Tighter Connections between Derandomization and Circuit Lower Bounds[*]

Marco L. Carmosino[1], Russell Impagliazzo[1], Valentine Kabanets[2], and Antonina Kolokolova[3]

1 Department of Computer Science, University of California San Diego
La Jolla, CA, USA
mcarmosi@cs.ucsd.edu, russell@eng.ucsd.edu

2 School of Computing Science, Simon Fraser University
Burnaby, BC, Canada
kabanets@cs.sfu.ca

3 Department of Computer Science, Memorial University of Newfoundland
St. John's, NL, Canada
kol@mun.ca

Abstract

We tighten the connections between circuit lower bounds and derandomization for each of the following three types of derandomization:

- general derandomization of promise-BPP (connected to Boolean circuits),
- derandomization of Polynomial Identity Testing (PIT) over fixed finite fields (connected to arithmetic circuit lower bounds over the same field), and
- derandomization of PIT over the integers (connected to arithmetic circuit lower bounds over the integers).

We show how to make these connections *uniform equivalences*, although at the expense of using somewhat less common versions of complexity classes and for a less studied notion of inclusion.

Our main results are as follows:

1. We give the first proof that a non-trivial (nondeterministic subexponential-time) algorithm for PIT over a *fixed finite field* yields arithmetic circuit lower bounds.
2. We get a similar result for the case of PIT over the integers, strengthening a result of Jansen and Santhanam [13] (by removing the need for advice).
3. We derive a Boolean circuit lower bound for NEXP ∩ coNEXP from the assumption of sufficiently strong non-deterministic derandomization of promise-BPP (without advice), as well as from the assumed existence of an NP-computable non-empty property of Boolean functions useful for proving superpolynomial circuit lower bounds (in the sense of natural proofs of [20]); this strengthens the related results of [11].
4. Finally, we turn all of these implications into equivalences for appropriately defined promise classes and for a notion of robust inclusion/separation (inspired by [9]) that lies between the classical "almost everywhere" and "infinitely often" notions.

1998 ACM Subject Classification F.2.3 Tradeoffs between Complexity Measures

Keywords and phrases derandomization, circuit lower bounds, polynomial identity testing, promise BPP, hardness vs. randomness

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.645

[*] This work was partially supported by the Simons Foundation and NSF grant CCF-121351 (M. Carmosino, R. Impagliazzo) and by NSERC Discovery grant (V. Kabanets, A. Kolokolova)

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 645–658
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

While randomness has become an indispensable tool for algorithm design, many (though not all) randomized algorithms have later been derandomized, i.e., shown to have equivalent, comparably fast deterministic algorithms. One highly successful method for derandomization of whole classes of algorithms has been the "hardness as randomness" paradigm [2, 7, 28, 19, 5, 12, 24, 22, 26]. In this paradigm, problems that are hard for the type of computation to be derandomized are converted into indistinguishable pseudo-random generators (PRGs) for the same class, which then replace the random choices of the randomized algorithm.

It has long been observed that this method seems to require hardness for the non-uniform version of the class (at least for worst-case derandomization). PRGs are a special case of "black-box" derandomization methods, where the algorithm to be accessed is only modified by replacing the random decisions with deterministic choices. It is relatively straightforward to show that "black-box" derandomization requires a circuit bound against the type of algorithm to be derandomized (see, e.g., [10]). Thus, via the hardness as randomness paradigm, strong circuit lower bounds can be proved *equivalent* to universal "black-box" derandomization.

More surprisingly, there are results that show the reverse direction, that derandomization requires strong circuit lower bounds, is true even for non-black-box algorithms [11]. In fact, even derandomizing a specific algorithm, the randomized polynomial identity test (PIT) of Schwartz and Zippel (also discovered by DeMillo and Lipton) [21, 29, 8], would imply strong lower bounds for arithmetic circuits [14, 13, 1]. However, the proofs of these statements are not direct reductions, but instead go through various complexity class collapses, and the conclusions in one direction are not usually exact matches for the other. So, unlike for "black-box" algorithms, we do not usually get a literal equivalence between a derandomization result and a circuit lower bound (one exception is [13]).

In this paper, we tighten the connections between circuit lower bounds and derandomization in several ways, for each of the three types of derandomization: general derandomization of promise-BPP (connected to Boolean circuits), derandomization of PIT over fixed finite fields (connected to arithmetic circuit lower bounds over the same field), and derandomization of PIT over the integers (connected to arithmetic circuit lower bounds over the integers). We show how to make these connections *equivalences*, although at the expense of using somewhat less common versions of complexity classes and a less studied notion of inclusion, the "robustly-often" inclusion introduced by [9]. Even for worst-case inclusion, we simplify and strengthen the known connections in several ways.

1.1 Our results

Following [13], let ml-NE denote the class of multilinear n-variate polynomials $F = \{f_n\}_{n\geq 0}$ over a domain $\mathcal{D}$ (for $\mathcal{D}$ the set of integers or a finite field) whose graph:

$$\{(a_1, \ldots, a_n, f_n(a_1, \ldots, a_n)) \mid n \geq 0,\ a_i \in \mathcal{D},\ 1 \leq i \leq n\}$$

is in the class $\mathsf{NE} = \mathsf{NTIME}[2^{O(n)}]$.

- For finite fields, we show that derandomization of PIT implies that some polynomial $F \in$ ml-NE does not have polynomial sized arithmetic circuits over the same field, nor does any polynomially bounded power of F.
  This is the first result getting circuit lower bounds from derandomization of a randomized PIT algorithm over fixed *finite* fields[1].

[1] Kabanets and Impagliazzo [14] prove a related but weaker version of the hardness-to-pseudorandomness

- We improve on the advice requirements for known connections between derandomization of PIT over the integers and arithmetic circuit lower bounds over the integers.
- Impagliazzo et al. [11] showed that (even nondeterministic) polynomial-time derandomization of promise-BPP would imply a Boolean circuit lower bound for NEXP. We strengthen this to the circuit lower bound for the smaller class NEXP $\cap$ coNEXP. This is the same class where sufficiently strong circuit lower bounds would imply that promise-BPP $\subseteq$ NP.
- [11] also showed that a Boolean circuit lower bound for NEXP would follow from the existence of an NP-computable non-empty property of Boolean functions that is useful for proving superpolynomial circuit lower bounds (in the sense of natural proofs of Razborov and Rudich [20]). We also strengthen this to the lower bound for the class NEXP $\cap$ coNEXP.
- The two notions of inclusion and hardness most frequently used in complexity are "every length" inclusion and hardness, and "infinitely often" inclusion and hardness. Unfortunately, the negation of everywhere inclusion is only infinitely often hardness, and vice versa. This prevents many of the connections above from being literal equivalences between lower bounds and derandomization. Using a third notion of inclusion, a variant of the "robustly often" inclusion of [9], we make all four of the above connections into literal equivalences[2].

1.2 Overview and related work

How can we argue that an upper bound on algorithmic complexity (efficient derandomization) yields a lower bound on algorithmic complexity (circuit lower bounds)? The various results in this area all follow the same general template (see, e.g., [3] for a formal treatment): We assume that we have both a circuit upper bound for some large class such as NEXP, and a general derandomization technique, and get a contradiction as follows:

Simulation: Meyer, Karp and Lipton [17] introduced techniques to give consequences of non-uniform (circuit) upper bounds for uniform classes. Using interactive proofs [6, 23], many of these can be extended to show that if a large class C such as EXP or PSPACE has small circuits, then C also has short (constant-round) interactive proofs (see, e.g., [5]).

Derandomization: Invoking the generic derandomization technique in the above simulation, we get that C can be simulated only with non-determinism.

Contradict a known lower bound: If C is NEXP itself, as in [11] and [27], the contradiction comes from some version of the non-deterministic time hierarchy theorem. An alternate method used by [1] is to pad up the non-trivial simulation of C in small non-deterministic time to show that some analogous class superpoly-C can be simulated in NEXP. The class superpoly-C will often be strong enough that a lower bound in that class can be shown by direct diagonalization (see, e.g., [16]). By simulation, NEXP will inherit the same lower bound.

result for finite fields: their arithmetic-complexity hardness assumption is not for a polynomial (defined over all extension fields), but rather for the function that *agrees* with the polynomial over a *particular* extension field.

[2] [13] has an alternative method of getting an equivalence for the case of PIT over the *integers*, but at the expense of moving to versions of the classes with *advice*.

If both the circuit upper bound assumed and the derandomization assumed are worst-case, then the above template can be filled out to get a variety of trade-offs. However, when one or the other is only assumed to occur for infinitely many lengths of input, care needs to be taken to compare the input sizes where the simulation is possible, the corresponding lengths where derandomization is possible, and what non-worst-case versions of the known lower bounds are true.

While the general issue of connections between circuit complexity and derandomization has been the subject of intense research by many people, the papers most directly parallel to this work are [11, 14, 1, 13, 18]. Here, we briefly describe the main results and techniques of these five papers, and compare them to our results and techniques.

Promise-BPP. [11] considers the question of derandomizing promise-BPP. While there are several other results in the paper, the main result for our purposes is an equivalence between a circuit lower bound for NEXP and a "non-trivial" simulation of promise-BPP.

▶ **Theorem 1** ([11]). NEXP $\not\subset$ P/poly *if and only if* promise-BPP $\subseteq \cap_{\epsilon>0}$io-NTIME$[2^{n^\epsilon}]/n^\epsilon$.

While this is a strong result, and in fact an equivalence, there are some questions raised by the details in this statement. Since promise-BPP $\subseteq$ P/poly, we know that advice can be powerful in this context. How significant is the small amount of advice allowed in the sub-exponential time algorithms for promise-BPP? Is it really necessary or just a by-product of the proof? Can we get a stronger conclusion if we assume a Circuit Acceptance Probability Problem (estimating the acceptance probability of a size-n Boolean circuit to within an additive error of at most $1/n$; *CAPP*) algorithm with no advice? Also, [11] use the "Easy Witness Lemma" to obtain their result, which is itself derived by a somewhat convoluted indirect argument. Is this use of the Easy Witness Lemma necessary?

Here, we give some answers to these questions. Using arguments parallel to those in [1, 18] with respect to PIT, we remove the Easy Witness Lemma. Then (as [1] did for PIT) we show that *sufficiently strong nondeterministic algorithms for* promise-BPP *(without advice) imply the stronger conclusion that* (NEXP $\cap$ coNEXP) $\not\subseteq$ P/poly (Theorem 5). So there does seem to be a real difference between derandomizations with and without advice.

One use of advice in the original argument is that, if we have a different CAPP algorithm for every $\epsilon > 0$, then, for each length, the best value of ϵ and the algorithm that achieves that ϵ can be both given as advice. To remove this advice, we need to have a *single* algorithm that runs in nondeterministic sub-exponential time, $2^{n^{\epsilon(n)}}$ for some *computable* $\epsilon(n) \in o(1)$. This seems an equally natural definition of a problem being computable in sub-exponential time, and we adopt it throughout our paper. We prove some closure properties and normal forms for this notion that might be of independent interest.

Circuit lower bounds and nontrivial useful properties. Razborov and Rudich [20] defined an NP-constructive property useful against P/poly to be an NP-computable predicate on 2^n-bit inputs (the truth tables of n-variate Boolean functions) such that, whenever the predicate holds for infinitely many input lengths of a given family of Boolean functions $f = \{f_n\}_{n\geq 0}$, it follows that $f \notin$ P/poly. Call such a property *nontrivial* if there are infinitely many input lengths n such that at least one truth table of size 2^n satisfies the property.

It was shown in [11] that the existence of a nontrivial NP-constructive property useful against P/poly implies that NEXP $\not\subseteq$ P/poly. We strengthen this to the circuit lower bound for NEXP $\cap$ coNEXP for nontrivial NP-constructive properties useful against computably superpolynomial circuit size. Moreover, we make this connection into an equivalence, using a variant of the "robustly often" notion of [9] (Theorem 7).

Polynomial identity testing over the rationals. [14] extends the connections between derandomization and circuit lower bounds to the arithmetic-complexity setting. In one direction, they show that if both $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ and the permanent function has polynomial-size arithmetic circuits over the rationals, then no nondeterministic sub-exponential time algorithm can solve PIT, even for infinitely many input sizes. In the other direction, either $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ or the permanent requiring super-polynomial sized circuits gives some nontrivial algorithm for PIT, although not quite a direct converse.

This again raises some questions. Stating the result in the contrapositive, a nontrivial algorithm for PIT would yield either a Boolean circuit lower bound or an arithmetic circuit lower bound. Intuitively, Boolean circuit lower bounds should be more difficult, so can we get a similar result that only mentions arithmetic circuit lower bounds?

This question was addressed in [1, 18] and [13]. First, [1] and [18] give an alternate version of the final contradiction step that allows them to avoid the Easy Witness Lemma. This not only simplified the proof, but allowed them to replace the condition that $\mathsf{NEXP} \subset \mathsf{P/poly}$ with the weaker condition that $(\mathsf{NEXP} \cap \mathsf{coNEXP}) \subset \mathsf{P/poly}$. ([14] had also shown that $(\mathsf{NEXP} \cap \mathsf{coNEXP}) \subset \mathsf{P/poly}$ and the easiness of the permanent sufficed to show $\mathsf{PIT} \not\in \mathsf{NP}$.) [13] show that the Boolean and arithmetic lower bounds can be in some sense combined. They show that derandomizing a certain version of PIT, low-PIT, in non-deterministic sub-exponential time is equivalent to proving a super-polynomial lower bound for circuits with restricted degrees on a multi-linear function whose values can be computed in $\mathsf{NEXP}/O(n)$. However, being determined by the correct advice, the functions where the arithmetic lower bound would hold in their result are somewhat non-constructive, and when the advice is incorrect, the corresponding functions might not be defined at all, so they cannot be combined into one universal function for each length.

We show that by using the [1, 18] type contradiction step, the two lower bounds can be replaced by a *single arithmetic circuit lower bound for a multilinear polynomial definable in* $\mathsf{NEXP} \cap \mathsf{coNEXP}$, thus removing this somewhat awkward non-uniformity.

Polynomial identity testing over a fixed finite field. The polynomial identity testing problem and arithmetic circuit complexity are also important over other fields, such as fixed finite fields. Here one fixes some constant-size finite field $\mathbb{F}$, and then asks for an efficient algorithm to decide if a given arithmetic circuit C over $\mathbb{F}$, defining a low-degree polynomial p (over $\mathbb{F}$ and any finite field extending $\mathbb{F}$), is such that $p \equiv 0$. The latter can be easily decided by a randomized polynomial-time Schwartz-Zippel algorithm evaluating p on random points from a sufficiently large (larger than the degree of p) extension field of $\mathbb{F}$.

Do similar connections between hardness and pseudorandomness hold for fixed finite fields? Before the work here, almost no such connections were known.

There are a number of obstacles to directly translating the [14] techniques to fixed finite fields. First, in the "derandomization of PIT to arithmetic hardness" direction, [14] use the clean algebraic recursive "expansion by minors" definition of the permanent, a problem complete for #P. While the same recursion holds for the permanent over finite fields, permanent over finite fields is not known to be #P complete, or even NP-hard (under deterministic reductions).

Secondly, in the "arithmetic hardness to derandomization of PIT" direction, if the hitting set generator from [14] fails using a given f as the hard function over a finite field, it only follows that some power of f had a small arithmetic circuit, not the f itself. Though this power of f can be manipulated to obtain an arithmetic circuit that agrees with f over some particular extension field, the polynomial computed by this circuit would not be identically equivalent to f (see Section 2.5 for the difference between testing for identity and agreement of polynomials over a finite field). Therefore, a power of f could still have small arithmetic circuits and this would not contradict the hardness assumption used by [14] in the hardness to randomness direction. Thus, [14] has only a weak hardness-to-randomness result in the case of circuits over finite fields.

We give the first proof that *nondeterministic polynomial identity testing over a fixed finite field yields an arithmetic circuit lower bound* (Theorem 2). We avoid the permanent by showing a simulation lemma for an even larger class, PSPACE, by using a version of a PSPACE-complete function of [25] with a similar algebraic recursive definition. We observe that, not only would a small circuit for this function itself yield the required simulation, but also any circuit for a small power of the function would have a similar consequence. This matches the type of arithmetic lower bounds needed to derandomize polynomial identity testing using the hitting set generator of [14]. Thus, the correct connection is between derandomizing PIT and proving lower bounds on circuits that compute powers of polynomials.

Equivalence between circuit lower bounds and derandomization. While the results above come closer to showing that each of these three derandomization problems are equivalent to a corresponding lower bound, they are not literal equivalences. This is because of the distinction between infinitely often computation and worst-case computation. Making worst-case assumptions about algorithms is necessary to get even infinitely often hardness, but infinitely often hardness only suffices to get algorithms that work infinitely often.

To make versions of our results that are literal equivalences, we consider an intermediate notion between worst-case and infinitely-often computation, robustly-often computation introduced by [13]. Informally, an algorithm solves a given problem *robustly often* if there are infinitely many *intervals* of superpolynomially many input lengths where the algorithm is correct on each length in the interval.

While this, somewhat less standard, notion of inclusion (as well as the related notion of separation) is exactly what we need to make the connections between circuit lower bounds and derandomization into equivalences (Theorems 3, 4, and 6), we feel that this notion is quite natural and can be justified, e.g., by the following considerations. As technology improves, in general, all computational elements will improve somewhat. But computational elements will often get more diverse, so that say, the CPU in cell phones will not be improving its computational power as quickly as the top super-computer will. A reasonable model is to assume that there are two different Moore's laws, one for "high-end" technology and one for "low-end", with different periods of doubling. Thus, high-end and low-end technologies will be polynomially related in their resources, and the intermediate technologies will span the range between them. Thus, it is natural to look at polynomially related ranges of parameters, not just single values. Robust computation does just that, looking at whether computation is possible not just on infinitely many single input lengths, but whether there are infinitely many "technology levels" of unbounded polynomial reach where this computation is possible.

Remainder of the paper. The definitions are in Section 2. We formally state our main results with proof sketches in Section 3, see the full version for complete proofs.

2 Definitions

2.1 Arithmetic circuit complexity classes

For a finite field $\mathbb{F}$, define $\mathsf{ASize}_{\mathbb{F}}[s(n)]$ to be the class of all families of n-variate polynomials $\{f_n\}$ over $\mathbb{F}$ such that, for all sufficiently large n, the polynomial f_n is computed by some arithmetic circuit C_n of size at most $s(n)$ over $\mathbb{F}$ (i.e., the polynomials $f_n(x_1, \ldots, x_n)$ and $C_n(x_1, \ldots, x_n)$ are formally the same when viewed as the sums of monomials). Over the integers, the class $\mathsf{ASize}_{\mathbb{Z}}[s(n)]$ is defined analogously.

We denote by $\mathsf{ASizeDeg}_{\mathbb{F}}[s(n)]$ the subclass of $\mathsf{ASize}_{\mathbb{F}}[s(n)]$ of families of n-variate polynomials f_n that have degree at most $s(n)$. Over the integers, we denote by $\mathsf{ASizeDeg}_{\mathbb{Z}}[s(n)]$ the restricted class of polynomial families $\{f_n\}$ that have *formal degree*[3] at most $s(n)$.

"Easy" polynomials. Over the integers, we shall consider a polynomial f "easy" if some constant multiple $c \cdot f$ is computable by a "small" arithmetic circuit. More formally, we define $\mathsf{ASizeDegMul}[s(n)]$ as the class of polynomial families $\{f_n\}$ over $\mathbb{Z}$ of degree at most $s(n)$ such that, for some function $g : \mathbb{N} \to \mathbb{N}$ we have $\{g(n) \times f_n\} \in \mathsf{ASizeDeg}_{\mathbb{Z}}[s(n)]$ and g is computed by a family of variable-free $\mathsf{ASizeDeg}_{\mathbb{Z}}[s(n)]$ circuits. Thus, for each input length n, we could compute a *different* constant multiple of f. $\mathsf{ASizeDegMul}$ is equivalent to the class $ASIZEDEG'$ of [13].

Over finite fields, we shall consider a polynomial f "easy" if some bounded power f^d of f is computable by a "small" arithmetic circuit. More formally, over a finite field $\mathbb{F}$, we define the class $\mathsf{ASizeDegPow}_{\mathbb{F}}[s(n)]$ as the class of polynomial families $\{f_n\}$ over $\mathbb{F}$ of degree at most $s(n)$ such that, for some function $d : \mathbb{N} \to \mathbb{N}$ with $d(n) \leq s(n)$, we have $\{f_n^{d(n)}\} \in \mathsf{ASizeDeg}_{\mathbb{F}}[s(n)]$.

2.2 Polynomials computable in NE: The class ml-NE

Fix an arbitrary finite field $\mathbb{F} = \mathbb{F}_{p^k}$ of characteristic p. Let $f = \{f_n(x_1, \ldots, x_n)\}_{n\geq 0}$ be an arbitrary family of polynomials over $\mathbb{F}$. For a polynomial f and a monomial m, denote by $\mathsf{coeff}(f, m)$ the coefficient of f at the monomial m (when f is written out as the sum of its monomials). Define the language$\textsc{monomial}(f) = \{(a_1, \ldots, a_n, c) \mid a_1, \ldots, a_n \in \{0, 1\},\ c \in \mathbb{F},\ \mathsf{coeff}(f_n, x_1^{a_1} \ldots x_n^{a_n}) = c\}$. We denote by $\textsc{monomial}(f_n)$ the restriction of $\textsc{monomial}(f)$ to the case of f_n, i.e., $\textsc{monomial}(f_n)$ defines the coefficients of the n-variate polynomial f_n. For multilinear polynomial families f over the integers, the language $\textsc{monomial}(f)$ is defined similarly, with the natural change that the coefficient c be an integer in binary.

We say that a family of multilinear polynomials $f = \{f_n\}_{n\geq 0}$ over a finite field $\mathbb{F}$ or over integers $\mathbb{Z}$ is in the class $\mathsf{ml\text{-}NE}$ if $\textsc{monomial}(f) \in \mathsf{NE}$.

Observe that if $\textsc{monomial}(f) \in \mathsf{NE}$, then $\textsc{monomial}(f) \in \mathsf{coNE}$ also. Thus, we have $\textsc{monomial}(f) \in \mathsf{NE} \iff \textsc{monomial}(f) \in (\mathsf{NE}\cap\mathsf{coNE})$, and so $\mathsf{ml\text{-}NE} = \mathsf{ml\text{-}(NE\cap coNE)}$; the same equivalence holds also for the definition of $\mathsf{ml\text{-}NE}$ used by Jansen and Santhanam [13][4].

2.3 Computably subexponential and superpolynomial bounded classes

We call a function $\alpha\colon \mathbb{N} \to \mathbb{R}^{\geq 0}$ *computably super-constant*, denoted by $\alpha(n) \in \omega_c(1)$, if there exists computable, monotone non-decreasing function $\alpha'\colon \mathbb{N} \to \mathbb{R}^{\geq 0}$ with $\alpha'(n) \to \infty$ such that, for all sufficiently large $n \in \mathbb{N}$, $\alpha(n) \geq \alpha'(n)$. Without loss of generality, we may always assume that our computably super-constant functions $\alpha(n)$ are computable in time $\mathsf{poly}(n)$ (see full version for details).

[3] For input gates, regardless of their label, the formal degree is 1; an addition gate has the formal degree equal to the maximum of the formal degree of its input gates; a multiplication gate has the formal degree equal to the sum of the formal degrees of its input gates.

[4] The graph definition of $\mathsf{ml\text{-}NE}$ used by [13] and the monomial-coefficient definition of $\mathsf{ml\text{-}NE}$ given here are equivalent because of efficient interpolation. We use the coefficient definition in our work just for convenience, as it makes it obvious that one can evaluate polynomials over extensions of finite fields.

The standard definition of NSUBEXP is $\bigcap_{\epsilon>0} \mathsf{NTIME}[2^{n^{\epsilon}}]$. This is problematic when we need to run a padding argument for some language in this class, because the amount of padding required will depend on ϵ. There is no uniform way to produce the specific ϵ required for a particular amount of padding, so any padding argument must rely on advice to use the correct value of ϵ. We use our notion of computably super-constant functions to trade a more restricted class for freedom from advice, and define computably subexponential-time classes as follows: We say that a language L is in nondeterministic computably subexponential time, denoted $L \in \mathsf{NSUBEXP}_c$, if $L \in \mathsf{NTIME}[2^{n^{1/\alpha(n)}}]$ for some $\alpha(n) \in \omega_c(1)$. We denote by $\mathsf{superpoly}_c$ any function $n^{\alpha(n)}$ for some $\alpha(n) \in \omega_c(1)$.

2.4 Robustness

To establish equivalences between circuit lower bounds and derandomization, we need notions that are intermediate between infinitely-often and almost-everywhere. In the spirit of [9], we define "robust" notions of containment and separation for complexity classes. Our robust ranges are some fixed *computably super-polynomial* function in length, whereas the ranges of [9] are for every fixed polynomial function. The reason we require this alternative notion is because the simulation steps of our arguments use superpolynomial amounts of padding. The [9] definition handles fixed polynomial-time translations or translations with fixed polynomial advice, but not superpolynomial translations. Our definition is an attempt to make the minimal extention possible to [9] that can handle superpolynomial translations. Thus we do not expect our results to hold under the [9] robustness notions, without major technical innovation in hardness-randomness tradeoffs that dispenses with the necessity for superpolynomial padding.

For functions $l, r : \mathbb{N} \to \mathbb{N}$, called left and right "interval functions", define the (l, r)*-core* of a set $S \subseteq \mathbb{N}$ to be the set of intervals $[l(m), r(m)]$ that are entirely contained in S, i.e., $\mathsf{core}(S) = \{m \in \mathbb{N} \mid \forall n \in \mathbb{N}\ (l(m) \leq n \leq r(m) \implies n \in S)\}$. A set S is called (l, r)-*robust* if the (l, r)-core of S is infinite. Finally, we say that S is *computably robust* if S is $(m^{1/\alpha(m)}, m^{\alpha(m)})$-robust for some $\alpha \in \omega_c(1)$. For brevity, we shall call such a set S simply *α-robust.*

Robust inclusions. For a language L and a complexity class $\mathcal{C}$, we say that L is *uniform robustly often in* $\mathcal{C}$, denoted $L \in \mathsf{ro}_\star\text{-}\mathcal{C}$, if there is a language $N \in \mathcal{C}$ such that the set $S = \{n \in \mathbb{N} \mid L_n = N_n\}$ is computably robust, where $L_n = L \cap \{0, 1\}^n$ is the nth slice of L. Our definition is "uniform" compared to the notion of [9] because the notion defined there has interval lengths that are defined by every fixed polynomial function – this is an infinite (but very regular) set of functions giving interval lengths. Our robust sets have a fixed pair of interval functions.

We say that a family $f = \{f_n\}$ of multilinear polynomials (over a finite field $\mathbb{F}$ or over integers $\mathbb{Z}$) is in $\mathsf{ro}_\star$-ml-NE, *robustly often* ml-NE, if, for some NE machine M, the set $S = \{n \in \mathbb{N} \mid M$ correctly decides $\textsc{monomial}(f_n)\}$ is computably robust.

Robust promise classes. For a language L and a *semantic* complexity $\mathcal{C}$, we say that L is in *uniform robustly promise* $\mathcal{C}$, denoted $L \in \mathsf{rp}_\star\text{-}\mathcal{C}$, if there is a Turing machine M such that

$$S = \{n \in \mathbb{N} \mid \text{ for all } x \in \{0,1\}^n,\ M(x) \text{ is a } \mathcal{C}\text{-type machine and } M(x) \text{ decides if } x \in L_n\}$$

is computably robust.

▶ Remark. In general, $\mathsf{ro}_\star$-$\mathcal{C}$ and $\mathsf{rp}_\star$-$\mathcal{C}$ are different for a semantic class $\mathcal{C}$. For example, $L \in \mathsf{ro}_\star$-$(\mathsf{NE} \cap \mathsf{coNE})$ if there is a language $N \in (\mathsf{NE} \cap \mathsf{coNE})$ that robustly often agrees with L. That is, there is a pair of NE machines M_1 and M_2 such that, for every $x \in \{0,1\}^n$, exactly one of M_1 and M_2 accepts x, and $S = \{n \in \mathbb{N} \mid M_1$ decides L_n and M_2 decides $\overline{L_n}\}$ is computably robust, where $\overline{L_n}$ is the complement of L_n.

In contrast, $L \in \mathsf{rp}_\star$-$(\mathsf{NE} \cap \mathsf{coNE})$ if there is a pair of NE machines M_1 and M_2 such that $S = \{n \in \mathbb{N} \mid M_1$ decides L_n and M_2 decides $\overline{L_n}\}$ is computably robust. Note that, in the second case, the machines M_1 and M_2 may not be "complementary" on the input lengths outside of S, and so we may not have any language $N \in (\mathsf{NE} \cap \mathsf{coNE})$ that agrees with L robustly often: in the $\mathsf{rp}_\star$- case, the promise is not required to hold for slices outside the robust set.

Significant separations. To complement the inclusion types above, we define *uniform significant separations* denoted $\mathsf{ro}_\star$-$\mathcal{C} \not\subseteq \mathsf{SIZE}[s(n)]$. We write $\mathsf{ro}_\star$-$\mathcal{C} \not\subseteq \mathsf{SIZE}[s(n)]$ if there is a language $L \in \mathsf{ro}_\star$-$\mathcal{C}$ over computably robust set S such that L_n cannot be computed by a circuit of size $s(n)$ for infinitely many values $n \in \mathsf{core}(S)$.

Intuitively, a uniform significant separation means that we can always locate hard lengths for the separated class in the "middle" of large, computably robust intervals. This is different from the [9] notion, which says (intuitively) that hard lengths are never too far apart. Under our definition, if the robust set comes with a promise, this means that hard lengths are located in the center of large ranges where the promise holds. This is what our arguments for equivalence will hinge on. The generalization of this definition to arithmetic circuits and uniform robust promise classes is obvious. For example, we define two uniform significant separations below.

We say that $\mathsf{ro}_\star$-ml-$\mathsf{NE} \not\subseteq \mathsf{ASize}[s(n)]$ if there is a polynomial family $f = \{f_n\} \in \mathsf{ro}_\star$-$\mathsf{ml}$-$\mathsf{NE}$, with MONOMIAL$(f)$ correctly decided by some NE machine on a computably robust set S, such that f_n cannot be computed by an arithmetic circuit of size $s(n)$ for infinitely many values $n \in \mathsf{core}(S)$. The case of other arithmetic circuit classes ($\mathsf{ASizeDeg}$ and $\mathsf{ASizeDegPow}$) is similar.

We say that $\mathsf{rp}_\star$-$(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[s(n)]$ if there is a promise problem $L \in \mathsf{rp}_\star$-$(\mathsf{NE} \cap \mathsf{coNE})$, with a pair of NE machines correctly deciding L and $\bar{L}$ over a computably robust set S of input lengths, such that L_n cannot be computed by a Boolean circuit of size $s(n)$ for infinitely many input lengths $n \in \mathsf{core}(S)$.

2.5 Derandomization of Polynomial Identity Testing

Let $\mathbb{F}$ be any finite field. For $D \in \{\mathbb{Z}, \mathbb{F}\}$, the Polynomial Identity Testing over D, denoted PIT_D, is the task to decide if a given arithmetic circuit C over D computes the identically zero polynomial (when the polynomial C is expanded as the sum of monomials[5]). The low-PIT_D variant is restricted to test only circuits that have degree (or, in the case of $\mathbb{Z}$, formal degree) less than their size.

A *hitting set generator* for PIT_D is a function $\mathcal{H}$, that on input 1^n, outputs a collection of t tuples $a_1, \dots, a_t \in D^n$ such that, for every arithmetic circuit $C(x_1, \dots, x_n)$ over D of

[5] Stating this definition using sums-of-monomials is crucial, because over a finite field $\mathbb{F}$, testing if a polynomial *agrees with* the zero polynomial for all inputs over $\mathbb{F}$ is actually coNP-complete. The sum-of-monomials definition that we use puts $\mathsf{PIT}_\mathbb{F} \in \mathsf{BPP}$, and thus it is meaningful to ask for a derandomization of this problem.

size at most n, if $C \not\equiv 0$, then there is at least one $i \in [t]$ where $C(a_i) \neq 0$. In case of a finite field $\mathbb{F}$, we allow the tuples $a_1, \ldots, a_t$ to come from $(\mathbb{F}')^n$ for some extension field $\mathbb{F}'$ of $\mathbb{F}$.

It was shown by [14] that a multilinear polynomial $f \notin \mathsf{ASizeDegPow}_{\mathbb{F}}[\mathsf{superpoly}_c]$ can be used to derandomize $\mathsf{low\text{-}PIT}_{\mathbb{F}}$ in subexponential time (by constructing a hitting set).

2.6 Derandomization of promise-BPP

Derandomizing promise-BPP is equivalent to getting an efficient algorithm for estimating the acceptance probability of a given Boolean circuit $C(x_1, \ldots, x_n)$ of size n to within an additive error $1/n$; the latter problem is called Circuit Acceptance Probability Problem (CAPP). We use the notation $\mathsf{promise\text{-}BPP} \subseteq \mathsf{NSUBEXP}_c$ to mean that there is a nondeterministic Turing machine that runs in computably subexponential time, and solves CAPP with an additive approximation error $1/n$. That is, on a given circuit $C(x_1, \ldots, x_n)$, the nondeterministic machine has at least one accepting computation, and every accepting computation yields a value r such that $|\mathbf{Pr}_{a \in \{0,1\}^n}[C(a) = 1] - r| \leq 1/n$.

We say that $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$ if a $\mathsf{NSUBEXP}_c$ algorithm correctly approximates CAPP only robustly often.

A collection $a_1, \ldots, a_t \in \{0,1\}^n$ is a *discrepancy set* of size t for circuits of size n if, for every Boolean circuit C of size n on n inputs, $\left|\mathbf{Pr}_{z \in \{0,1\}^n}[C(z) = 1] - \mathbf{Pr}_{i \in [t]}[C(a_i) = 1]\right| \leq 1/n$. It was shown by [5] that the truth table of a superpolynomial circuit-complexity Boolean function can be used to get a subexponential-size discrepancy set (in subexponential time).

3 Our results

3.1 PIT vs. arithmetic circuit lower bounds

▶ **Theorem 2.** *Fix an arbitrary finite field* $\mathbb{F}$*. We have*
1. $\mathsf{low\text{-}PIT} \in \mathsf{NSUBEXP}_c \;\Rightarrow\; \mathsf{ml\text{-}NE} \not\subseteq \mathsf{ASizeDegPow}[\mathsf{superpoly}_c]$.
2. $\mathsf{ml\text{-}NE} \not\subseteq \mathsf{ASizeDegPow}[\mathsf{superpoly}_c] \;\Rightarrow\; \mathsf{low\text{-}PIT} \in \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$.

Proof sketch. (1) Assume a nondeterministic subexponential-time algorithm for low-PIT, but that ml-NE has "small" arithmetic circuits. We arithmetize TQBF to get a PSPACE-complete multilinear polynomial $f = \{f_n\}$ over $\mathbb{F}$. This polynomial f turns out to be computable in ml-NE, and so, by our assumption, some powers $f_n^{d_n}$, for "small" d_n, have small arithmetic circuits over $\mathbb{F}$.

For each n, we nondeterministically guess a small circuit and a small d_n. Using the ideas of the $\mathsf{PSPACE} = \mathsf{IP}$ proof, we then verify that the guessed circuit computes the polynomial $f_n^{d_n}$. This verification algorithm uses our assumed low-PIT algorithm, and runs in $\mathsf{NSUBEXP}_c$.

Computing powers $f_n^{d_n}$ is still PSPACE-complete, and so we get $\mathsf{PSPACE} \subseteq \mathsf{NSUBEXP}_c$. By padding, we obtain $\mathsf{SPACE}[\mathsf{superpoly}_c] \subseteq \mathsf{NE}$. By diagonalization, $\mathsf{SPACE}[\mathsf{superpoly}_c]$ contains a language L of some computably superpolynomial Boolean circuit complexity, almost everywhere. It follows that the multilinear extension of L over $\mathbb{F}$ requires arithmetic circuits of computably superpolynomial size, almost everywhere. On the other hand, each coefficient of this multilinear polynomial is computable in $\mathsf{SPACE}[\mathsf{superpoly}_c]$, and hence in NE.

(2) Assume that some family $g = \{g_n\}$ of polynomials in ml-NE is such that all powers $g_n^{d_n}$, for "small" d_n, require superpolynomial arithmetic circuit complexity for infinitely many input lengths n. By [14], we get that low-PIT is in $\mathsf{NSUBEXP}_c$ infinitely often. The input

lengths where low-PIT is easy (derandomized) correspond to the (smaller) input lengths where the polynomials g_n are actually hard.

To improve this "infinitely often" result to the "robustly often" one, we do the following: when asked to derandomize low-PIT for a certain input length n, we go to the related smaller length n', and consider polynomials over a superpolynomial interval of input lengths above n' as candidate hard polynomials; we use each such polynomial to construct a candidate hitting set by [14]. If the given interval above n' contains a length m such that g_m is hard, then our derandomization succeeds. Since there infinitely many intervals containing a length m where g_m is hard, there will be infinitely many intervals of superpolynomial length where our low-PIT algorithm is correct. ◀

▶ **Theorem 3.** *Fix an arbitrary finite field $\mathbb{F}$. We have*

$$\mathsf{low\text{-}PIT} \in \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c \;\Leftrightarrow\; \mathsf{rp}_\star\text{-}\mathsf{ml\text{-}NE} \not\subseteq \mathsf{ASizeDegPow}[\mathsf{superpoly}_c].$$

Proof sketch. ($\Rightarrow$) We start as in the proof of Theorem 2, implication (1). We get a PSPACE-complete multilinear polynomial $f = \{f_n\}$ over $\mathbb{F}$ such that some powers $f_n^{d_n}$ are computable by small arithmetic circuits, for almost all input lengths n. Since our low-PIT algorithm is correct for infinitely many superpolynomial intervals of input lengths, we can guarantee the successful verification of an arithmetic circuit for $f_n^{d_n}$ for the corresponding superpolynomial intervals of input lengths only. This yields PSPACE $\subseteq$ $\mathsf{ro}_\star$-NSUBEXP$_c$, and by padding, SPACE[superpoly$_c$] $\subseteq$ $\mathsf{ro}_\star$-NE. Finally, by diagonalization and multilinear extension, we get a family of multilinear polynomals g_n over $\mathbb{F}$ that require computably superpolynomial arithmetic circuit complexity almost everywhere, and yet we can compute the coefficients of g_n in NE for infinitely many superpolynomial intervals of input sizes n.

($\Leftarrow$) As in the proof of Theorem 2, implication (2), we will use hard polynomials to derandomize low-PIT by [14]. The difference now is that a given NE machine computes a valid polynomial only over some computably robust set S of input lengths n, and that this polynomial is hard only for infinitely many lengths $n \in \mathsf{core}(S)$. Still we can use this NE machine to construct a candidate hitting set for a given low-PIT instance so that, for infinitely many lengths $n \in \mathsf{core}(S)$, we get a correct hitting set, and so low-PIT $\subseteq$ io-NSUBEXP$_c$. To boost this to the robustly often inclusion, we employ a similar trick as before: use a superpolynomial-size interval of input lengths to get a collection of candidate hitting sets, and take their union. When all input lengths fall into an interval where our NE machine computes a valid polynomial, we get that the union of such candidate hitting sets is well-defined. If, in addition, the polynomial computed by our NE machine is actually hard on some length in this interval, then we get a correct hitting set. ◀

We have analogous results also for the case of integers $\mathbb{Z}$, with analogous proofs (using the analysis of [13] showing that Kaltofen's [15] polynomial factorization algorithm over integers respects formal degree). In particular, we have the following equivalence.

▶ **Theorem 4.** *Over $\mathbb{Z}$,*

$$\mathsf{low\text{-}PIT} \in \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c \;\Leftrightarrow\; \mathsf{rp}_\star\text{-}\mathsf{ml\text{-}NE} \not\subseteq \mathsf{ASizeDegMul}[\mathsf{superpoly}_c].$$

3.2 Promise-BPP vs. Boolean circuit lower bounds

▶ **Theorem 5.** *We have*

1. promise-BPP $\subseteq$ NSUBEXP$_c$ $\Rightarrow$ (NE $\cap$ coNE) $\not\subseteq$ SIZE[superpoly$_c$].
2. (NE $\cap$ coNE) $\not\subseteq$ SIZE[superpoly$_c$] $\Rightarrow$ promise-BPP $\subseteq$ $\mathsf{ro}_\star$-NSUBEXP$_c$.

Proof sketch. (1) Assume that CAPP is correctly approximated by an $\mathsf{NSUBEXP}_c$ algorithm, but every language in $(\mathsf{NE} \cap \mathsf{coNE})$ has small superpolynomial-size Boolean circuits. The latter means that $\mathsf{E} \subseteq \mathsf{SIZE}[\mathsf{superpoly}_c]$, and hence, by [4], that $\mathsf{E} \subseteq \mathsf{MA}[\mathsf{superpoly}_c]$, for a related small superpolynomial time complexity. Using our CAPP algorithm, we get $\mathsf{MA}[\mathsf{superpoly}_c] \subseteq \mathsf{NSUBEXP}_c$. By padding the inclusion $\mathsf{E} \subseteq \mathsf{NSUBEXP}_c$, we get that $\mathsf{TIME}[2^{\mathsf{superpoly}_c}] \subseteq \mathsf{NE}$. Finally, by diagonalization, we get a language $L \in \mathsf{TIME}[2^{\mathsf{superpoly}_c}] \subseteq \mathsf{NE}$ that requires computably superpolynomial circuit complexity (almost everywhere). Since $\bar{L} \in \mathsf{TIME}[2^{\mathsf{superpoly}_c}] \subseteq \mathsf{NE}$, the conclusion follows.

(2) Assume we have a pair of NE machines that compute L and $\bar{L}$ for a language L requiring $\mathsf{superpoly}_c$-size circuits infinitely often. By the hardness-randomness tradeoff of [5], we get a $\mathsf{NSUBEXP}_c$ algorithm that correctly approximates CAPP infinitely often; the input lengths where the CAPP algorithm is correct correspond to the (smaller) input lengths where the language L is actually hard. To boost this to the desired $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$ inclusion, we use the same "interval trick" as in the arithmetic case (Theorem 2, (2)), to show that there is a $\mathsf{NSUBEXP}_c$ algorithm that, robustly often, produces a discrepancy set for circuits of size n. ◀

We extend the two implications of Theorem 5 to the equivalence, by carefully adapting the arguments above to the setting of robust inclusions and separations.

▶ **Theorem 6.** $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c \;\Leftrightarrow\; \mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c]$.

3.3 Robustly-often nontrivial useful properties

A *property* of Boolean functions is a family $\mathcal{P} = \{\mathcal{P}_n\}_{n \geq 0}$ of predicates $\mathcal{P}_n : \{0,1\}^{2^n} \to \{0,1\}$. For a function $s : \mathbb{N} \to \mathbb{N}$, we say that a property $\mathcal{P}$ is *useful against* $\mathsf{SIZE}[s]$ *at length* n if, whenever $\mathcal{P}_n$ accepts the truth table of a Boolean n-variate function $f_n : \{0,1\}^n \to \{0,1\}$, this means that f_n requires circuit size at least $s(n)$. We say that $\mathcal{P}$ is *nontrivial at length* n if $\mathcal{P}_n$ accepts at least one truth table of length 2^n. Finally, we say that $\mathcal{P}$ is *robustly-often nontrivially useful against* $\mathsf{SIZE}[\mathsf{superpoly}_c]$ *(denoted* $\mathsf{ro}_\star$*-useful)* if, for some $s(n) \in \mathsf{superpoly}_c$,

1. $S = \{n \in \mathbb{N} \mid \mathcal{P} \text{ is nontrivial at length } n\}$ is computably robust, and
2. $\mathcal{P}$ is useful against $\mathsf{SIZE}[s]$ at length n for infinitely many lengths $n \in \mathsf{core}(S)$.

As a corollary of Theorem 6, we get the following equivalence between circuit lower bounds for $\mathsf{NEXP} \cap \mathsf{coNEXP}$ and the existence of $\mathsf{ro}_\star$-useful properties.

▶ **Theorem 7.** *The following are equivalent:*

- $\mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c] \iff$
- *there is a* NP*-computable* $\mathsf{ro}_\star$*-useful property*
- *there is a* P*-computable* $\mathsf{ro}_\star$*-useful property*

Proof. ($\Rightarrow$) Assume we have a pair of NE machines M_1 and M_0 that correctly compute L and $\bar{L}$, respectively, for some computably robust set S of input lengths, where L requires $\mathsf{superpoly}_c$ circuit size for infinitely many input lengths in $\mathsf{core}(S)$. Define a property $\mathcal{P}$ as follows:

"On input $T \in \{0,1\}^{2^n}$, guess 2^n candidate witnesses $a_1, \ldots, a_{2^n}$ of length $2^{O(n)}$ each, and check, for every $1 \leq i \leq 2^n$, if $T_i = b$, for $b \in \{0,1\}$, then $M_b(i)$ accepts a_i as a witness. If succeed for all i's, then accept. Otherwise, reject."

Clearly, $\mathcal{P}$ is NP-computable. Note that $\mathcal{P}_n$ accepts the truth table of L_n (and nothing else) for the computably robust set S of input lengths n, and so $\mathcal{P}$ is robustly-often nontrivial. Finally, since L requires circuit size $\mathsf{superpoly}_c$ for infinitely many input lengths $n \in \mathsf{core}(S)$, we get that $\mathcal{P}$ is useful against $\mathsf{SIZE}[\mathsf{superpoly}_c]$ at length n for infinitely many $n \in \mathsf{core}(S)$. Thus $\mathcal{P}$ is $\mathsf{ro}_\star$-useful.

($\Leftarrow$) Given an NP-computable property $\mathcal{P}$ that is nontrivial on a computably robust set S of input lengths n, and useful against $\mathsf{SIZE}[\mathsf{superpoly}_c]$ for infinitely many $n \in \mathsf{core}(S)$, we use $\mathcal{P}$ to nondeterministically guess a truth table T of length 2^n, nondeterministically verify that T is accepted by $\mathcal{P}_n$, and use T as a "hard" Boolean function to derandomize $\mathsf{promise\text{-}BPP}$ (using the hardness-randomness tradeoff of [5]).

We get that $\mathsf{promise\text{-}BPP} \subseteq \mathsf{io\text{-}NSUBEXP}_c$, since infinitely often we get a truth table T of superpolynomial circuit complexity. Using the "interval trick" (as in the proof of Theorem 5, implication (2)), we boost this inclusion to get $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$, which, by Theorem 6, implies $\mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c]$. ◀

References

1 S. Aaronson and D. van Melkebeek. On circuit lower bounds from derandomization. *Theory of Computing*, 7(1):177–184, 2011.

2 M. Ajtai and A. Wigderson. Deterministic simulation of probabilistic constant depth circuits. In *Proceedings of the Twenty-Sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 11–19, 1985.

3 B. Aydinlioglu and D. van Melkebeek. Nondeterministic circuit lower bounds from mildly derandomizing arthur-merlin games. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 269–279. IEEE, 2012.

4 L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

5 L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

6 L. Babai and S. Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

7 M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13:850–864, 1984.

8 R.A. DeMillo and R.J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7:193–195, 1978.

9 L. Fortnow and R. Santhanam. Robust simulations and significant separations. In *Automata, Languages and Programming – 38th International Colloquium, ICALP, Proceedings, Part I*, pages 569–580, 2011.

10 J. Heintz and C.-P. Schnorr. Testing polynomials which are easy to compute. *L'Enseignement Mathématique*, 30:237–254, 1982.

11 R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

12 R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *STOC'97*, pages 220–229, 1997.

13 M.J. Jansen and R. Santhanam. Stronger lower bounds and randomness-hardness tradeoffs using associated algebraic complexity classes. In Christoph Dürr and Thomas Wilke, editors, *STACS*, volume 14 of *LIPIcs*, pages 519–530. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.

14 V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1–2):1–46, 2004.

15 E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. JAI Press, Greenwich, CT, 1989.

16 R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982.

17 R.M. Karp and R.J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(3-4):191–209, 1982.

18 J. Kinne, D. van Melkebeek, and R. Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Computational Complexity*, 21(1):3–61, 2012.

19 N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

20 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

21 J.T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27(4):701–717, 1980.

22 R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the Association for Computing Machinery*, 52(2):172–216, 2005.

23 A. Shamir. IP=PSPACE. *Journal of the Association for Computing Machinery*, 39(4):869–877, 1992.

24 M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

25 L. Trevisan and S.P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

26 C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

27 R. Williams. Nonuniform acc circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

28 A.C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

29 R.E. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of an International Symposium on Symbolic and Algebraic Manipulation (EUROSAM'79)*, LNCS, pages 216–226, 1979.