a small number of inputs of any given length ℓ. The test in step 2 above may no longer be sound nor complete. We can make the test sound if we are given the number $\mathrm{fp}(\ell, \epsilon)$ of false positives of length ℓ, i.e., the number of inputs of length ℓ that are accepted by N_ϵ but do not belong to ACZ. This is because we can guess the list of those $\mathrm{fp}(\ell, \epsilon)$ inputs of length ℓ, verify that they are accepted by N_ϵ but do not compute the zero polynomial, and then check that the given input of length ℓ does not appear on the list. We can make the test complete by increasing ℓ a bit and exploiting the paddability of ACZ. Since the number of errors of N_ϵ is relatively small, for any correct circuit C there has to be a pad that N_ϵ accepts. Our test can guess such a pad and check it. If N_ϵ makes no more than 2^{ℓ^ϵ} errors at length ℓ, we obtain simulations of $\mathrm{P}^{\text{0-1-Perm[1]}}$ in NSUBEXP with subpolynomial advice. We conclude that the latter class does not have circuits of fixed polynomial size, which implies that NSUBEXP doesn't, from which we conclude as before that NEXP does not have circuits of polynomial size. This ends our proof sketch of Theorem 7.

Extensions We observe a few variations of Theorems 6 and 7. First, the theorems also hold when we simultaneously replace ACZ by AFZ (the restriction of ACZ to arithmetic formulas), and "arithmetic circuits" by "arithmetic formulas". Second, we can play with the underlying i.o. and a.e. quantifiers. For example, in the case of Theorem 7 it suffices for the nondeterministic machines N_ϵ to correctly decide ACZ on all but at most 2^{n^ϵ} of the inputs of length n for *infinitely many* n. Related to the latter variation, we point out that by [7] EXP differs from BPP iff all of BPP has deterministic typically-correct derandomizations that run in subexponential time and err on no more than a polynomial fraction of the inputs of length n for infinitely many n. Thus, extending this i.o.-version of Theorem 7 to the setting with polynomial error rates would show that EXP$\neq$BPP implies circuit lower bounds.

Acknowledgments We would like to thank Oded Goldreich for suggesting the term "typically-correct derandomization," and Matt Anderson, Salil Vadhan, and anonymous reviewers for helpful comments. The third author thanks Salil Vadhan for suggesting this research direction to him and for collaboration at an early stage of this research.

References

[1] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *JCSS*, 45(2):204–232, 1992.

[2] J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SICOMP*, 22(5), 1993.

[3] O. Goldreich and A. Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *RANDOM*, pages 209–223, 2002.

[4] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Comput. Compl.*, 12(3–4):85–130, 2003.

[5] J. Håstad. *Computational limitations of small-depth circuits*. MIT Press, Cambridge, MA, USA, 1987.

[6] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *JCSS*, 65(4):672–694, 2002.

[7] R. Impagliazzo and A. Wigderson. Randomness vs time: Derandomization under a uniform assumption. *JCSS*, 63(4):672–688, 2001.

[8] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *JCSS*, 63(2):236–252, 2001.

[9] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Compl.*, 13(1/2):1–46, 2004.

[10] R. Kannan. Circuit-size lower bounds and nonreducibility to sparse sets. *Inf. Cont.*, 55(1):40–56, 1982.

[11] A. R. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SICOMP*, 31(5):1501–1526, 2002.

[12] P. B. Miltersen. Derandomizing complexity classes. In *Handbook of Randomized Computing*, pages 843–941. Kluwer Academic Publishers, 2001.

[13] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theor. Comp. Sci.*, 107(1):135–144, 1993.

[14] N. Nisan and A. Wigderson. Hardness vs. randomness. *JCSS*, 49(2):149–167, 1994.

[15] O. Reingold. Undirected connectivity in log-space. *JACM*, 55(4), 2008.

[16] R. Shaltiel. Weak derandomization of weak algorithms: explicit versions of Yao's lemma. In *Proc. Conf. Comput. Compl.*, 2009.

[17] R. Shaltiel and C. Umans. Low-end uniform hardness vs. randomness tradeoffs for AM. In *Proc. of the ACM Symp. Theory of Comp.*, pages 430–439, 2007.

[18] S. Toda. PP is as hard as the polynomial-time hierarchy. *SICOMP*, 20(5):865–877, 1991.

[19] L. Trevisan. Extractors and pseudorandom generators. *JACM*, 48(4):860–879, 2001.

[20] L. Trevisan and S. P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Comput. Compl.*, 16(4):331–364, 2007.

[21] E. Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SICOMP*, 36(5):1387–1403, 2006.

[22] V. Zanko. #P-completeness via many-one reductions. *Intl. J. Found. Comp. Sci.*, 2(1):77–82, 1991.

[23] M. Zimand. Exposure-resilient extractors. In *Proc. Conf. Comput. Compl.*, pages 61–72, 2006.

[24] M. Zimand. Exposure-resilient extractors and the derandomization of probabilistic sublinear time. *Comput. Compl.*, 17(2):220–253, 2008.