Pseudorandomness for Approximate Counting and Sampling

Ronen Shaltiel*
Department of Computer Science
University of Haifa
Mount Carlel, Haifa 31905, Israel.
ronen@haifa.ac.il

Christopher Umans†
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125.
umans@cs.caltech.edu.

October 19, 2006

Abstract

We study computational procedures that use both randomness and nondeterminism. Examples are Arthur-Merlin games and approximate counting and sampling of NP-witnesses. The goal of this paper is to derandomize such procedures under the weakest possible assumptions.

Our main technical contribution allows one to "boost" a given hardness assumption. One special case is a proof that

$$\mathrm{EXP} \not\subseteq \mathrm{NP}/\mathrm{poly} \Rightarrow \mathrm{EXP} \not\subseteq \mathrm{P}_{||}^{\mathrm{NP}}/\mathrm{poly}.$$

In words, if there is a problem in EXP that cannot be computed by poly-size nondeterministic circuits then there is one which cannot be computed by poly-size circuits that make non-adaptive NP oracle queries. This in particular shows that the various assumptions used over the last few years by several authors to derandomize Arthur-Merlin games (i.e., show AM $=$ NP) are in fact all *equivalent*. In addition to simplifying the framework of AM derandomization, we show that this "unified assumption" suffices to derandomize several other probabilistic procedures.

For these results we define two new primitives that we regard as the natural pseudorandom objects associated with *approximate counting* and *sampling* of NP-witnesses. We use the "boosting" theorem and hashing techniques to construct these primitives using an assumption that is no stronger than that used to derandomize AM. As a consequence, under this assumption, there are *deterministic* polynomial time algorithms that use *non-adaptive* NP-queries and perform the following tasks:

- approximate counting of NP-witnesses: given a Boolean circuit A, output r such that

$$(1-\epsilon)|A^{-1}(1)| \leq r \leq |A^{-1}(1)|.$$

- pseudorandom sampling of NP-witnesses: given a Boolean circuit A, produce a polynomial-size sample space that is computationally indistinguishable from the uniform distribution over $A^{-1}(1)$.

We also present applications. For example, we observe that Cai's proof that $\mathrm{S}_2^{\mathrm{P}} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ and the learning algorithm of Bshouty et al. can be seen as reductions to sampling that are not probabilistic. As a consequence they can be derandomized under the assumption stated above, which is weaker than the assumption that was previously known to suffice.

*Some of this work was done while at the Weizmann Institute and supported by the Koshland Scholarship. This research was also supported by BSF grant 2004329.

†This research was supported by NSF grant CCF-0346991, BSF grant 2004329, and an Alfred P. Sloan Research Fellowship.

1 Introduction

One of the major areas in complexity is the study of the power of randomness in various computational settings. In certain contexts randomness affords additional power. But for broad classes of problems it has been demonstrated over the last decade that randomness can be simulated deterministically, under widely accepted complexity assumptions.

The central object used in these derandomization results is a *pseudorandom generator* (PRG), which is an efficient deterministic procedure that generates a *discrepancy set* – a set of strings with the property that no test (from a pre-specified class of tests) can distinguish a random string in the discrepancy set from a uniformly random string. We say that a PRG *fools* this class of tests. A probabilistic procedure is derandomized by replacing its random bits with strings from the discrepancy set; the procedure cannot behave noticeably differently than it would with truly random bits, as then it would constitute a distinguishing test. As a consequence derandomizing stronger probabilistic algorithms typically requires pseudorandom generators that produce discrepancy sets for stronger classes of tests.

An efficient pseudorandom generator for some class of tests immediately implies an efficiently computable function which is hard for these tests. More specifically, an efficient pseudorandom generator that fools small circuits implies the existence of a language in a uniform complexity class (e.g., E $=$ DTIME$(2^{O(n)})$) that lies outside a non-uniform complexity class (e.g. P$/$poly). Thus constructing such pseudorandom generators amounts to proving circuit lower bounds for explicit functions, a task that is currently beyond our reach. Consequently, this line of research focuses on constructing pseudorandom generators under a *hardness assumption*[1]. In this context the goal is to achieve derandomization results under the weakest possible hardness assumptions.

One of the main efforts in derandomization over the last decade has focused on the class BPP which can be derandomized given access to pseudorandom generators that fool small circuits. Here the appropriate hardness assumption is that there exists a language in E that requires exponential size circuits (i.e., the language cannot be computed by size $2^{\epsilon n}$ circuits, for some $\epsilon > 0$).[2] A sequence of results [NW94, BFNW93, Imp95, IW97] showed that under this hardness assumption BPP $=$ P. A further sequence of papers achieved a *quantitatively optimal* hardness vs. randomness tradeoff [ISW99, ISW00, SU01, Uma03].

An analogous line of work [AK01, KvM02, MV99, SU01] derandomized Arthur-Merlin games [Bab85, GMR89]. (Recall that the class AM contains important and natural problems like graph non-isomorphism that are not known to be in NP). These works achieved AM $=$ NP under progressively *qualitatively* weaker hardness assumptions. The first results required average-case hardness for circuits that make non-adaptive queries to an NP oracle, while the latest results require only hardness for SV-nondeterministic circuits.[3] In this paper we show that the various different assumptions used to derandomize AM are in fact equivalent.

A prior line of research [Sto83, JVV86, BGP00] addresses procedures which approximately count and sample NP-witnesses. More precisely, given a Boolean circuit A the first task is to approximately count the number of accepting inputs of A, and the second is to sample a random accepting input. Note that both problems are NP-hard and thus any such procedure must use nondeterminism unless NP=P. The current

[1]This "hardness vs. randomness paradigm" was initiated by [BM84, Yao82]. It should be noted that the notion of pseudorandom generators in these papers is different than the one we use here. In particular, in this paper we follow a paradigm initiated by [NW94] which allows pseudorandom generators which given a size bound s, run in time polynomial in s and output a discrepancy set for size s circuits. The reader is referred to [Gol98] for a survey on pseudorandomness and its applications and to [Kab02] for a recent survey which focuses on derandomization.

[2]One of the confusing aspects of all the results in this area is that the assumptions involve "exponential time" classes. In actual applications these assumptions are "scaled down" to say that there exists a function on $O(\log n)$ bits which is computable in polynomial time and cannot be computed by size n^c circuits (for some constant c).

[3]SV-nondeterministic circuits are the nonuniform analog of the class NP $\cap$ coNP (see definition 2.2).

known procedures for these tasks also use randomization: they are probabilistic algorithms that use an NP-oracle. In this paper we show how to derandomize these procedures and show that under a hardness assumption that is no stronger than that used to derandomize AM, both of these tasks can be performed by polynomial time *deterministic* algorithms that make *non-adaptive* NP-queries.

In order to achieve these results we make a technical contribution and a conceptual contribution. Our main technical result is a "downward collapse theorem" that implies (as a special case):[4]

$$\mathrm{E} \subseteq \mathrm{P}_{||}^{\mathrm{NP}}/\mathrm{poly} \Rightarrow \mathrm{E} \subseteq \mathrm{NP}/\mathrm{poly}.$$

This downward collapse shows that all of the various flavors of nondeterministic hardness assumptions considered in the literature are equivalent. This unifies and simplifies a number of past results. This result is also helpful when derandomizing other probabilistic procedures that involve randomness and nondeterminism. It allows us to start from a weak hardness assumption, boost it to a stronger hardness assumption, and then use pseudorandom generators for stronger classes of tests, namely circuits which make non-adaptive NP-queries.

Our conceptual contribution lies in defining what we regard as the natural "derandomization objects" associated with approximate counting and sampling. These are *relative-error approximators* (for approximate counting) and *conditional discrepancy sets* (for sampling). The first is a strengthening of additive-error approximators (which derandomize BPP), and the second is a generalization of discrepancy sets (which "sample" from the uniform distribution). We show how to obtain relative-error approximators and conditional discrepancy sets in polynomial time with non-adaptive NP oracle access, under a hardness assumption no stronger than that used for derandomizing AM. Note that this suggests that the "true complexity" of these problems is $\mathrm{P}_{||}^{\mathrm{NP}}$. Loosely speaking, our technique uses the strong pseudorandom generators obtained by boosting the initial hardness assumption to derandomize the probabilistic procedures for approximate counting and sampling. Some additional work is needed to obtain procedures that make *nonadaptive* queries to an NP-oracle.

We also give several applications of relative error approximators and conditional discrepancy sets. We obtain the following collapses under a hardness assumption no stronger than that used for derandomizing AM: $\mathrm{S}_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}}$ and $\mathrm{BPP}_{path} = \mathrm{P}_{||}^{\mathrm{NP}}$. The first collapse comes from viewing Cai's result [Cai01] (that places $\mathrm{S}_2^{\mathrm{P}}$ in $\mathrm{ZPP}^{\mathrm{NP}}$) as a reduction of $\mathrm{S}_2^{\mathrm{P}}$ to sampling that uses an NP oracle but is *not probabilistic*. This allows a derandomization via conditional discrepancy sets. Similarly, we view a fundamental result by Bshouty et al. [BCG+96] (concerning the learning of circuits using equivalence queries) as a reduction to sampling, and derandomize it in the same way.

Outline

In Section 2 we present definitions of the various types of nondeterministic circuits and hardness assumptions. In Section 3 we describe our main results and relation to prior work. In Section 4 we present corollaries and applications of our main results. In Section 5 we describe the major ideas and techniques used in the proofs; Sections 6 and 7 contain the full proofs. Finally in Section 8 we conclude with some open problems.

[4] The notation $A_{||}^{B}$ says that A uses *non-adaptive* queries to oracle B.

2 Nondeterministic circuits, hardness, and PRGs

We assume that the reader is familiar with (deterministic) Boolean circuits. We use the convention that the size of a circuit is the total number of wires and gates. Nondeterministic circuits come in several flavors, which we define below. We remark that a main contribution of this paper lies in showing that the several hardness assumptions defined below are all equivalent – unfortunately, in order to show that, we need to be able to discuss all of the various assumptions below.

Definition 2.1 (nondeterministic and co-nondeterministic circuits). *A nondeterministic (resp. co-nondeterministic) circuit is a Boolean circuit C which receives two inputs: x of length n and a second input y. The* function computed by C*, denoted $f_C : \{0,1\}^n \to \{0,1\}$ is defined by $f_C(x) = 1$ iff $\exists y\ C(x,y) = 1$ (resp. $\forall y\ C(x,y) = 0$).*

The uniform analogue of poly-size nondeterministic circuit is the class NP. The uniform analogue of poly-size co-nondeterministic circuits is coNP. Poly-size single-valued nondeterministic circuits have NP $\cap$ coNP as their uniform analogue.

Definition 2.2 (single-valued nondeterministic circuits). *A single-valued nondeterministic (or SV-nondeterministic) circuit is a Boolean circuit C which receives two inputs: x of length n and a second input y, and has two output gates:* **value** *and* **flag**. *Circuit C computes the function $f : \{0,1\}^n \to \{0,1\}$ if the following hold:*

- *for every x, y, if $C(x,y)$ has 1 at its* **flag** *gate then $C(x,y)$ has $f(x)$ at its* **value** *gate, and*
- *for every x, there exists some y for which $C(x,y)$ has 1 at its* **flag** *gate.*

Note that a circuit C may meet the syntactic demands of this definition, and yet not compute any function (if the two listed *semantic* requirements for "computing a function" are not met). When we refer to a SV-nondeterministic circuit, we always mean a circuit C that in fact computes a function according to this definition, and we refer to that unique function as the *function computed by C*. We also remark that a function has a size $O(s)$ SV-nondeterministic circuit if and only if it has both a size $O(s)$ nondeterministic circuits and a size $O(s)$ co-nondeterministic circuit.

Definition 2.3 (adaptive and non-adaptive SAT-oracle circuits). *A SAT-oracle circuit is a Boolean circuit C that is also permitted to use SAT-oracle gates. A SAT-oracle gate is a gate with many inputs and a single output that is 1 iff the input is in SAT.*

A nonadaptive SAT-oracle circuit is a pair of Boolean circuits C_{pre} and C_{post}. On input x, C_{pre} outputs a number of queries $q_1, q_2, \ldots, q_m$. Circuit C_{post} receives x together with m bits $a_1, a_2, \ldots, a_m$, where $a_i = 1$ iff q_i is in SAT, and outputs a single answer bit.

We could also have defined nonadaptive SAT-oracle circuits to be SAT-oracle circuits in which no path from the output gate to an input gate encounters more than one SAT-oracle gate; the above definition makes explicit the pre- and post- processing phase. For nonadaptive SAT-oracle circuits so defined, their size is the sum of the sizes of C_{pre} and C_{post}.

We will frequently speak of a language L that is "hard for" a class of circuits. Of course this hardness can be quantified by the size of the circuit. For clarity, we have chosen only to present the "high-end" results that follow when this hardness is exponential, even though more general results are true using our methods. Consequently, we only need the following definitions:

Definition 2.4 (worst-case hardness for exponential-size circuits). *A language L is worst-case hard for exponential-size (deterministic, nondeterministic, co-nondeterministic, SV-nondeterministic, adaptive or nonadaptive SAT-oracle -) circuits if there exists a constant $\epsilon > 0$ such that for every sufficiently large n, every circuit of the prescribed type and size at most $2^{\epsilon n}$, fails to compute L restricted to inputs of length n.*

Definition 2.5 (average-case hardness for exponential-size circuits). *A language L is α-hard for exponential-size (deterministic, nondeterministic, co-nondeterministic, SV-nondeterministic, adaptive or nonadaptive SAT-oracle -) circuits if there exists a constant $\epsilon > 0$ such that for every sufficiently large n, every circuit of the prescribed type and size at most $2^{\epsilon n}$, succeeds to compute L restricted to inputs of length n on at most $\alpha \cdot 2^n$ such inputs.*

Note that the definition of $(1 - 2^{-n})$-hard coincides with the definition of worst-case hard.

Definition 2.6 (worst-case and average-case hardness of complexity classes). *A complexity class $\mathcal{C}$ is worst-case hard (resp. α-hard) for exponential-size circuits of a given type if there exists a language $L \in \mathcal{C}$ that is worst-case hard (resp. α-hard) for exponential-size circuits of that type.*

We also sometimes say "$\mathcal{C}$ *requires* exponential-size circuits" of a given type to mean $\mathcal{C}$ is worst-case hard for exponential-size circuits of that type.

2.1 Discrepancy sets and pseudorandom generators

In this paper we define pseudorandom generators in terms of discrepancy sets.

Definition 2.7 (discrepancy set). *Let $\mathcal{D}$ be a subset of all functions from $\{0,1\}^n$ to $\{0,1\}$. A multiset $T \subseteq \{0,1\}^n$ is an (n, ϵ)-*discrepancy set *for $\mathcal{D}$ if for every $D \in \mathcal{D}$,*

$$\left| \Pr_{x \in \{0,1\}^n}[D(x) = 1] - \Pr_{t \in T}[D(t) = 1] \right| \leq \epsilon.$$

Commonly $\mathcal{D}$ is the set of functions with size s deterministic circuits; in this case we use the shorthand (n, s, ϵ)-*discrepancy set* (as in subsection 3.2.2). A pseudorandom generator is a function whose output is a discrepancy set[5].

Definition 2.8 (pseudorandom generator). *Let $\mathcal{C}$ be a complexity class. A pseudorandom generator (PRG) for $\mathcal{C}$ is a procedure which on input 1^n outputs a $(n, 1/n)$-discrepancy set for the set $\mathcal{D}$ of all characteristic functions of languages in $\mathcal{C}$ restricted to length n.*

In this paper $\mathcal{C}$ will typically be the class of those languages with nondeterministic circuits of a given type, and whose size is a fixed polynomial.

[5] A more standard formulation is that a pseudorandom generator "stretches" a short seed into a long pseudorandom string, with the property that the set of all pseudorandom strings is a discrepancy set. Our definition asks the pseudorandom generator to output all pseudorandom strings at once. This difference is immaterial in this paper as we will be concentrating on discrepancy sets with polynomial size, and thus the entire set can be output in polynomial time if each individual string can be generated in polynomial time.

3 Main results

Several of our results apply to any complexity class for which one can compute the low-degree extension within that class. To make these results easier to state we introduce the following definition:

Definition 3.1. *We say that a complexity class $\mathcal{C}$* allows low-degree extension *if $E^{\mathcal{C}^{\leq O(n)}} \subseteq \mathcal{C}$, where the notation $\mathcal{C}^{\leq O(n)}$ means that the E oracle machine makes only linear-length queries.*

Examples of complexity classes $\mathcal{C}$ that support low-degree extension are: E, NE $\cap$ coNE, E^{NP}, $\mathrm{E}_{||}^{\mathrm{NP}}$.

3.1 Unifying hardness assumptions

Several authors [AK01, KvM02] have observed that the PRG constructions intended to derandomize BPP can be adapted to construct discrepancy sets that fool efficient *non-deterministic* tests under stronger hardness assumptions. Just as PRGs that fool efficient deterministic tests imply BPP $=$ P, PRGs that fool efficient non-deterministic tests imply AM $=$ NP.

Several hardness assumptions sufficient to achieve AM $=$ NP have been considered in the literature. All of these hardness assumptions (and the others we will consider in this paper) have the following form: there exists a language L in some "high" uniform class (examples are E, NE $\cap$ coNE, $\mathrm{E}_{||}^{\mathrm{NP}}$ and E^{NP}) that requires exponential size circuits from some non-uniform circuit class[6]. Three non-uniform circuit classes have been discussed in the literature in relation to AM. These are

- SV-nondeterministic circuits, used by Milersen and Vinodchandran [MV99] and later Shaltiel and Umans [SU01],
- non-deterministic (and co-nondeterministic) circuits, used by Arvind and Kobler [AK01], and
- Nonadaptive SAT-oracle circuits, used by Klivans and van Melkebeek [KvM02][7],

listed in order from weaker to stronger. Perhaps the best way to understand these circuit classes is to think of them as nonuniform analogs of NP $\cap$ coNP, NP (and coNP), and $\mathrm{P}_{||}^{\mathrm{NP}}$, respectively. Figure 1 summarizes the various hardness assumptions and pseudorandom objects implying AM $=$ NP and known relationships between them.

Notice that with the exception of the "AM $=$ NP" box, prior to this work there were two strongly connected components, consisting of the top row and the bottom two rows. In this paper we show that *all of the hardness assumptions considered in the literature are equivalent*. In addition to clarifying the situation, this result somewhat simplifies the task of building a PRG sufficient to derandomize AM. One can replace previous constructions [MV99, SU01] that are specialized for derandomizing AM under an SV-nondeterministic hardness assumption by *any* relativizing construction of ordinary pseudorandom generators (designed to derandomize BPP).

[6] We stress that it is the choice of the nonuniform circuit class that typically plays an important role in the argument. Loosely speaking, this choice determines the class of tests to be fooled by the generator. The choice of the uniform class determines the efficiency of the generator. For example, choosing this class to be E gives a generator which runs in P, whereas $NE \cap coNE$ (or E^{NP}) give a generator which runs in NP $\cap$ coNP (or P^{NP}). We encourage the reader to ignore the precise choice of the uniform class at a first reading and focus on the choice of the nonuniform class.

[7] Actually, the paper in question refers to SAT oracle circuits, but their argument works just as well for nonadaptive SAT-oracle circuits, giving a stronger result.

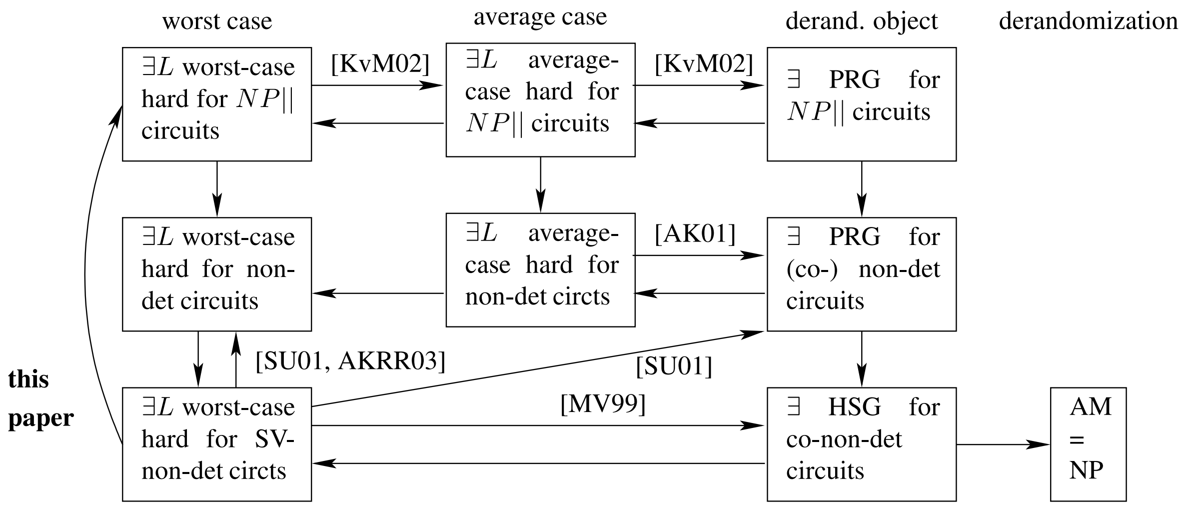

Figure 1: Assumptions implying AM $=$ NP. In all cases L is a language in NE $\cap$ coNE. The phrase "L worst-case (resp., average-case) hard for" means "L cannot be computed exactly by (resp., approximated by) size $2^{\epsilon n}$ for some $\epsilon > 0$." Arrows indicate implications; unlabelled arrows correspond to implications that follow from standard arguments.

3.1.1 A downward collapse theorem

The equivalence of the various hardness assumptions is implied by the following downward collapse theorem, which may be of independent interest:

Theorem 3.2 (downward collapse). *Let $\mathcal{C}$ be any complexity class that allows low-degree extension. If every language in $\mathcal{C}$ has nonadaptive SAT-oracle circuits of size $s(n)$ then every language in $\mathcal{C}$ has SV-nondeterministic circuits of size $s(n)^{O(1)}$.*

A special case of Theorem 3.2 is:

$$\mathrm{E} \subseteq \mathrm{P}_{||}^{\mathrm{NP}}/\mathrm{poly} \Rightarrow \mathrm{E} \subseteq \mathrm{NP}/\mathrm{poly} \cap \mathrm{coNP}/\mathrm{poly}.$$

We remark that it is widely believed that $\mathrm{P}_{||}^{\mathrm{NP}}$ is stronger than NP $\cap$ coNP and that nonadaptive SAT-oracle circuits are stronger than SV-nondeterministic circuits. Nevertheless, a collapse of E to the stronger class implies a further collapse to the weaker class.

The following Corollary is the contrapositive version of Theorem 3.2 which states that a "weak" hardness assumption implies a stronger one:

Corollary 3.3. *For every class $\mathcal{C}$ that allows low-degree extension, if $\mathcal{C}$ is worst-case hard for exponential-size SV-nondeterministic circuits then $\mathcal{C}$ is worst-case hard for exponential-size nonadaptive SAT-oracle circuits.*

Corollary 3.3 will allow us to derandomize many probabilistic algorithms and classes using hardness for SV-nondeterministic circuits by first "boosting" this assumption to hardness for nonadaptive SAT-oracle circuits, and then working with the pseudorandom generators obtained from the latter assumption.

3.2 Derandomization objects for approximate counting and sampling

In this section we define two generic computational objects – relative-error approximators, and conditional discrepancy sets. These objects are natural and make no reference to nondeterminism. They are intended to capture approximate counting and sampling, and they generalize and strengthen two existing and widely used objects: additive-error approximators and (ordinary) discrepancy sets.

3.2.1 Relative-error approximators

Ordinary pseudo-random generators allow one to obtain an *additive* approximation of the acceptance probability of circuits:

Definition 3.4. *An (additive-error)* approximator *is a procedure that takes as input a Boolean circuit A, and $\epsilon > 0$, and outputs a real number ρ for which*

$$\left|\Pr_x[A(x) = 1] - \rho\right| \leq \epsilon.$$

Indeed additive approximation is in some sense the *raison d'etre* of ordinary PRGs, because additive approximation of the acceptance probability of circuits allows one to derandomize BPP. Relative error approximation allows *approximate counting*, and is much more difficult (it is NP-hard). We will be concerned with *relative-error* approximations of the acceptance probability of circuits:

Definition 3.5. *A* relative-error approximator *is a procedure that takes as input a Boolean circuit A, and $\epsilon > 0$, and outputs a real number ρ for which*

$$(1 - \epsilon) \Pr_x[A(x) = 1] \leq \rho \leq \Pr_x[A(x) = 1].$$

We give a construction of deterministic relative-error approximators under a hardness assumption for SV-nondeterministic circuits.

Theorem 3.6 (construction of relative-error approximators). *If $E_{||}^{NP}$ requires exponential size SV-nondeterministic circuits, then there is a deterministic* relative-error approximator *that runs in time polynomial in the length of its input and $1/\epsilon$, with* non-adaptive *access to an NP oracle.*

As an immediate corollary, we obtain

Corollary 3.7. *If $E_{||}^{NP}$ requires exponential size SV-nondeterministic circuits, then for every $\#P$ function $f : \{0,1\}^n \to \mathbb{N}$, and every $\epsilon > 0$, there is a deterministic procedure P running in poly(n, ϵ^{-1}) time with non-adaptive access to an NP-oracle for which (for all x):*

$$(1 - \epsilon)f(x) \leq P(x) \leq f(x);$$

in other words, every problem in $\#P$ can be approximated in $P_{||}^{NP}$.

Note that it was shown in [Sto83, JVV86, BGP00] that using randomness and an NP-oracle, it is possible to uniformly sample NP-witnesses. This implies that every problem in $\#P$ has a *fully polynomial-time randomized approximation scheme* (FPRAS) with access to an NP-oracle. However, no deterministic *fully polynomial-time approximation schemes* (FPAS's) with access to an NP-oracle are known for any $\#P$-complete problem; the above corollary gives FPAS's that make non-adaptive NP oracle queries for all problems in $\#P$, albeit under a complexity assumption.

3.2.2 Conditional discrepancy sets

Ordinary pseudo-random generators are sometimes called "discrepancy set generators," since they produce the following object:

Definition 3.8. *An* (n, s, ϵ)-discrepancy set *is a subset* $T \subseteq \{0,1\}^n$ *with the property that for all Boolean circuits* C *of size at most* s*:*

$$\left| \Pr_x[C(x) = 1] - \Pr_{t \in T}[C(t) = 1] \right| \leq \epsilon.$$

A discrepancy set is a "good sample" of *strings* $x \in \{0,1\}^n$, with respect to any property $\mathcal{P}$ that is decidable by small Boolean circuits. Of course one particularly useful such property is the property that a BPP machine with a fixed input accepts when given string x as its random coins.

Frequently one wishes to obtain a "good sample" of *strings* $x \in S$ for some subset $S \subseteq \{0,1\}^n$. Again, the sample should be good with respect to any property $\mathcal{P}$ that is recognizable by small Boolean circuits. For example S may be the set of proper 3-colorings of a given graph; a property of interest might be the property of having two specified nodes colored with the same color. A large body of literature is devoted to sampling various structures (e.g., colorings, matchings, contingency tables, etc...), often employing Markov Chain Monte Carlo methods.

We define *conditional discrepancy sets* as the derandomization object associated with such sampling in its full generality. We will allow the set S to be any set recognizable by a small Boolean circuit; that is, $S = A^{-1}(1)$ for some small circuit A. Conditional discrepancy sets capture "pseudorandomly sampling an accepting input of A" and can be seen to be a natural generalization of ordinary discrepancy sets.

Definition 3.9. *Let* $S \subseteq \{0,1\}^n$ *be some subset. An* S-conditional (n, s, ϵ)-discrepancy set *is a subset* $T \subseteq S$ *with the property that for all Boolean circuits* C *of size at most* s*:*

$$\left| \Pr_x[C(x) = 1 | x \in S] - \Pr_{t \in T}[C(t) = 1 | t \in S] \right| \leq \epsilon.$$

Our main result here is a procedure to efficiently generate conditional discrepancy sets under a hardness assumption (which is no stronger than the hardness assumption used to derandomize AM):

Theorem 3.10 (construction of conditional discrepancy sets). *If* $E_{||}^{NP}$ *(resp.* E^{NP}*) requires exponential size SV-nondeterministic circuits, then there is a deterministic procedure that takes as input a Boolean circuit* A *that accepts a subset* $S \subseteq \{0,1\}^n$*, an integer* s*, and* $\epsilon > 0$*, and outputs an* S-conditional (n, s, ϵ)-discrepancy set $T \subseteq S$*. The procedure runs in poly*$(|A|, n, s, 1/\epsilon)$ *time with* non-adaptive *(resp. adaptive) access to an NP oracle.*

4 Corollaries and Applications

In this section we show several applications of our main results (Theorems 3.2, 3.6, and 3.10). In most of them we are able to achieve certain "derandomization tasks" under assumptions which are seemingly weaker than previously known.

4.1 Derandomizing $\mathrm{BPP}_{||}^{\mathrm{NP}}$

Using the downward collapse theorem (Theorem 3.2), we get that $\mathrm{BPP}_{||}^{\mathrm{NP}} = \mathrm{P}_{||}^{\mathrm{NP}}$ under a hardness assumption. Previously, this conclusion required hardness for non-adaptive SAT-oracle circuits [KvM02], while here we use only hardness for SV-nondeterministic circuits:

Theorem 4.1. *If $E_{||}^{NP}$ requires exponential size SV-nondeterministic circuits, then $BPP_{||}^{NP} = P_{||}^{NP}$.*

The proof follows after a brief discussion of relativizing PRGs. Klivans and van Melkebeek [KvM02] formalized the notion of a *relativizing* PRG construction, and observed that such constructions can be used to fool circuit classes that are stronger than deterministic circuits, if one is willing to make a similarly stronger hardness assumption. One example is that assuming there exist languages that are hard for nonadaptive SAT-oracle circuits, one can construct PRGs that fool nonadaptive SAT-oracle circuits. Our Corollary 3.3 states that hardness for SV-nondeterministic circuits implies hardness for nonadaptive SAT-oracle circuits. As a consequence, existing relativizing PRG constructions (e.g. [IW97, STV01]) may be used directly to fool nonadaptive SAT-oracle circuits, assuming only hardness for SV-nondeterministic circuits. As stated in Theorem 4.1, this in turn derandomizes the class $\mathrm{BPP}_{||}^{\mathrm{NP}}$ using a weaker assumption than previously known. The exact details follow.

The following is a slight refinement of a theorem in [KvM02] (we use the additional fact that the NP oracle access in their argument is always non-adaptive):

Theorem 4.2 ([KvM02]). *If $E_{||}^{NP}$ (resp. E) requires exponential size non-adaptive SAT-oracle circuits then there is a PRG for linear-size non-adaptive SAT-oracle circuits that runs in polynomial time with non-adaptive access to an NP oracle (resp. polynomial time).*

We obtain the following improvement:

Theorem 4.3. *If $E_{||}^{NP}$ (resp. E) requires exponential size SV-nondeterministic circuits then there is a PRG for linear-size non-adaptive SAT-oracle circuits that runs in polynomial time with non-adaptive access to an NP oracle (resp. polynomial time).*

Proof. Combine Theorem 4.2 with Corollary 3.3. □

Theorem 4.1 now follows in a completely standard way, which we recount here for completeness:

Proof of Theorem 4.1. Given a $\mathrm{BPP}_{||}^{\mathrm{NP}}$ algorithm $A(x, y)$ for language L and an input x, define $C_x(y) = A(x, y)$. After padding with dummy inputs, C_x can be computed by a linear-size non-adaptive SAT-oracle circuit. We run the PRG of Theorem 4.3 on input $1^{|x|}$ to produce a discrepancy set T that fools circuit C_x. We compute $A(x, t)$ for each $t \in T$ in parallel, and output the majority. This constitutes a deterministic algorithm that decides language L in polynomial time with non-adaptive access to an NP oracle. Formally, one must appeal to Lemma 7.2 (appearing in a later section) to ensure that the overall procedure can all be done with non-adaptive queries. □

Also, as explained in the introduction, Theorem 4.1 gives an alternative way of constructing PRGs for nondeterministic circuits from an SV-nondeterministic hardness assumption. This permits the use of "standard constructions" in this setting, whereas previous constructions [MV99, SU01] were specialized to the nondeterministic case.

4.2 Finding NP witnesses in $\mathrm{P}_{||}^{\mathrm{NP}}$

We now present an important additional application of the downward collapse Theorem (Theorem 3.2):

Theorem 4.4. *If $E_{||}^{NP}$ requires exponential size SV-nondeterministic circuits, then there is a procedure that, given a circuit C, outputs a satisfying assignment for C if one exists, and runs in polynomial time with* non-adaptive *NP-oracle access.*

Again, this is a conclusion that was known to hold under a hardness assumption for non-adaptive SAT-oracle circuits [KvM02]. Applying Corollary 3.3 immediately gives us Theorem 4.4, which reaches the same consequence from a weaker assumption. We highlight this particular application because we will make use of it later, in the proof of Theorem 3.10.

For completeness, we describe the proof idea from [KvM02] (which builds on earlier work by [BDCGL90]). They prove that if $\mathrm{E}_{||}^{\mathrm{NP}}$ requires exponential size nonadaptive SAT-oracle circuits, then there is a polynomial-time procedure to *produce* a satisfying assignment of a given circuit C that uses *non-adaptive* access to an NP-oracle. Note that the standard method uses adaptive access. The non-adaptive procedure comes from noting that there is a polynomial time algorithm that makes non-adaptive NP queries to test whether the outcome of applying the Valiant-Vazirani reduction to a satisfiable circuit C (for a specific choice of random bits) succeeds in producing a circuit that has a unique satisfying assignment. Using a PRG for nonadaptive SAT-oracle circuits, it is then possible to deterministically produce a list of candidate circuits from C, one of which is guaranteed to have a unique satisfying assignment. For this circuit C', one can find the satisfying assignment by making the following queries in parallel: "Does C' have a satisfying assignment that assigns x_i true?" and "Does C' have a satisfying assignment that assigns x_i false?" for all i. The overall procedure requires only non-adaptive NP-oracle access, as promised.

4.3 Hardness amplification for nondeterministic circuits

Hardness amplification results transform functions which are hard on the worst case into functions which are hard on the average. In a sequence of works [BFNW93, Imp95, IW97, STV01] it was shown that for every class which allows low degree extension if the class is hard on the worst case for small *deterministic* circuits then the class is hard on average for small *deterministic* circuits. The first hardness amplification result for nondeterministic circuits was given in [SU01]:

Theorem 4.5 ([SU01]). *Let $\mathcal{C}$ be a complexity class that allows low-degree extension. For every $\epsilon > 0$, if $\mathcal{C}$ is hard for size s nondeterministic circuits then $\mathcal{C}$ is $(1/2+\epsilon)$-hard for size $s' = (s\epsilon/n)^{\Omega(1)}$ nondeterministic circuits.*

Using Theorem 3.2 together with the "hardness amplification" results of [STV01] (for deterministic circuits) gives a hardness amplification result for *nondeterministic* circuits. Altogether this is a simpler and more modular proof of Theorem 4.5.

We now present the new proof. We first restate the results of [STV01] in the following way:

Theorem 4.6 ([STV01]). *Let $\mathcal{C}$ be a class which allows low degree extension. There exists a constant c such that for every function $f : \{0,1\}^n \to \{0,1\}$ such that $f \in \mathcal{C}$ and $\epsilon > 2^{-n}$ there is a function $\bar{f} : \{0,1\}^{n'=O(n)} \to \{0,1\}$ such that $\bar{f} \in \mathcal{C}$ and for every function $D : \{0,1\}^{n'} \to \{0,1\}$ such that*

$$\Pr_{x\in\{0,1\}^{n'}}[D(x) = \bar{f}(x)] \geq 1/2 + \epsilon$$

there is an oracle circuit C such that C^D computes f using only non-adaptive queries to D, and the size of C is $(n/\epsilon)^c$.

Indeed, this is a complicated way to say that if f is hard for size s deterministic circuits then $\bar{f}$ is hard on average for slightly smaller deterministic circuits. We chose to state Theorem 4.6 this way, because in this form it also gives a hardness amplification result for other classes of circuits. The following corollary is an example.

Corollary 4.7. *Let $\mathcal{C}$ be a class which allows low degree extension. If $\mathcal{C}$ is hard for size s non-adaptive SAT-oracle circuits then $\mathcal{C}$ is $1/2 + \epsilon$-hard for size $s' = (s\epsilon/n)^{\Omega(1)}$ non-adaptive SAT-oracle circuits.*

Proof. One only has to notice that if D is a size s' non-adaptive SAT-oracle circuit then C^D (from Theorem 4.6) is a size $s' \cdot \mathrm{poly}(n, 1/\epsilon)$ non-adaptive SAT-oracle circuit. □

It is important to note that this argument does not work directly for nondeterministic circuits. The reason is that it does not follow that if D is a nondeterministic circuit and C is a deterministic circuit then C^D is a nondeterministic circuit. (Consider for example the case where D computes SAT and C flips the result. The circuit C^D computes co-SAT which is not believed to be computable by a small nondeterministic circuit.) However, Theorem 3.2 allows us to convert hardness for nondeterministic circuits into hardness for non-adaptive SAT-oracle circuits which can be used in Corollary 4.7. The exact details follow:

Proof of Theorem 4.5. We are assuming that $\mathcal{C}$ is hard for size s non-deterministic circuits (and hence also for size s SV-nondeterministic circuits). By Theorem 3.2 we have that there exists a function $\bar{f} : \{0,1\}^{O(n)} \to \{0,1\}$ in $\mathcal{C}$ that is hard for size $s' = s^{\Omega(1)}$ non-adaptive SAT-oracle circuits, and the theorem then follows from Corollary 4.7, which gives that $\mathcal{C}$ is hard on average even for non-adaptive SAT-oracle circuits of size $(s\epsilon/n)^{\Omega(1)}$. □

4.4 Derandomizing BPP$_{path}$

The class BPP$_{path}$ was defined by Han, Hemaspaandra and Theirauf [HHT97]. It is the class of languages L for which there exists a non-deterministic polynomial-time Turing Machine M for which

$$\begin{aligned} x \in L &\Rightarrow \geq 2/3 \text{ of } M\text{'s computation paths accept} \\ x \notin L &\Rightarrow \geq 2/3 \text{ of } M\text{'s computation paths reject.} \end{aligned}$$

Notice that the computation paths need not all make the same number of non-deterministic choices; if they are required to, we just get BPP. In contrast to BPP, BPP$_{path}$ is quite powerful: it is known to contain $\mathrm{P}_{||}^{\mathrm{NP}}$ [HHT97]. The next theorem suggests it is probably *equal* to $\mathrm{P}_{||}^{\mathrm{NP}}$.

Theorem 4.8. *If $E_{||}^{NP}$ requires exponential size SV-nondeterministic circuits, then $BPP_{path} = P_{||}^{NP}$.*

Proof of Theorem 4.8. Let L be a language in BPP$_{path}$ with associated non-deterministic Turing Machine M. Let $p(n)$ be an upper bound on the running time of M on an input of length n.

Fix an input x. Let D_x be a circuit outputting 1 iff the following procedure accepts: given $y \in \{0,1\}^{p(|x|)}$, simulate M using successive bits of y as M's non-deterministic choices. When M halts, if the remainder of y is all-zeros, then accept, otherwise reject.

Let C_x be a circuit outputting 1 iff the following procedure accepts: given $y \in \{0,1\}^{p(|x|)}$, simulate M using successive bits of y as M's non-deterministic choices and accept if and only if M accepts.

Observe that the probability over computation paths of M that M accepts input x is exactly:

$$\alpha = \Pr_y[C_x(y) = 1 | D_x(y) = 1],$$

since each 1 of D_x corresponds to a unique computation path.

We use the relative-error approximator of Theorem 3.6 twice (in parallel), once with input C_x, and once with input $D_x \wedge C_x$, and $\epsilon = 1/10$. Let ρ_1 and ρ_2 be the two approximations. Notice that

$$(1-\epsilon)\alpha \leq (\rho_2/\rho_1) \leq (1-\epsilon)^{-1}\alpha.$$

We accept iff $\rho_2/\rho_1 > 1/2$, which is guaranteed to happen iff $\Pr_y[C_x(y) = 1 | D_x(y) = 1] \geq 2/3$. The entire procedure runs in time poly$(|x|)$ with non-adaptive NP oracle access. □

4.5 Collapsing $\mathrm{S}_2^{\mathrm{P}}$ to P^{NP}

The class $\mathrm{S}_2^{\mathrm{P}}$ was defined by [Can96] and [RS98]. It is the class of languages L for which there is a polynomial-time predicate R for which:

$$x \in L \quad \Rightarrow \quad \exists y \, \forall z \; R(x,y,z) = 1 \tag{1}$$

$$x \notin L \quad \Rightarrow \quad \exists z \, \forall y \; R(x,y,z) = 0. \tag{2}$$

Cai [Cai01] recently showed that the class $\mathrm{S}_2^{\mathrm{P}}$ (which contains P^{NP} and MA) is contained in $\mathrm{ZPP}^{\mathrm{NP}}$. One consequence of this result is that under a hardness assumption sufficient to derandomize $\mathrm{ZPP}^{\mathrm{NP}}$, the class $\mathrm{S}_2^{\mathrm{P}}$ collapses to P^{NP}. This is remarkable because $\mathrm{S}_2^{\mathrm{P}}$ is defined by alternating quantifiers and has more of the flavor of the Polynomial-Time Hierarchy than any randomized complexity class; yet derandomization techniques yield a surprising collapse.

We view Cai's result as a reduction of $\mathrm{S}_2^{\mathrm{P}}$ to sampling, and thus obtain the following collapse as an application of Theorem 3.10. Note that this result does *not* follow directly from $\mathrm{S}_2^{\mathrm{P}} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ using straightforward derandomization techniques, as any derandomization via derandomizing $\mathrm{ZPP}^{\mathrm{NP}}$ requires a stronger hardness assumption (given current technology) to cope with adaptive NP-queries.

Theorem 4.9. *If E^{NP} requires exponential size SV-nondeterministic circuits, then $S_2^P = P^{NP}$.*

Proof. Let L be a language in $\mathrm{S}_2^{\mathrm{P}}$, and let R be the associated polynomial-time predicate for which Eqs. (1) and (2) hold. By padding if necessary we may assume that $|x| = |y| = |z| = n$. Let s be the running time of R.

The procedure to decide if $x \in L$ operates in rounds. Initially, we set $i = 0$, and $S_0 = \{0,1\}^n$, and observe that S_0 is clearly recognized by a trivial circuit C_0. We now begin round 0.

In round i we do the following:

1. In P^{NP}, generate the S_i-conditional $(n, s^3, 1/2)$-discrepancy set $T_i \subseteq S_i$ (using Theorem 3.10). The size of T_i and the time to generate it are both at most poly$(|C_i|, n, s) =$ poly(n, s, i).

2. If $\forall z \; \bigvee_{t \in T_i} R(x,t,z) = 1$ then accept.

3. Otherwise, find z_i for which $\bigvee_{t \in T_i} R(x,t,z_i) = 0$.

4. Define $S_{i+1} = \{y : y \in S_i \wedge R(x, y, z_i) = 1\}$, and observe that S_{i+1} is recognized by a circuit C_{i+1} of size $O(s^2 + |C_i|)$.

5. If $S_{i+1} = \emptyset$, then reject; otherwise, begin round $i + 1$.

Notice that step 2 requires a single NP-oracle query, as does step 5, and that step 3 involves finding an NP-witness in the usual way with multiple adaptive NP-oracle queries.

For correctness, observe that if we accept, we have found that the complement of Eq. (2) holds; if we reject, then $\forall y\ \exists z_i\ R(x, y, z_i) = 0$, and thus the complement of Eq. (1) holds.

The main claim is that the number of rounds before this procedure either accepts or rejects is at most $n + 1$. Notice that at step 3, we must have that

$$\Pr_y[R(x, y, z_i) = 1 | y \in S_i] \leq 1/2,$$

since $\Pr_{t \in T_i}[R(x, t, z_i) = 1 | t \in S_i] = 0$ and the circuit computing R with x and z_i hard-wired has size at most $O(s^2) < s^3$, and T_i is an S_i-conditional $(n, s^3, 1/2)$-discrepancy set. Thus $|S_{i+1}| \leq |S_i|/2$ for all i. Since we start with $|S_0| = 2^n$, we have $|S_{n+1}| \leq 1/2$ which implies $|S_{n+1}| = 0$, so we halt after at most $n + 1$ rounds. □

4.6 Learning circuits in P^{NP}

A classical result by Bshouty et al. [BCG+96] is concerned with learning Boolean circuits, when given access to an oracle for NP and an oracle that answers equivalence queries with respect to the unknown circuit C to be learned. In an *equivalence query* one supplies some circuit C' and receives an answer whether C and C' compute the same function. If the answer is negative the answer also includes a *counterexample* – an input x on which $C(x) \neq C'(x)$.

Bshouty et al. [BCG+96] present a randomized algorithm that achieves this goal. In a similar manner to the previous section, this learning algorithm may be also regarded as a non-randomized reduction to sampling. We thus can derandomize this algorithm using Theorem 3.10 and obtain:

Theorem 4.10. *If E^{NP} requires exponential size SV-nondeterministic circuits, then there is a deterministic procedure with access to an NP-oracle that learns an unknown Boolean circuit C of size s on n inputs in time poly(s, n) using equivalence queries.*

Proof. We use the notation $[y]$ to indicate the function computed by the Boolean circuit described by string y. Define the function $R : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}$ by $R(y, z) = [y](z)$.

The learning procedure is very similar to the algorithm in the proof of Theorem 4.9. The procedure operates in rounds. Initially, we set $i = 0$, and $S_0 = \{0,1\}^s$, and observe that S_0 is clearly recognized by a trivial circuit C_0. We now begin round 0.

In round i we do the following:

1. In P^{NP}, generate the S_i-conditional $(s, s^3, 1/4)$-discrepancy set $T_i \subseteq S_i$ (using Theorem 3.10). The size of T_i and the time to generate it are both at most poly$(|C_i|, s) =$ poly(s, i).

2. Make the equivalence query: "$\mathrm{maj}_{t \in T_i} R(t, z) \equiv C(z)$?" If the answer is YES, then we are done.

3. If the answer is NO, then we are given a counterexample z_i for which $\mathrm{maj}_{t \in T_i} R(t, z_i) \neq C(z_i)$.

4. Define $S_{i+1} = \{y : y \in S_i \wedge R(y, z_i) = C(z_i)\}$, and observe that S_{i+1} is recognized by a circuit C_{i+1} of size $O(s^2 + |C_i|)$.

5. Begin round $i + 1$.

As in the proof of Theorem 4.9 the main claim is that the number of rounds before completion is at most $O(s)$. At step 3, we claim that

$$\Pr_y[R(y, z_i) = C(z_i) | y \in S_i] \leq 3/4.$$

This is true because we know $\Pr_{t \in T_i}[R(t, z_i) = C(z_i) | t \in S_i] \leq 1/2$, and the circuit computing R with z_i hard-wired has size at most $O(s^2) < s^3$, and T_i is an S_i-conditional $(s, s^3, 1/4)$-discrepancy set, which implies $\Pr_y[R(y, z_i) = C(z_i) | y \in S_i] \leq 1/2 + \epsilon = 3/4$, as claimed.

Thus $|S_{i+1}| \leq (3/4)|S_i|$ for all i. We start with $|S_0| = 2^s$, and for all i, S_i is non-empty since it contains y for which $[y] = C$, so we must halt after at most $O(s)$ rounds with a positively answered equivalence query. $\square$

We remark that Theorem 4.9 and Theorem 4.10 are just two examples where a ZPP$^{\text{NP}}$ algorithm for sampling is used as a critical subroutine (see, e.g., the discussion in [BGP00] regarding applications in interactive proofs). Often this is the *only* randomness used in these procedures, and so conditional discrepancy sets suffice for derandomization in a variety of settings.

5 Overview of the techniques

In this section we present the main technical ideas in the proofs of the main theorems in an informal manner; the full proofs appear in later sections.

5.1 Proof of the downward collapse theorem

We show in Theorem 3.2 that for every sufficiently strong complexity class $\mathcal{C}$, if $\mathcal{C}$ is computable by small nonadaptive SAT-oracle circuits then $\mathcal{C}$ is computable by small SV-nondeterministic circuits. This certainly does *not* mean that one can always transform small nonadaptive SAT-oracle circuits into small SV-nondeterministic circuits. Note that the uniform versions of these classes are $\text{P}_{||}^{\text{NP}}$ and $\text{NP} \cap \text{coNP}$ and it is widely believed that $\text{P}_{||}^{\text{NP}} \not\subseteq \text{NP} \cap \text{coNP}$. More precisely, there are small nonadaptive SAT-oracle circuits for Satisfiability and we do not expect Satisfiability to have small SV-nondeterministic circuits, as this would mean that $\text{NP} \subseteq \text{coNP/poly}$ and collapse the polynomial hierarchy.

Indeed, this observation demonstrates the main problem we need to overcome. Whenever a nonadaptive SAT-oracle circuit calls its NP-oracle, it gets a result no matter whether the query asked is answered positively or negatively. An SV-nondeterministic circuit can attempt to simulate a nonadaptive SAT-oracle circuit by guessing which queries are answered positively, together with witnesses for those queries – in this way it can "verify" some queries that are answered positively. But it can not be sure that it has correctly guessed *all* of the positively answered queries, precisely because it is incapable of verifying negative answers (assuming $\text{NP} \not\subseteq \text{coNP/poly}$).

The main idea in the proof is that when the function to be computed is a low degree multivariate polynomial, a small SV-nondeterministic circuit *can* in fact verify negative answers, in an indirect way. It is known that every function in a sufficiently strong class $\mathcal{C}$ has a multivariate polynomial "low-degree extension" [BF90] that lies in the same class. Thus, we can without loss of generality concentrate on the case where the function we are trying to compute by an SV-nondeterministic circuit is a low degree multivariate polynomial.

We now describe the idea that exploits the low-degree extension[8]. We're given a small nonadaptive SAT-oracle circuit which computes some low degree multivariate polynomial $f : \mathbb{F}^d \to \mathbb{F}$ (for some field $\mathbb{F}$ of size q). For simplicity, let's assume that this circuit makes a single NP-query. We want to construct a small SV-nondeterministic circuit for f. For every input x in the domain of f, let $A(x)$ denote the answer to the NP-query asked on x. Let p denote the fraction of x's for which the query is answered positively. We hardwire p to our SV-nondeterministic circuit[9]. Now, on input x the new circuit passes a random low degree curve through x (we denote the degree of this curve by r). Except for x, the other q points on this curve are r-wise independent and therefore with high probability the fraction of points y on the curve for which $A(y) = 1$ is in the range $(p - \delta, p + \delta)$ for some small δ.[10] The circuit now guesses $(p - \delta)q$ points on the curve along with witnesses showing that the queries corresponding to these points are answered positively. The circuit assumes that these queries are answered positively and the queries for the remaining points on the curve are answered negatively. The critical observation is that this assumption can be incorrect on at most a 2δ fraction of the points on the curve. The circuit now simulates the nonadaptive SAT-oracle circuit (which makes no further NP queries) on all q points on the curve, and the final evaluations it receives differ from the correct evaluations on at most $2\delta q$ points. Finally, because the function f restricted to the curve is a low-degree polynomial, the circuit can run a decoding algorithm for Reed-Solomon codes [WB86] to correct the errors and obtain the correct answers for all points on the curve, and in particular the circuit obtains $f(x)$.

5.2 Building relative-error approximators

Our relative-error approximators build on a line of work which gives probabilistic algorithms that use an NP-oracle to approximately count NP-witnesses [Sto83, JVV86, BGP00] (for more information see the discussion in [BGP00]). Such algorithms are given a deterministic circuit A on n bits and wish to produce a relative approximation of the size of the set $S = \{x | A(x) = 1\}$. The algorithm presented in [BGP00] works by finding a hash function $h : \{0,1\}^n \to \{0,1\}^k$ with the property that for every image $y \in \{0,1\}^k$ the size of the preimage $S_y = \{x \in S | h(x) = y\}$ is roughly n^2, which implies that $|S|$ is approximately $n^2 2^k$.

To find such a hash function, one chooses a random hash function $h : \{0,1\}^n \to \{0,1\}^k$ from an n-wise independent hash family, and use the NP oracle to check whether there exists a $y \in \{0,1\}^k$ whose preimage has size greater than n^2. This is done for $k = 1, 2, 3, \ldots$, stopping with the first h that is good in the sense that there does not exist such a y whose preimage is "too large". By the pigeonhole principle, a good h does not exist for k such that $n^2 2^k < |S|$; for slightly larger k a random h from the n-wise independent hash family is good with high probability. Thus, the algorithm stops with the "correct" value of k, with high probability.

We would like to derandomize this procedure. Since it is not a decision problem we cannot use PRGs directly[11]. Instead we derandomize this procedure by using the particular way it operates (a general method

[8]A similar idea was used in [SU01] to build PRGs for nondeterministic circuits. It may also be viewed as a non-trivial "scaling down" of $\mathrm{EXP}^{\mathrm{NP}}_{||} \subseteq \mathrm{NEXP/poly} \cap \mathrm{coNEXP/poly}$ – a containment credited to Harry Buhrman on Lance Fortnow's weblog.

[9]The same idea was used to obtain the main result of [FF93].

[10]By choosing the degree r large enough we can show that there exist *fixed* points $v_1, \cdots, v_r \in F^d$ such that for every x the fraction of points y such that $A(y) = 1$ on the degree r curve that passes through $x; v_1, \cdots, v_r$ is in the range $(p - \delta, p + \delta)$. In the final construction we also hardwire the points $v_1, \cdots, v_r$ to the circuit.

[11]For the case of decision problems every probabilistic algorithm can be derandomized if one has a sufficiently strong pseudo-random generator. However, there are tasks (which are not decision problems) that can be easily solved by a probabilistic algorithm

that has been suggested by [KvM02] for such circumstances). Rather than choosing the hash functions randomly, we try all of the hash functions that are described by outputs of a PRG for nondeterministic circuits. For the "correct" k, one of the hash functions we try is good, because the generator fools the nondeterministic circuit which, given h, checks whether it is good. Having identified the "correct" k, we can now output an estimate of $|S|$. In the full argument, some additional care must be taken to obtain less-coarse approximations, and to ensure that the overall procedure runs in $\mathrm{P}_{||}^{\mathrm{NP}}$, rather than P^{NP}.

5.3 Constructing conditional discrepancy sets

An S-conditional discrepancy set for small circuits is a set $T \subseteq S$ such that no small (deterministic) circuit can distinguish a random element from T from a random element in S. This generalizes "regular" discrepancy sets for small circuits (for which the set S is simply $\{0,1\}^n$). Given a set S, encoded by a circuit A such that $S = \{x|A(x) = 1\}$, our goal is to output an S-conditional discrepancy set T.

As with relative-error approximation, our approach is based on algorithms which uses an NP-oracle to sample (or count) accepting inputs of A [Sto83, JVV86, BGP00]. Fix a hash function $h : \{0,1\}^n \to \{0,1\}^k$ which is good in the sense defined above. To sample a random element from S, one can choose a random image y, use the NP oracle to find all the preimages of y (there are approximately n^2 of them), and choose a random one.

Our procedure for producing conditional discrepancy sets is a derandomization of this algorithm. It relies on hardness for nondeterministic circuits, which by our results buys us a PRG for nonadaptive SAT-oracle circuits. We first find a good hash function h as explained above. Then, we include in the conditional discrepancy set T the preimages in S of *only* those y that are outputs of a PRG G for nonadaptive SAT-oracle circuits. We stress that using Theorem 4.4 we can (under the hardness assumption) compute the preimages making only nonadaptive NP oracle queries.

Here we make use of Theorem 4.4 to perform this step using only non-adaptive NP oracle access.

The proof that T is in fact an S-conditional discrepancy set is somewhat subtle. Given a (deterministic) circuit that distinguishes a random element in T from a random element in S, we need to construct a nonadaptive SAT-oracle circuit D that is a distinguisher for the PRG G, thus leading to a contradiction. Care is needed to ensure that the distinguisher D makes only non-adaptive NP oracle queries – and this is especially crucial here because a distinguisher that makes adaptive queries is not guaranteed to be fooled by the PRG G that is based on only an SV-nondeterministic hardness assumption.

6 Proof of Theorem 3.2

We begin with some definitions and preliminaries.

6.1 Preliminaries

Given a function $f : X \to Y$ and $S \subseteq X$ we use $f(S)$ to denote the (multi-)set $\{f(x)|x \in S\}$.

6.1.1 Low-degree polynomials

The low-degree extension of a function embeds the function in a low-degree polynomial.

---

and cannot be solved by a deterministic algorithm. For example, a probabilistic algorithm can easily produce a string with high Kolmogorov complexity whereas no deterministic algorithm can output such a string.

Definition 6.1 (low-degree extension). *Let $f : \{0,1\}^n \to \{0,1\}$ be a function, $\mathbb{F}_q$ the field with q elements, and h and d integers for which $h^d \geq 2^n$. Let H be a subset of $\mathbb{F}_q$ of size h, and let I be an efficiently computable injective mapping from $\{0,1\}^n$ to H^d.*

The low-degree extension of f with respect to q, h, d *is the (unique) d-variate polynomial $\hat{f} : \mathbb{F}_q^d \to \mathbb{F}$ with degree $h-1$ in each variable, for which $\hat{f}(I(x)) = f(x)$ for all $x \in \{0,1\}^n$ and $\hat{f}(v) = 0$ for $v \in (H^d \setminus Im(I))$.*

It is often helpful to think of field elements as binary strings of length $\log q$. From this viewpoint, $\hat{f}$ is a function from $d \log q$ bits to $\log q$ bits. We will often consider a version of the low degree extension which outputs a single bit. This *boolean version* of the low-degree extension is denoted $\hat{f}_{bool} : \{0,1\}^{d \log q + \log \log q} \to \{0,1\}$ and is defined by $\hat{f}_{bool}(x, i) = \hat{f}(x)_i$.

The following properties of low-degree extensions are trivial and standard:

Proposition 6.2 (properties of the low-degree extension). *For $\hat{f}$ and $\hat{f}_{bool}$ as defined above, the following hold:*

- *$\hat{f}$ has total degree hd, and*
- *$\hat{f}_{bool}$ is computable in time poly$(h^d, \log q, d)$ given oracle access to f.*

Complexity classes that allow low-degree extension (see Definition 3.1) contain the (boolean) low-degree extensions of every function in that class; Theorem 3.2 applies to all such classes.

Definition 6.3 (parametric curves). *Let $\mathbb{F}_q$ be the field with q elements, and let $f_1, f_2, \ldots f_q$ be an enumeration of the elements of $\mathbb{F}_q$. Given $v_1, v_2, \ldots, v_r \in \mathbb{F}_q^d$, for $r \leq q$, we define the* curve passing through *$v_1, v_2, \cdots, v_r$ to be the unique degree $r-1$ polynomial function $c : \mathbb{F}_q \to \mathbb{F}_q^d$ for which $c(f_i) = v_i$ for all i. A curve c is* one to one *if $i \neq j$ implies $c(f_i) \neq c(f_j)$.*

The function $\hat{f} \circ c$ is the *restriction of $\hat{f}$ to the curve c*. It is a low-degree univariate polynomial; in coding terms, it is a Reed-Solomon codeword.

Theorem 6.4 (decoding of Reed-Solomon codes [WB86]). *Let $\mathbb{F}_q$ be the field with q elements. Given t pairs (x_i, y_i) of elements of $\mathbb{F}_q$, there is a at most one polynomial $g : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most u for which $g(x_i) = y_i$ for at least a pairs, provided $a > (t+u)/2$. Furthermore, there is a polynomial time algorithm that finds g or reports that such a g does not exist.*

6.2 Random curves that pass through a fixed point

In this subsection we prepare some technical machinery needed for the proof of Theorem 3.2. We will repeatedly use the following tail-inequality for r-wise independent random variables:

Lemma 6.5 ([BR94]). *Let $r \geq 4$ be an even integer. Suppose $X_1, X_2, \ldots, X_q$ are r-wise independent random variables taking values in $[0,1]$. Let $X = \sum X_i$, and $A > 0$. Then:*

$$\Pr[|X - E[X]| \geq A] \leq 8 \cdot \left(\frac{r \cdot E[X] + r^2}{A^2} \right)^{r/2}.$$

We prove a technical lemma regarding the sampling properties of low-degree parametric curves. The points on a random degree r parametric curve are r-wise independent; a well-known consequence of this fact is that the points on such a curve are a good "oblivious sampler" (see the survey [Gol97]). This means that for any function $h : \mathbb{F}^d \to [0,1]$ the average of $h(x)$ over the points on a random curve is with high probability close to the average over the whole space. We show below that this holds even if an adversary gets to choose the first point on the curve, because the remaining points on the curve are still r-wise independent, and so it remains a good sampler.

We need the following notation:

Definition 6.6. *Let $W \subseteq Z$ be finite sets and let $h : Z \to [0,1]$ be an arbitrary function. The* average of h over W *is defined by:*

$$\mu_W(h) = \frac{1}{|W|} \sum_{i \in W} h(i)$$

We will use $c_{(x,v_1,v_2,\ldots,v_r)}$ to denote the curve passing through $x, v_1, v_2, \ldots v_r$ (see Definition 6.3). We require that $c_{(x,v_1,v_2,\ldots,v_r)}(0) = x$; i.e., the enumeration of the field elements in Definition 6.3 starts with 0. Also, below $\mathbb{F}_q$ is the field of size q, and $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$.

Lemma 6.7. *Let r be an integer for which $2 \le r < q$. For every point $x \in \mathbb{F}_q^d$, function $h : \mathbb{F}_q^d \to [0,1]$, and $\delta > 0$, the following hold:*

1. $\Pr_{v_1,\ldots,v_r \in \mathbb{F}_q^d}\left[\left|\mu_{c_{(x,v_1,\ldots,v_r)}(\mathbb{F}_q^*)}(h) - \mu_{\mathbb{F}_q^d}(h)\right| \ge \delta\right] \le 8 \cdot \left(\frac{2r}{(q-1)\delta^2}\right)^{r/2}$*, and*
2. $\Pr_{v_1,\ldots,v_r \in \mathbb{F}_q^d}\left[c_{(x,v_1,\ldots,v_r)}\text{ isn't one to one}\right] \le \frac{1}{q^{d-2}}$.

Proof. Fix x and h, and let $v_1, \ldots, v_r$ be chosen uniformly and independently from $\mathbb{F}_q^d$. Define random variables Y_a by $Y_a = c_{(x,v_1,\ldots,v_r)}(a)$. It is standard that for every $a \in F_q^*$, Y_a is uniformly distributed over $\mathbb{F}_q^d$, and that the random variables $\{Y_a\}_{a \in F_q^*}$ are r-wise independent. Now we define the random variables $R_a = h(Y_a)$. It follows that for every $a \in F_q^*$, $E[R_i] = \mu_{\mathbb{F}_q^d}(h)$, and that $\{R_a\}_{a \in F_q^*}$ are r-wise independent. Let $R = \sum_{a \in F_q^*} R_a$. We apply Lemma 6.5 with $A = |F_q^*|\delta = (q-1)\delta$ to conclude:

$$\Pr_{v_1,\cdots,v_r \in \mathbb{F}_q^d}\left[\left|\mu_{c_{(x,v_1,\ldots,v_r)}(\mathbb{F}_q^*)}(h) - \mu_{\mathbb{F}_q^d}(h)\right| \ge \delta\right] = \Pr[|R - E[R]| \ge A] \le 8\left(\frac{2r}{(q-1)\delta^2}\right)^{r/2}.$$

This proves (1). For (2), we observe that for every $a \neq a' \in \mathbb{F}_q$,

$$\Pr_{v_1,\ldots,v_r \in \mathbb{F}_q^d}[c_{(x,v_1,\ldots,v_r)}(a) = c_{(x,v_1,\ldots,v_r)}(a')] = \frac{1}{q^d},$$

and taking a union bound over all (at most q^2) such pairs yields the desired result. ☐

We will be interested in curves that are good samplers for k functions simultaneously. The following is a corollary of the above lemma; it is an easy application of a union bound:

Corollary 6.8. *Let r be an integer for which $2 \le r < q$. Let $h_1, h_2, \ldots, h_k$ be functions from $\mathbb{F}_q^d$ to $[0,1]$. For every point $x \in \mathbb{F}_q^d$ and $\delta > 0$, the probability over a random choice of points $v_1, \ldots, v_r \in \mathbb{F}_q^d$ that $c_{(x,v_1,\ldots,v_r)}$ is one-to-one and*

$$\left|\mu_{c_{(x,v_1,\ldots,v_r)}(\mathbb{F}_q^*)}(h_i) - \mu_{\mathbb{F}_q^d}(h_i)\right| < \delta$$

for all $1 \leq i \leq k$, *is at least*

$$1 - \left(8k\left(\frac{2r}{(q-1)\delta^2}\right)^{r/2} + \frac{1}{q^{d-2}}\right).$$

6.3 Proof of the downward collapse theorem

In this subsection we prove Theorem 3.2. We refer the reader to the informal description of the technique in section 5.1.

Let L be an arbitrary language in $\mathcal{C}$, and let $f : \{0,1\}^n \to \{0,1\}$ be the restriction of (the characteristic function of) L to inputs of length n. Throughout the proof we assume that n is sufficiently large, $n \leq s(n) \leq 2^n$, and that $s(O(n)) \leq s(n)^{O(1)}$.

Let $\hat{f}$ be the low-degree extension of f with respect to parameters q, h, d chosen as follows (they are expressed in terms of a fourth parameter r):

- $r = 2(n + \log(32s(n)^5))$
- $h = (4r)^2(9s(n))^4$
- $d = \lceil n/\log h\rceil + 3$
- q smallest prime power larger than $9hdr$.

Note that $\mathcal{C}$ allows low-degree extension, and so by Proposition 6.2, the function family consisting of (boolean versions of) the low-degree extensions of L for each input length, with parameters as defined above, lies in $\mathcal{C}$.

Thus, by the hypothesis of the theorem, $\hat{f}_{bool}$ has a nonadaptive SAT-oracle circuit of size $s(n')$, where $n' = \log(q^d)+\log(q) = O(n)$ is the input length of $\hat{f}_{bool}$. We will construct a *probabilistic* SV-nondeterministic circuit C' computing $\hat{f}_{bool}$ of size $s' = s(n')^c$, for a constant c (it will be clear in the exposition below what is meant by a "probabilistic SV-nondeterministic circuit"). We will then transform C' into an SV-nondeterministic circuit C'' computing f by fixing a "good" random string, and using the function I that accompanies the low-degree extension (recall Definition 6.1). The resulting circuit C'' will have size $s(n')^c + poly(n)$. Since $s(n')^c = s(O(n))^c = s(n)^{O(1)}$, we will conclude that L has circuits of size $s(n)^{O(1)}$. As L was arbitrary, this will prove the theorem.

Let C_{pre}, C_{post} be the Boolean circuits that describe the nonadaptive SAT-oracle circuit of size $s(n')$ that computes $\hat{f}_{bool}$ (recall Definition 2.3). With $\log q$ parallel copies of C_{pre} and C_{post}, we can construct a nonadaptive SAT-oracle circuit with $\log q$ outputs that computes $\hat{f}$. Let $Q_1(x), \ldots, Q_k(x)$ and $A_1(x), \ldots, A_k(x)$ be the queries and answers associated with this circuit, respectively, on input $x \in \mathbb{F}_q^d$. Without loss of generality we assume that exactly k queries are made on every input x. We define $p_i = \mu_{\mathbb{F}_q^d}(A_i)$.

We focus first on constructing C', the probabilistic SV-nondeterministic circuit. Circuit C' makes use of C_{pre} and C_{post}, as well as $p_1, p_2, \ldots, p_k$ as non-uniform advice. We set $\delta = 1/(9k)$. On input (x, b), circuit C' wants to compute $\hat{f}_{bool}(x, b)$; it performs the following steps:

- Pick $v_1, v_2, \ldots, v_r \in \mathbb{F}_q^d$ uniformly at random, and set $x_a = c_{(x,v_1,v_2,\ldots,v_r)}(a)$, so the x_a are the q points along a random curve passing through $x, v_1, v_2, \ldots v_r$. Simulate C_{pre} to compute queries $Q_i(x_a)$ for $1 \leq i \leq k$ and $a \in \mathbb{F}_q^*$.

- Set $n_i = \lfloor (p_i - \delta)(q-1) \rfloor$. For $1 \le i \le k$, guess $z_i \in \{0,1\}^{\mathbb{F}_q^*}$ with exactly n_i ones, and strings $\{w_{i,a}\}_{a \in \mathbb{F}_q^*}$.
- For $1 \le i \le k$ and $a \in \mathbb{F}_q^*$, check that $(z_i)_a = 1$ implies $w_{i,a}$ is a witness that query $Q_i(x_a)$ is answered positively; otherwise, set the **flag** output to 0 and halt.
- Compute $y_a = C_{post}(x_a, (z_1)_a, (z_2)_a, \ldots, (z_k)_a)$ for $a \in \mathbb{F}_q^*$.
- Run the algorithm of Theorem 6.4 on the $q-1$ pairs (f_a, y_a) with $u = hdr$ to obtain a polynomial $g : \mathbb{F}_q \to \mathbb{F}_q$ of degree u (if one exists). Set the **value** output to the b-th bit of $g(0)$ (or 0 if g does not exist), and set the **flag** output to 1.

The following claim will allow us to fix the coin-flips of circuit C', described above, to get an SV-nondeterministic circuit computing f.

Claim 6.8.1. *For every $x \in \mathbb{F}_q^d$ and $b \in [\log q]$, with probability at least $1 - \frac{2^{-n}}{2\log q}$ over the choice of $v_1, \ldots, v_r$, the following two conditions hold:*

1. *For all guesses $z_i, w_{i,a}$ for which the* **flag** *output is set to one, the* **value** *output is $\hat{f}_{bool}(x, b)$.*
2. *There exist guesses $z_i, w_{i,a}$ such that the* **flag** *output is set to one.*

Proof. Fix an $x \in \mathbb{F}_q^d$. We apply Corollary 6.8 to conclude that the probability over a random choice of points $v_1, \ldots, v_r \in \mathbb{F}_q^d$ that

$$c_{(x,v_1,\ldots,v_r)} \text{ is one-to-one and } \forall 1 \le i \le k \left| \mu_{c_{(x,v_1,\ldots,v_r)}(\mathbb{F}_q^*)}(A_i) - \mu_{\mathbb{F}_q^d}(A_i) \right| < \delta \tag{3}$$

is at least

$$1 - \left(8k \left(\frac{2r}{(q-1)\delta^2} \right)^{r/2} + \frac{1}{q^{d-2}} \right).$$

By our choice of parameters:

$$\left(8k \left(\frac{2r}{(q-1)\delta^2} \right)^{r/2} + \frac{1}{q^{d-2}} \right) \le 8s(n) \log q \left(\frac{1}{2} \right)^{r/2} + \frac{1}{q^{d-2}} \le \frac{2^{-n}}{4\log q} + \frac{2^{-n}}{4\log q} \le \frac{2^{-n}}{2\log q}.$$

The first inequality it true because $k \le s(n)\log q$, $\delta^{-2} = (9k)^2 \le (9s(n)\log q)^2$ and

$$(q-1)/\log^2 q \ge \sqrt{q} \ge \sqrt{h} \ge (4r)(9s(n))^2$$

(for sufficiently large q). The second inequality follows from our choice of r and d, and the fact that $\log q = O(n) \le s(n)^2$ (for sufficiently large q).

We will show that whenever (3) holds, the two items in the claim hold. We begin with the second item. Since (3) holds, for each i we know that there are at least n_i distinct indices for which $A_i(x_a) = 1$; we choose z_i to be a string with ones in exactly n_i of these indices. For each index a for which $(z_i)_a = 1$, there is a witness $w_{i,a}$ showing that query $Q_i(x_a)$ is answered positively (since $A_i(x_a) = 1$). Thus there exists a choice of the $z_i, w_{i,a}$ for which the **flag** output is set to one.

Now, we turn to the first item. Once the verification in the third bullet above is complete, we know that for all i, and all $a \in F_q^*$, $(z_i)_a = 1$ implies $A_i(x_a) = 1$, and that there are at least n_i such a for which

$(z_i)_a = 1$. We also know, by (3), that the number of a for which $A_i(x_a) = 1$ is at most $\lceil (p_i + \delta)(q-1) \rceil$. Thus we can bound the number of "errors attributable to query i" as follows:

$$| \left\{ a : a \in \mathbb{F}_q^*, A_i(x_a) \neq (z_i)_a \right\} | \leq \lceil (p_i + \delta)(q-1) \rceil - \lfloor (p_i - \delta)(q-1) \rfloor \leq 2\delta q,$$

and the number of "errors" overall as follows:

$$| \left\{ a : a \in \mathbb{F}_q^* \text{ for which } \exists i \; A_i(x_a) \neq (z_i)_a \right\} | \leq 2\delta q k.$$

For every a that is not an "error," $y_a = \hat{f}(x_a)$. We conclude that for at least $(q-1) - 2\delta q k = (1-2\delta k)q - 1$ of the pairs (a, y_a), we have $y_a = p(a)$, where $p(w)$ is the degree hdr "restriction to the curve" $p(w) = \hat{f} \circ c_{(x_1, v_1, v_2, \ldots, v_r)}(w)$.

If the number of pairs that agree with $p(w)$ is greater than $(q - 1 + hdr)/2$, then the algorithm of Theorem 6.4 returns $p(w)$, and our circuit outputs the b-th bit of $p(0) = \hat{f}(x)$ as desired. Thus to conclude the proof we verify that

$$(1 - 2\delta k)q - 1 = \frac{7/9}{q} - 1 > \frac{q - 1 + hdr}{2},$$

which holds by our choice of q. □

Now, recall that the low-degree extension is accompanied by a polynomial-time computable function I from $\{0,1\}^n$ into $\mathbb{F}_q^d$. Consider the set of inputs to C' given by

$$S = \{(x, b) : x \in I(\{0,1\}^n), b \in [\log q]\}$$

and note that $|S| = (\log q)2^n$. Thus there must be a fixing of the coin-flips of C' so that the two statements in the above claim hold for all inputs in S.

Our SV-nondeterministic circuit C'' computing f is built as follows:

- on input $y \in \{0,1\}^n$, compute $x = I(y)$
- use circuit C' with the "good" random coin-flips hardwired to compute $\hat{f}_{bool}(x, b)$ for every $b \in [\log q]$.
- these $\log q$ bits give us $\hat{f}(x) = \hat{f}(I(y)) = f(y)$. Output $f(y)$.

Because non-adaptive queries to an SV-nondeterministic circuit may be simulated by an SV-nondeterministic circuit, the resulting circuit C'' is an SV-nondeterministic circuit. Finally, we can verify that its size is poly(n) + $s(n')^c$ for some constant c. This concludes the proof of Theorem 3.2.

We remark that in the proof above we used Theorem 6.4 to decode Reed-Solomon codes by an efficient *deterministic* procedure. However, in our setup we are allowed to use an efficient SV-nondeterministic procedure for decoding (as we are shooting to construct an SV-nondeterministic circuit). And, an efficient deterministic encoding algorithm immediately induces an efficient SV-nondeterministic decoding procedure by guessing the appropriate codeword and verifying that it is indeed close to the given word.

7 Proofs of Theorem 3.6 and Theorem 3.10

We begin with a few preliminaries that will be needed later.

7.1 Preliminaries

First, we show (as in [BGP00]) that the preimages of a random n-wise independent hash function partition an arbitrary set S well:

Lemma 7.1. *Let $H_{n,k}$ be an n-wise independent family of hash functions mapping n bits to k bits, and let $S \subseteq \{0,1\}^n$. Then for every $1 \geq \delta > 0$, and sufficiently large n:*

$$\Pr_{h \in H_{n,k}}\left[\exists y \textit{ for which } |\{x : h(x) = y \wedge x \in S\}| > (1+\delta)\frac{|S|}{2^k}\right] \leq 1/2,$$

provided $2^k \leq \delta^2 n^{-3}|S|$.

Proof. Fix $y \in \{0,1\}^k$, and let I_x be the indicator random variable for the event $h(x) = y$. Notice that $\mathrm{E}[I_x] = 2^{-k}$ and that the I_x are n-wise independent. Define $I = \sum_{x \in S} I_x$; we have $\mathrm{E}[I] = |S|2^{-k}$ by linearity of expectation. Applying Lemma 6.5, we get:

$$\begin{aligned}
\Pr\left[|\{x : h(x) = y \wedge x \in S\}| > (1+\delta)\frac{|S|}{2^k}\right] &\leq \Pr\left[I - \mathrm{E}[I] \geq \delta\mathrm{E}[I]\right] \leq 8 \cdot \left(\frac{n\mathrm{E}[I] + n^2}{(\delta\mathrm{E[I]})^2}\right)^{n/2} \\
&\leq 8 \cdot \left(\frac{2n}{\delta^2\mathrm{E[I]}}\right)^{n/2} \leq 8 \cdot \left(\frac{2}{n^2}\right)^{n/2} < 2^{-(n+1)}.
\end{aligned}$$

Applying a union bound over all $2^k < 2^n$ different y, we obtain the stated result. $\square$

We will also need the following fact about composing functions computable with *non-adaptive* NP oracle access:

Lemma 7.2. *Let $f = \{f_n\}$ and $g = \{g_n\}$ be length-preserving function families in FTIME$(t(n))_{||}^{NP}$ and FTIME$(s(n))_{||}^{NP}$ respectively. Then the function family $(f \circ g)$ defined by $(f \circ g)(x) = f(g(x))$ is in FTIME(poly$(t(n)s(n)n))_{||}^{NP}$.*

Proof. We are given an input x of length n, and we wish to compute $f(g(x))$. Let M_f and M_g be the deterministic oracle Turing Machines associated with f and g.

We describe how to determine if the j-th bit of $f(g(x))$ is 1, using non-adaptive NP oracle queries. Suppose we know that out of all of the non-adaptive NP oracle queries $M_g(x)$ makes, exactly n_g are answered positively; similarly, suppose that we know that on input $g(x)$, out of all of the non-adaptive NP oracle queries $M_f(g(x))$ makes, exactly n_f are answered positively. Then with a *single* NP oracle query, we can guess:

- an n-bit string y, and
- which n_g oracle queries in the computation $M_g(x)$ are answered positively, and which n_f oracle queries in the computation $M_f(y)$ are answered positively, and
- witnesses for the n_g positively answered oracle queries made by $M_g(x)$, and witnesses for the n_f positively answered oracle queries made by $M_f(y)$,

and verify that the witnesses are all valid, that $M_g(x)$ with the guessed yes/no answers outputs y, and that $M_f(y)$ with the guessed yes/no answers outputs a string whose j-th bit is 1.

Assuming n_g and n_f are correct, this NP query will be answered positively iff the j-th bit of $f(g(x)) = 1$: it is easy to see that n_g being correct means that the only valid witnesses will have $y = g(x)$, and that n_f being correct means that the only valid witnesses correctly simulate $M_f(y)$ and thus are accepted iff the j-th bit of $f(y)$ is 1.

We could try making this single NP oracle query for each value of n_f and n_g, in parallel. The only problem is that we don't know which answer is the correct one. This can easily be fixed by making the following two NP oracle queries for each possible value of n_f and n_g.

1. guess which n_g oracle queries in the computation $M_g(x)$ are answered positively, together with witnesses for them, and verify that the witnesses are all valid.

2. guess an n-bit string y, which n_g oracle queries in the computation $M_g(x)$ are answered positively, which n_f oracle queries in the computation $M_f(y)$ are answered positively, and witnesses for all the positively answered queries; verify that the witnesses are all valid and that $M_g(x)$ with the guessed yes/no answers outputs y.

It is easy to see that the largest value of n_g for which the first query above is answered positively is the correct value. Then, the largest value of n_f (paired with the correct value of n_g) for which the second query above is answered positively is the correct value for n_f. From this information we know which of the original set of queries to trust, and we successfully determine the j-th bit of $f(g(x))$.

Since $t(n)$ is an upper bound on n_f and $s(n)$ is an upper bound on n_g, the procedure above entails $3(t(n) + 1)(s(n) + 1)$ non-adaptive NP oracle queries, and we perform it in n times in parallel to compute each of the n output bits of $f(g(x))$. Overall the running time is poly$(s(n)t(n)n)$ as claimed. □

Finally, we will use the following variant of Theorem 4.2 several times below.

Lemma 7.3 ([KvM02]). *There exist constants $\gamma, c > 0$, for which the following holds for all sufficiently large t: given the truth table T of a function on t bits that cannot be computed by non-adaptive SAT-oracle circuits of size $2^{\gamma t}$ there is a polynomial-time procedure that produces a $(2^{ct}, 2^{-ct})$ discrepancy set for non-adaptive SAT-oracle circuits of size 2^{ct}.*

7.2 The main lemma

The main procedure that is used in the proofs of Theorem 3.10 and Theorem 3.6 is encapsulated in the next lemma. It takes a circuit C that accepts a subset S of $\{0,1\}^n$, and outputs a hash function from n bits to k bits whose preimages partition S nearly evenly. Additionally, it outputs an S-conditional discrepancy set, in implicit form.

Lemma 7.4. *There is a function family that takes as input:*

- *a parameter δ such that $1/32 > \delta > 0$, and*
- *a circuit C on n bits that accepts at least $16\lceil\delta^{-2}n^3\rceil$ inputs, and*
- *an integer s, and*
- *the truth table T of a function on $t = O(\log|C|, \log n, 1/\delta, \log s)$ bits that cannot be computed by non-adaptive SAT-oracle circuits of size $2^{\gamma t}$ for some constant $\gamma > 0$,*

and outputs:

- *an integer* k*, and*
- *a hash function* $h : \{0,1\}^n \to \{0,1\}^k$*, and*
- *an integer* B *with* $B = poly(n, 1/\delta)$*, and*
- *a multiset* R

for which the following hold:

- $\forall y \in \{0,1\}^k \quad |\{x : h(x) = y \wedge C(x) = 1\}| \leq B$*, and*
- $\frac{2^k B}{1+2\delta} \leq |C^{-1}(1)|$*, and*
- *the multiset* $S = \{x : h(x) \in R \wedge C(x) = 1\}$ *is a* $C^{-1}(1)$*-conditional* $(n, s, 3\delta)$*-discrepancy set*[12].

This function family is in FTIME$(2^{O(t)})^{NP}_{||}$.

The proof appears in subsection 7.3. We first show how this lemma easily gives us both Theorem 3.6 and Theorem 3.10.

Proof of Theorem 3.6. We are given a circuit A on n bits, and $\epsilon > 0$. Set $\delta = \epsilon/(2 - 2\epsilon)$ and set t as in the statement of Lemma 7.4. We describe our procedure in several steps, and then apply Lemma 7.2 to assemble them into a single procedure that uses non-adaptive NP-oracle access.

- We check whether $|A^{-1}(1)| < 16\lceil \delta^{-2} n^3 \rceil$, and if so, we compute its size exactly using that many parallel NP queries, and set $\rho = \Pr_x[A(x) = 1]$ which is exact in this case.
- We are assuming that $\mathrm{E}^{\mathrm{NP}}_{||}$ requires exponential size SV-nondeterministic circuits. By Corollary 3.3, $\mathrm{E}^{\mathrm{NP}}_{||}$ also contains languages that require exponential size non-adaptive SAT-oracle circuits. Let L be such a language in $\mathrm{E}^{\mathrm{NP}}_{||}$. We produce the truth table T of L restricted to length t inputs. Since $L \in \mathrm{E}^{\mathrm{NP}}_{||}$ this procedure is in FTIME$(2^{O(t)})^{\mathrm{NP}}_{||}$.
- Apply the function family of Lemma 7.4, with inputs A, δ, n, T. This produces output k, h, B and R, and runs in time FTIME$(2^{O(t)})^{\mathrm{NP}}_{||}$.
- The resulting output has integers k and B for which

  $$\frac{2^k B}{1 + 2\delta} \leq |A^{-1}(1)| \leq 2^k B.$$

  We can then output $\rho = (1 - \epsilon)(2^k B)/(2^n)$, and the above equation implies:

  $$(1 - \epsilon) \Pr_x[A(x) = 1] \leq \rho \leq \Pr_x[A(x) = 1]$$

  as required.

[12] Recall that R is a multiset, and we intend each $x \in S$ to be reproduced as many times as $h(x)$ appears in R.

After applying Lemma 7.2, the overall running time of the procedure is polynomial in $|A|, n$ and $1/\epsilon$ and it uses only non-adaptive NP oracle access. $\square$

Proof of Theorem 3.10. We are given a circuit A on n bits, an integer s, and $\epsilon > 0$, and we want to produce a $A^{-1}(1)$-conditional (n, s, ϵ)-discrepancy set. Set $\delta = \epsilon/3$ and set t as in the statement of Lemma 7.4. We describe our procedure in several steps, and then apply Lemma 7.2 to assemble them into a single procedure that uses non-adaptive NP-oracle access.

- We check whether $|A^{-1}(1)| < 16\lceil\delta^{-2}n^3\rceil$, and if so, we compute the entire set $A^{-1}(1)$, which is trivially an $A^{-1}(1)$-conditional (n, s, ϵ)-discrepancy set. By Theorem 4.4, we can do this in poly$(n, 1/\delta)$ time with nonadaptive NP queries (since we are assuming that $\mathrm{E}_{||}^{\mathrm{NP}}$ requires exponential size SV-nondeterministic circuits).
- We are assuming that $\mathrm{E}_{||}^{\mathrm{NP}}$ requires exponential size SV-nondeterministic circuits. By Theorem 3.2, $\mathrm{E}_{||}^{\mathrm{NP}}$ also contains languages that require exponential size non-adaptive SAT-oracle circuits. Let L be such a language in $\mathrm{E}_{||}^{\mathrm{NP}}$. We first produce the truth table T of L restricted to length t inputs. Since $L \in \mathrm{E}_{||}^{\mathrm{NP}}$ this procedure is in $\mathrm{FTIME}(2^{O(t)})_{||}^{\mathrm{NP}}$.
- Apply the function family of Lemma 7.4, with inputs A, δ, s, T. This produces output k, h, B and R, and runs in time $\mathrm{FTIME}(2^{O(t)})_{||}^{\mathrm{NP}}$.
- Finally, we produce from R an enumeration of the multiset $S = \{x : h(x) \in R \wedge A(x) = 1\}$. This can be accomplished by making queries of the form "Is there a set S_i of size i for which $S_i \subseteq \{x : h(x) \in R \wedge A(x) = 1\}$?" for each i up to $2^{O(t)}B$ (which is an upper bound on $|S|$). By Theorem 4.4, we can actually produce such sets S_i using non-adaptive NP oracle queries, and we find all of the S_i in parallel. The largest set identified contains all of the distinct elements of the multiset S, and we can duplicate them as needed according to the multiplicity of their image (under h) in R. This multiset is S, the desired $A^{-1}(1)$-conditional (n, s, ϵ)-discrepancy set.

After applying Lemma 7.2, the overall running time of the procedure is polynomial in $|A|, n, s$ and $1/\epsilon$ and it uses only non-adaptive NP oracle access.

If we assume instead that E^{NP} requires exponential-size SV-nondeterministic circuits then step 2 runs in $\mathrm{FTIME}(2^{O(t)})^{\mathrm{NP}}$ and the first and last step can use an NP oracle adaptively to find witnesses in the usual way. In this case the procedure has the same overall running time but uses adaptive NP oracle access. $\square$

7.3 Proof of Lemma 7.4

Set $N = 8\lceil\delta^{-2}n^3\rceil$. For each $k = 1, 2, \ldots, n$, let $H_{n,k}$ be an n-wise independent family of hash functions mapping n bits to k bits. For all $1 \leq k \leq n$ and all $0 \leq e < N$, given a description of some $h \in H_{n,k}$ (which, using standard constructions, requires poly(n) bits) we can test if

$$\exists y \text{ for which } |\{x : h(x) = y \wedge C(x) = 1\}| > (N + e) \tag{4}$$

in nondeterministic time $m = \text{poly}(N, |C|)$.

Using Lemma 7.3, we produce from T a $(m, 1/4)$-discrepancy set $U \subseteq \{0,1\}^m$ for non-adaptive SAT-oracle circuits of size m. Let M_k be an efficiently computable mapping from strings of length m to $H_{n,k}$

such that M_k is uniform on $H_{n,k}$ when its input is chosen uniformly (by which we may view the strings in $\{0,1\}^m$ as "descriptions" of members of the hash family $H_{n,k}$). For all pairs (k,e) with $k = 1, 2, \ldots n$, $e = 0, 1, \ldots N-1$, we test whether Eq. (4) holds for all hash functions $\{h = M_k(u) : u \in U\}$. Each such test entails $|U|$ parallel NP queries, and there are nN tests performed, in parallel. We label each of these tests with a pair (k,e), and order the pairs lexicographically (with k changing the slowest).

Now, we select the *first* (k,e) in the lexicographic order for which the test fails, together with the hash function $h = M_k(u)$ that witnesses that failure, i.e., h for which:

$$\forall y \ \ |\{x : h(x) = y \wedge C(x) = 1\}| \leq (N+e). \tag{5}$$

We set $B = (N+e)$.

Finally, let s' be some fixed polynomial in $B, |C|, s$ to be determined later. Using Lemma 7.3, we produce from T a (k,δ)-discrepancy set $R \subseteq \{0,1\}^k$ for non-adaptive SAT-oracle circuits of size m. We output k, the hash function h, the integer B, and the multiset R.

We break the remainder of the proof into two halves; the first verifies that h, k, and B satisfy the properties stated in the Lemma, and the second verifies that R implicitly defines a conditional discrepancy set as claimed in the Lemma.

7.3.1 First half: the hash function h, k, and B

In this half of the proof we show that the integer k, the hash function h, and the integer $B = (N+e)$ satisfy the properties stated in the Lemma.

Observe that there is a unique pair (k^*, e^*) with $k^* \in \{1, 2, \ldots, n\}$ and $e^* \in \{0, 1, \ldots, N-1\}$ for which

$$N + e^* \leq \frac{|C^{-1}(1)|}{2^{k^*}} < N + e^* + 1, \tag{6}$$

which can be seen by choosing $k^* = \lfloor \log_2(|C^{-1}(1)|/N) \rfloor$, and then e^* to satisfy the above inequalities.

We have three claims; the first says that for all (k,e) before (k^*,e^*) in the lexicographic order the test must succeed, and the next two show that for some (k,e) not too much beyond (k^*,e^*) in the lexicographic order, the test must fail. In this way we obtain an "approximation" of (k^*,e^*). Observe that by our choice of k^*, $|C^{-1}(1)|/2^{k^*}$ lies between N and $2N$. Thus k gives us an approximation to within granularity 2; determining e as well gives us a finer approximation, to within granularity $(1+2\delta)$.

Claim 7.4.1. *For all (k,e) such that $2^k(N+e) < 2^{k^*}(N+e^*)$, Eq. (4) holds for all hash functions $\{h = M_k(u) : u \in U\}$.*

Proof. For every hash function $h : \{0,1\}^n \to \{0,1\}^k$, by the pigeonhole principle, one of the 2^k disjoint sets $S_y = \{x : h(x) = y \wedge C(x) = 1\}$ has size greater than $(N+e)$, because

$$\sum_y |S_y| = |C^{-1}(1)| \geq 2^{k^*}(N+e^*) > 2^k(N+e).$$

□

Claim 7.4.2. *The first (k,e) in the lexicographic order for which $2^k(N+e) > (1+\delta)2^{k^*}(N+e^*+1)$ satisfies*

$$2^k(N+e) - 2^k \leq (1+\delta)2^{k^*}(N+e^*+1) \tag{7}$$

Proof. Since (k, e) is the *first* such pair, we know that the previous pair (k', e') in the lexicographic order fails to satisfy $2^k(N + e) > (1 + \delta)2^{k^*}(N + e^* + 1)$; i.e., it holds that

$$2^{k'}(N + e') \leq (1 + \delta)2^{k^*}(N + e^* + 1).$$

If $e \geq 1$, then $(k', e') = (k, e - 1)$, and so the left-hand-side equals $2^k(N + e) - 2^k$; if $e = 0$, then $(k', e') = (k - 1, N - 1)$, and so the left-hand-side equals $2^{k-1}(2N - 1) = 2^k(N + e) - 2^{k-1}$. In both cases, Eq. (7) follows. □

Claim 7.4.3. *For the first (k, e) in the lexicographic order for which $2^k(N + e) > (1 + \delta)2^{k^*}(N + e^* + 1)$, it is* not *the case that Eq. (4) holds for all hash functions $\{h = M_k(u) : u \in U\}$.*

Proof. Using Eq. (6) and the premise of the Claim, we have

$$(N + e) > (1 + \delta)2^{k^*}(N + e^* + 1)/2^k \geq (1 + \delta)|C^{-1}(1)|/2^k$$

and thus, using Lemma 7.1,

$$\begin{aligned} &\Pr_{h \in H_{n,k}}[\exists y \text{ for which } |\{x : h(x) = y \wedge C(x) = 1\}| > (N + e)] \\ &\quad \leq \Pr_{h \in H_{n,k}}[\exists y \text{ for which } |\{x : h(x) = y \wedge C(x) = 1\}| > (1 + \delta)|C^{-1}(1)|/2^k] \leq 1/2, \end{aligned}$$

provided that $2^k \leq \delta^2 n^{-3}|C^{-1}(1)|$ (so that Lemma 7.1 applies). In other words, subject to this condition, Lemma 7.1 states that Eq. (4) holds for at most half of the $h \in H_{n,k}$. Since U is a discrepancy set, it must be that for some $u \in U$, Eq. (4) does not hold for $h = M_k(u)$.

We just need to check that the condition required for Lemma 7.1 is satisfied. This follows from Claim 7.4.2. The right-hand-side of Eq. (7) is at most $4|C^{-1}(1)|$ by Eq. (6), while the left-hand-side is at least $2^k(N - 1)$, which is at least $2^k(4\delta^{-2}n^3)$ by our choice of N. We thus have $2^k(4\delta^{-2}n^3) \leq 4|C^{-1}(1)|$, as required. □

Finally, we verify that $B = (N + e)$ satisfies the statement of the Lemma. Using Eq. (6) and Eq. (7), we get:

$$2^k(N + e) \leq (1 + \delta)|C^{-1}(1)| + 2 \cdot 2^{k^*} + 2^k.$$

Now, Eq. (7) implies $2^k < 16 \cdot 2^{k^*}$, and Eq. (6) gives $2^{k^*} \leq |C^{-1}(1)|/N$. Lastly, $1/N \leq \delta^2$ (for $n > 16$). Therefore the right hand side is at most $(1 + \delta + 18\delta^2)|C^{-1}(1)|$. Since $\delta < 1/18$, we conclude that

$$\frac{2^k(N + e)}{1 + 2\delta} \leq |C^{-1}(1)|,$$

as required.

7.3.2 Second half: the multiset R

In this second half of the proof, we show that

$$S = \{x : h(x) \in R \wedge C(x) = 1\}$$

is a $C^{-1}(1)$-conditional $(n, s, 3\delta)$-discrepancy set as required by the statement of the Lemma.

Suppose for the purpose of contradiction that there is a distinguisher $f : \{0,1\}^n \to \{0,1\}$ computable by a size s circuit for which

$$\left|\Pr_x[f(x) = 1|C(x) = 1] - \Pr_{t\in S}[f(t) = 1|C(t) = 1]\right| > 3\delta. \tag{8}$$

We use f to describe a distinguisher $g : \{0,1\}^k \to \{0,1\}$ computable by a size s' non-adaptive SAT-oracle circuit that "catches" the discrepancy set R. On input $y \in \{0,1\}^k$, g uses B *non-adaptive* NP queries to determine $\ell_y = |\{x : h(x) = y \wedge C(x) = 1 \wedge f(x) = 1\}|$ (which is guaranteed to be at most B), and g then outputs 1 with probability ℓ_y/B (and 0 with the remaining probability).

We know that $2^k B \leq (1 + 2\delta)|C^{-1}(1)|$. Thus

$$\Pr_x[f(x) = 1|C(x) = 1] = \frac{\sum_y \ell_y}{|C^{-1}(1)|} \leq \frac{(1+2\delta)\sum_y \ell_y}{2^k B} = \frac{1+2\delta}{2^k}\sum_y \frac{\ell_y}{B} = (1+2\delta)\Pr_y[g(y) = 1]$$

We also know using Eq. (5), that $|S| \leq B|R|$. Thus

$$\Pr_{r\in R}[g(r) = 1] = \frac{1}{|R|}\sum_{r\in R}\frac{\ell_r}{B} \leq \frac{\sum_{r\in R}\ell_r}{|S|} = \Pr_{t\in S}[f(t) = 1|C(t) = 1].$$

We may assume that Eq. (8) holds without the absolute value, by complementing f if necessary. Then we get:

$$\Pr_{r\in R}[g(r) = 1] \leq \Pr_{t\in S}[f(t) = 1|C(x) = 1] < \Pr_x[f(x) = 1|C(x) = 1] - 3\delta \leq (1+2\delta)\Pr_y[g(y) = 1] - 3\delta$$

and so g distinguishes a random element from R from a truly random element with advantage greater than δ. We may fix g's random coins to preserve this advantage, and notice that g is computable by a size $s' = \text{poly}(B, |C|, s)$ non-adaptive SAT-oracle circuit. This contradicts the fact that R is a discrepancy set for size s' non-adaptive SAT-oracle circuits, and so S must indeed be an $C^{-1}(1)$-conditional $(n, s, 3\delta)$-discrepancy set, as desired.

This concludes the proof of Lemma 7.4.

8 Conclusions and open problems

All known "hardness versus randomness tradeoffs" work by using a hard function to construct a PRG that derandomizes the given probabilistic procedure. The proofs show that if the derandomization fails, this probabilistic procedure can be used as a subroutine to efficiently compute the supposedly hard function, which is a contradiction. One consequence of this type of argument is that to derandomize some class of probabilistic procedures $\mathcal{A}$, one requires a function that is hard for procedures that are *at least as strong* as $\mathcal{A}$. This paper gives several results that break this "barrier" by derandomizing "strong" classes using "weak" lower bounds. The most striking result in this vein is perhaps Theorem 4.9. Since it is known that $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{S}_2^{\mathrm{P}}$ (in other words that $\mathrm{S}_2^{\mathrm{P}}$ is strong enough to simulate *adaptive* NP-queries) it is highly unlikely that $\mathrm{S}_2^{\mathrm{P}}$ is computable by small nondeterministic circuits, and yet we show that $\mathrm{S}_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}}$ using "only" hardness for nondeterministic circuits.

One can ask how far this "weakening" of hardness assumptions can go. For example we do not know whether the existence of relative error approximators, or conditional discrepancy set generators, imply the nondeterminstic hardness assumption that we have used to construct them in this paper. The standard argument that shows that "pseudorandomness entails hardness" only gives hardness for *deterministic* circuits. Is it possible to construct these objects using a weaker hardness assumption? Constructing them from hardness for deterministic circuits would have some interesting consequences, like placing approximate counting in $\mathrm{ZPP}^{\mathrm{NP}}$ unconditionally.

Our downward collapse theorem states that for every sufficiently strong class $\mathcal{C}$ if $\mathcal{C}$ has small nonadaptive SAT-oracle circuits then C has small SV-nondeterministic circuits. A very natural open problem is try to extend the downward collapse theorem to handle *adaptive* NP queries. That is, show that if E is computable by small adaptive SAT-oracle circuits then E is computable by small nonadaptive SAT-oracle circuits.

Another interesting open problem is to give a *uniform* version of the downward collapse theorem, or more precisely, to prove that $\mathrm{EXP} \subseteq \mathrm{P}_{||}^{\mathrm{NP}} \Rightarrow \mathrm{EXP} = \mathrm{AM}$. We remark that the argument of this paper can be slightly modified to give $\mathrm{EXP} \subseteq \mathrm{P}_{||}^{\mathrm{NP}} \Rightarrow \mathrm{EXP} \subseteq \mathrm{AM}/\log$. In a subsequent work Fortnow and Klivans [FK05] also consider starting from the uniform assumption: $\mathrm{EXP} \subseteq \mathrm{P}_{||}^{\mathrm{NP}}$. They build on the main result of this paper and are able to get a stronger conclusion, namely: $\mathrm{EXP} \subseteq \mathrm{NP}/\log$.

Acknowledgements

We thank Lance Fortnow, Oded Goldreich, Russell Impagliazzo, Rahul Santhanam, Amnon Ta-Shma and Salil Vadhan for helpful comments. We also thank the anonymous referees for numerous helpful comments and suggestions.

References

[AK01] V. Arvind and J. Kobler. On pseudorandomness and resource-bounded measure. *Theoretical Computer Science*, 255, 2001.

[AKRR03] E. Allender, M. Koucky, D. Ronneburger, and S. Roy. Derandomization and distinguishing complexity. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, pages 209–220, 2003.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, 1985.

[BCG+96] N. H. Bshouty, R. Cleve, R. Gavalda, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

[BDCGL90] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 379–386, 1990.

[BF90] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *LNCS*, pages 37–48. Springer, 1990.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BGP00] M. Bellare, O. Goldreich, and E. Petrank. Uniform generation of NP-witnesses using an NP-oracle. *Information and Computation (formerly Information and Control)*, 163, 2000.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, November 1984.

[BR94] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In Shafi Goldwasser, editor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[Cai01] J.-Y. Cai. $\mathrm{S}_2^{\mathrm{P}} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 620–629, 2001.

[Can96] R. Canetti. On BPP and the polynomial-time hierarchy. *Information Processing Letters*, 57:237–241, 1996.

[FF93] J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[FK05] L. Fortnow and A. Klivans. NP with small advice. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 228–234, 2005.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[Gol97] O. Goldreich. A sample of samplers – A computational perspective on sampling (survey). In *Electronic Colloquium on Computational Complexity, technical report TR 97-020*, 1997.

[Gol98] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Springer-Verlag, Algorithms and Combinatorics, 1998.

[HHT97] Y. Han, L.A. Hemaspaandra, and T. Thierauf. Threshold computation and cryptographic security. *SIAM J. Comput.*, 26(1):59–78, 1997.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *40th Annual Symposium on Foundations of Computer Science*, pages 181–190, 1999.

[ISW00] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-random generators with optimal seed length. In *Proceedings of the thirty second annual Symposium on Theory of Computing*, pages 1–10, 2000.

[IW97] R. Impagliazzo and A. Wigderson. $\mathrm{P} = \mathrm{BPP}$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[JVV86] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43(2-3):169–188, 1986.

[Kab02] V. Kabanets. Derandomization: A brief overview. In *Electronic Colloquium on Computational Complexity, technical report, TR 02-008*, 2002.

[KvM02] A. R. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, October 2002.

[MV99] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *40th Annual Symposium on Foundations of Computer Science*, pages 71–80, 1999.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[RS98] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7(2):152–162, 1998.

[Sto83] L. Stockmeyer. The complexity of approximate counting. In *Proceedings of the fifteenth annual Symposium on Theory of Computing*, pages 118–126, 1983.

[STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62:236–266, 2001.

[SU01] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd Symposium on Foundations of Computer Science*, pages 648–657, 2001.

[Uma03] C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67:419–440, 2003.

[WB86] L.R. Welch and E.R. Berlekamp. Error correction for algebraic block codes. U.S. Patent No. 4,633,470, issued December 30, 1986.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23th Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982.