Thesis for the degree
Doctor of Philosophy

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

By
Roei Tell

עבודת גמר (תזה) לתואר
דוקטור לפילוסופיה

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

מאת
רועי תל

דרנדומיזציה, דרנדומיזציה כמותית, ויחסי הגומלין שלהן עם חסמים תחתונים

Derandomization, Quantified Derandomization, and Their Interplay with Lower Bounds

Advisor:
Prof. Oded Goldreich

מנחה:
פרופ. עודד גולדרייך

April 2020

ניסן ה'תש"פ

Abstract

What is the role of *randomness* in computation? This thesis focuses on the $prBPP = prP$ conjecture, which asserts that randomness is not crucial for efficiently solving decision problems, as well as for solving a broad class of search problems.

The contributions in this thesis are two-fold. First, we make unconditional progress in the long-term effort towards proving the $prBPP = prP$ conjecture. We consider several restricted settings that are prominent frontiers in the study of the conjecture, and within these restricted settings we study a relaxed version of the conjecture, called quantified derandomization. We then show two complementary results: We construct algorithms that solve this relaxed version within these restricted settings, and we show that even very mild improvement to the parameters achieved by these algorithms would suffice to prove the original ("non-relaxed") version of the conjecture in these restricted settings. Moreover, we point at an inherent limitation of certain "black-box" techniques in this context, which preclude algorithms that rely only on these techniques from closing the remaining gap in the parameters.

Secondly, we significantly strengthen the connections between the $prBPP = prP$ conjecture and the question of lower bounds for algorithms and for non-uniform circuits. Specifically, we show that any proof that $prBPP = prP$ implies circuit lower bounds that are significantly stronger compared to what was previously known; and (in the other direction), we show that strong lower bounds for uniform probabilistic algorithms imply (almost-)polynomial-time average-case derandomization of BPP. Lastly, we prove that certain lower bounds for a weak uniform model of computation are both sufficient and necessary in order to prove that the $prBPP = prP$ conjecture is *completely equivalent* to specific circuit lower bounds.

Acknowledgements

> "I wonder what it feels like to work in a real police department". Detective Jimmy McNulty, *The Wire S05E01*; written by David Simon.[1]

There is an inherent distance between the ideal notion of an institution and any real-world materialization of that institution. I thank my advisor, Oded Goldreich, for creating an experience of academia for me during my PhD studies that seems remarkably close to the ideal notion of academia. From day one of my studies Oded made it clear that he views me as an independent scientist taking my first steps, and that his role as my advisor is to support me, to enable me to conduct my research, and to educate me as a member of the scientific community. Oded strongly encouraged me to follow my intellectual interests, and to focus on knowledge and understanding out of an inherent interest in the relevant question itself. And he gave me full academic freedom to do so: The resources available to me somehow appeared boundless, and he never put any constraint on my research interests (not even mild pressure), and emboldened me to ignore external pressures as much as possible.

I am extremely grateful to Oded for his thorough mentorship and guidance as well as for his emotional support throughout my PhD. The focus of Oded's guidance was not on how to solve any specific research problem, or on how to conduct research in general (I'm actually still looking for an answer to the latter). Instead, his focus was on the "big" questions: What is the role of science and of academy, what is the purpose of our academic discipline, what is my role within it, and all this with a strong emphasis on ethics and on commitment to a purpose. And alongside these "big" questions, Oded shaped my worldview of complexity theory in countless conversations, and provided specific detailed comments on the contents of the works included in this thesis. All the while and in tandem, the focus of Oded's emotional support was on supplying continuous encouragement, advice and feedback, alongside validating what he calls the "inherent emotional difficulty" in conducting research.

My education in theoretical computer science was carried out within the amazingly rich intellectual environment of the Weizmann Institute, and it saddens me to leave such a vibrant and inspiring scientific community. The TCS community in Weizmann instills an ethos of intellectual curiousity, high standards, and personal kindness, and these values make it a wonderful community to be part of. I had the privilege of interacting with (and learning from) scientists whom I look up to in Weizmann, and

[1] Profanity excluded from the original quote.

I am very grateful to Itai Benjamini, to Zvika Brakerski, to Irit Dinur, to Moni Naor, to Guy Rothblum, and to Amnon Ta-Shma (who is from the neighboring Tel Aviv University) for sharing their thoughts in courses and discussions, as well as for their support and guidance. I am particularly indebted to Irit and Guy for supporting my research financially from their grants during periods of my PhD. I also thank Irit and Zvika for serving as my PhD committee and guiding me with feedback and advice.

During my PhD I visited Rocco Servedio at Columbia University in the spring of '17, Rahul Santhanam at Oxford in the spring of '18, and Ryan Williams at MIT in the summer of '18, for prolonged academic visits. I am very grateful to Rocco, to Rahul and to Ryan for hosting me so kindly and warmly and for opening new intellectual horizons for me; these visits were an invaluable part of my education. And I greatly enjoyed collaborating with many creative and brilliant researchers, who taught me first and foremost how fun collaboration can be: I am grateful to my co-authors Lijie Chen, Dean Doron, Igor Oliveira, Ron Rothblum, Rahul Santhanam, Amnon Ta-Shma, and Eylon Yogev for sharing their insights and for the opportunity to work with them.

Lastly, I thank colleagues, friends and senior students in Weizmann (at the time) – Gil Cohen, Karthik C.S., Tom Gur, Inbal Livni, and Avishay Tal – for their friendship and support, and for sharing parts of our PhD experiences together.

Contents

Chapter 1

Introduction

> "The marriage of randomness and computation has been one of the most fertile ideas in computer science". *Mathematics and Computation*, Avi Wigderson, 2017.

This thesis focuses on the *promise-*$\mathcal{BPP}$ $=$ *promise-*$\mathcal{P}$ conjecture, which asserts that randomness has *limited usefulness* for solving a broad class of computational problems.

As noted in the quote above, randomness plays a key role in many settings in computer science. For example, three settings in which randomness is *crucial* are sublinear-time algorithms, cryptography, and learning theory. In contrast, many interesting computational problems can be efficiently solved by deterministic algorithms that do not use randomness at all. This naturally raises the question of classification: Which computational problems can only be efficiently solved using randomness? And more generally, what role does randomness play in computation?

The *promise-*$\mathcal{BPP}$ $=$ *promise-*$\mathcal{P}$ conjecture, or $pr\mathcal{BPP} = pr\mathcal{P}$ in short, asserts that randomness is not crucial for *efficiently solving decision problems*; specifically, it asserts that every decision problem that can be solved in polynomial time using randomness can also solved in polynomial time without using randomness.[1] Note that this conjecture does not assert that randomness is completely useless in this context, but only that randomness is not *crucial* in this context, and can be removed at a relatively-mild cost (i.e., with at most a polynomial overhead in the running time). For example, the conjecture allows for the possibility that certain problems can be solved polynomially faster using randomness, and also that randomness can yield simpler algorithms.

This $pr\mathcal{BPP} = pr\mathcal{P}$ conjecture has a fundamental place in complexity theory. First, as is typical in many other settings in theoretical computer science, our focus on decision problems is a methodological simplification, and also captures a more general class of problems: For example, if $pr\mathcal{BPP} = pr\mathcal{P}$, then a corresponding class of $pr\mathcal{BPP}$ *search problems* can also be solved in deterministic polynomial time (see [Gol11]). And secondly, this conjecture is closely-related to many other funda-

[1] An appropriate reference for the $pr\mathcal{BPP} = pr\mathcal{P}$ conjecture seems to be [Gil77], which introduced the relevant concepts and raised a similar conjecture (see [Gil77, End of Sec 3]).

mental questions in complexity theory, and in particular to questions that refer to lower bounds for algorithms and circuits (for more details see Chapter 4).

The $prBPP = prP$ conjecture is often interpreted as an *algorithmic* problem, namely the problem of constructing efficient deterministic algorithms that simulate (or replace) efficient randomized algorithms. Historically, there have been several famous examples of decision problems for which initially only probabilistic algorithms were known, but that were subsequently discovered to also be solvable by deterministic algorithms (the most famous example is that of deciding whether a given integer is prime; see [AKS04]). However, there are still decision problems that we currently only know how to solve efficiently using randomness. The most prominent of these problems is a "meta-algorithmic" problem called the Circuit Acceptance Probability Problem (CAPP), which is in fact *complete* for $prBPP$ (i.e., CAPP can be solved in deterministic polynomial time if and only if $prBPP = prP$; see Proposition 2.4.2).

The main contributions in this thesis are two-fold. First, we make unconditional progress in the long-term effort towards proving the $prBPP = prP$ conjecture. We study a problem called quantified derandomization, which was introduced by Goldreich and Wigderson [GW14] and constitutes a relaxed version of the CAPP problem mentioned above (see Section 3.1 for details about this relaxed problem). Focusing on several restricted settings that are well-known frontiers in the study of CAPP, we show two complementary types of results: On the one hand we construct relatively-efficient algorithms that solve the quantified derandomization problem within these restricted settings; and on the other hand we show that even very mild improvement to the parameters achieved by the foregoing algorithms would suffice to solve CAPP (i.e., solve the original and "non-relaxed" problem) in these restricted settings. In some of these settings, the gap between the parameters of the algorithm that we construct and parameters that would suffice to solve CAPP is remarkably small (see, e.g., Sections 3.3 and 3.5). We also point at an inherent limitation of certain "black-box" techniques in the context of quantified derandomization, which preclude algorithms that rely only on these techniques from closing the remaining gap in the parameters.

Secondly, we significantly strengthen the connections between the $prBPP = prP$ conjecture and other fundamental questions in theoretical computer science, and in particular the question of lower bounds for algorithms and for non-uniform circuits. First, we show that any proof that $prBPP = prP$ implies *circuit lower bounds* that are significantly stronger compared to what was previously known (see Section 4.2). Secondly, in the other direction, we show that near-exponential lower bounds for *uniform* probabilistic algorithms against certain functions computable in polynomial space imply *(almost-)polynomial-time* average-case derandomization of BPP (see Section 4.3). Lastly, we prove that certain lower bounds for a weak uniform model of computation imply that the $prBPP = prP$ conjecture is *completely equivalent* to specific circuit lower bounds (see Section 4.4).

1.1 Works on which the thesis is based

In Chapter 3 we include contributions that focus on quantified derandomization. Sections 3.2, 3.3, and 3.4 are based on the work [Tel19a]. Section 3.5 is based on the work [Tel18b] and on a joint work with Chen [CT19]. Section 3.6 is based on a joint work with Doron and Ta-Shma [DTST19]. And Section 3.7 is based on the work [Tel17b].

In Chapter 4 we include contributions that focus on connections between the $prBPP = prP$ conjecture and lower bounds. Section 4.2 is based on the work [Tel19b]. Sections 4.3 and 4.4 are based on a joint work with Chen, Rothblum, and Yogev [CRT+19].

In Appendix A we provide high-level descriptions of several other results that were obtained during our doctoral studies, but that were not mentioned above. Specifically, Section A.1 is based on a joint work with Oliveira and Santhanam [OST19], Section A.2 is based on the work [Tel18a], and Section A.3 is based on the work [Tel20].

Finally, in Appendix B we include five surveys and expositions of known results that were initially published in our personal website.

Chapter 2

Preliminaries

We denote by $\{\mathfrak{D} \to \mathfrak{R}\}$ the set of functions from domain $\mathfrak{D}$ to range $\mathfrak{R}$. We will be interested in Boolean functions, represented either as functions $f\colon \{0,1\}^n \to \{0,1\}$ or as functions $f\colon \{-1,1\}^n \to \{-1,1\}$. We say that a function $f\colon \{-1,1\}^n \to \{-1,1\}$ accepts an input $x \in \{-1,1\}^n$ if $f(x) = -1$. For two Boolean functions f and g over a domain $\mathfrak{D}$, we say that f and g are δ-close if $\Pr_{x\in\mathfrak{D}}[f(x) = g(x)] \geq 1 - \delta$.

For any set $L \subseteq \{0,1\}^*$ and $n \in \mathbb{N}$, we denote by $L_n = L \cap \{0,1\}^n$ the restriction of L to n-bit inputs. Similarly, for $f : \{0,1\}^* \to \{0,1\}^*$, we denote by $f_n : \{0,1\}^n \to \{0,1\}^*$ the restriction of f to the domain of n-bit inputs.

Distributions and random variables will be denoted by boldface letters. Given a set Σ, which will typically be clear from the context, we denote by $\mathbf{u}_k$ the uniform distribution over Σ^k. Given a distribution $\mathbf{d}$, we write $x \sim \mathbf{d}$ to denote a value x that is sampled according to $\mathbf{d}$; when we write $x \in \Sigma^k$ in probabilistic expressions, we mean the uniform distribution over Σ^k.

For a vector $w = (w_1, ..., w_n) \in \mathbb{R}^n$, we denote by $\|w\|_2$ the standard ℓ_2-norm $\|w\|_2 = \sqrt{\sum_{i\in[n]} w_i^2}$. For $h < n$, we denote $w_{>h} = (w_{h+1}, ..., w_n) \in \mathbb{R}^{n-h}$ and $w_{\geq h} = (w_h, ..., w_n) \in \mathbb{R}^{n-h+1}$. For two vectors $w, x \in \mathbb{R}^n$, we denote $\langle w, x\rangle = \sum_{i\in[n]} w_i \cdot x_i$.

2.1 Standard mathematical facts

2.1.1 Distributions with limited independence

We say that random variables $\mathbf{x}_1, ..., \mathbf{x}_n \in \{0,1\}^n$ are t-wise independent if for every set $S \subseteq [n]$ of size $|S| = t$, the marginal distribution $(\mathbf{x}_i)_{i\in S}$ is uniform over $\{0,1\}^t$. We will use the following well-known tail bound (for a proof see [BR94, Lemma 2.3]):

Fact 2.1.1 (tail bound for t-wise independent distributions)**.** *Let $n \in \mathbb{N}$, and let $t \geq 4$ be an even number. Let $\mathbf{x}_1, ..., \mathbf{x}_n$ be random variables in $\{0,1\}$ that are t-wise independent, and denote $\mu = \mathbb{E}\left[\frac{1}{n} \cdot \sum_{i\in[n]} \mathbf{x}_i\right]$. Then, for any $\zeta > 0$ it holds that $\Pr\left[\left|\frac{1}{n} \cdot \sum_{i\in[n]} \mathbf{x}_i - \mu\right| \geq \zeta\right] \leq 8 \cdot \left(\frac{t\cdot\mu\cdot n + t^2}{\zeta^2 \cdot n^2}\right)^{t/2}$.*

We say that $\mathbf{x}_1, ..., \mathbf{x}_n \in \{0,1\}^n$ are δ-almost t-wise independent if for every set $S \subseteq [n]$ of size $|S| = t$, the statistical distance between $(\mathbf{x}_i)_{i \in S}$ and the uniform distribution over $\{0,1\}^t$ is at most δ. Then, the following well-known tail bound holds:

Fact 2.1.2 (tail bound for almost t-wise independent distributions; see, e.g., [LRT+09, Lem 18])**.** *Let $n \in \mathbb{N}$, let $t \geq 4$ be an even number, and let $\delta > 0$. Let $\mathbf{x}_1, ..., \mathbf{x}_n$ be random variables in $\{0,1\}$ that are δ-almost t-wise independent, and denote $\mu = \mathbb{E}\left[\frac{1}{n} \cdot \sum_{i \in [n]} \mathbf{x}_i\right]$. Then, for any $\zeta > 0$ it holds that $\Pr\left[\left|\frac{1}{n} \cdot \sum_{i \in [n]} \mathbf{x}_i - \mu\right| \geq \zeta\right] < 8 \cdot \left(\frac{t \cdot \mu \cdot n + t^2}{\zeta^2 \cdot n^2}\right)^{t/2} + (2 \cdot n)^t \cdot \delta$.*

In particular, for $t = \Theta(1)$ and $\zeta = \mu/2$ and $\delta = 1/p(n)$, where p is a sufficiently large polynomial, we have that

$$\Pr\left[\frac{1}{n} \cdot \sum_{i \in [n]} \mathbf{x}_i \in \mu \pm (\mu/2)\right] = O\left((\mu \cdot n)^{-t/2}\right) .$$

We will also need the following fact, which, loosely speaking, asserts that concatenating two independently-chosen distributions that are almost t-wise independent yields a distribution that is still almost t-wise independent.

Fact 2.1.3 (concatenating almost t-wise independent distributions)**.** *Let $n, n' \in \mathbb{N}$, let $\delta, \delta' < \frac{1}{2}$, and let $t \in \mathbb{N}$. Let $\mathbf{y}$ be a distribution over $\{0,1\}^n$ that is δ-almost t-wise independent, and let $\mathbf{z}$ be a distribution over $\{0,1\}^{n'}$ that is δ'-almost t-wise independent. Let $\mathbf{r} = \mathbf{y} \circ \mathbf{z}$ be a distribution that is obtained by concatenating a sample from $\mathbf{y}$ and an independent sample from $\mathbf{z}$. Then, the distribution $\mathbf{r}$ is $(\delta + \delta')$-almost t-wise independent.*

Proof. Fix a set $S \subseteq [n + n']$ of size $|S| = t$, and let us prove that the ℓ_1-distance between $\mathbf{r}_S$ and the uniform distribution is at most $2 \cdot (\delta + \delta')$ (which implies that the statistical distance between them is at most $\delta + \delta'$). Partition S into $W = S \cap [n]$ and $W' = S \setminus [n]$, and denote $w = |W|$ and $w' = |W'|$. Then, we have that:

$$\begin{aligned} \|\mathbf{r}_S - \mathbf{u}_t\|_1 &= \|\mathbf{y}_W \circ \mathbf{z}_{W'} - \mathbf{u}_w \circ \mathbf{u}_{w'}\|_1 \\ &\leq \|\mathbf{y}_W \circ \mathbf{z}_{W'} - \mathbf{y}_W \circ \mathbf{u}_{w'}\|_1 + \|\mathbf{y}_W \circ \mathbf{u}_{w'} - \mathbf{u}_w \circ \mathbf{u}_{w'}\|_1 \\ &= \|\mathbf{z}_{W'} - \mathbf{u}_{w'}\|_1 + \|\mathbf{y}_W - \mathbf{u}_w\|_1 , \end{aligned}$$

which is upper-bounded by $2 \cdot \delta' + 2 \cdot \delta$. ∎

2.1.2 Concentration and anti-concentration inequalities

We will rely on two standard facts that assert concentration and anti-concentration bounds for certain distributions. Specifically, we will need a standard version of Hoeffding's inequality, and a corollary of the Berry-Esséen theorem:

Theorem 2.1.4 (Hoeffding's inequality; for a proof see, e.g., [DP09, Sec. 1.7])**.** *Let $w \in \mathbb{R}^n$, and let $\mathbf{z}$ be a uniformly-chosen random vector in $\{-1,1\}^n$. Then, for any $t > 0$ it holds that*

$$\Pr\left[|\langle w, \mathbf{z} \rangle| \geq t \cdot \|w\|_2\right] \leq \exp(-\Omega(t^2)) .$$

Theorem 2.1.5 (a corollary of the Berry-Esséen theorem; see, e.g., [DGJ+10, Thm 2.1, Cor 2.2]). *Let $w \in \mathbb{R}^n$ and $\mu > 0$ such that for every $i \in [n]$ it holds that $|w_i| \leq \mu \cdot \|w\|_2$, and let $\mathbf{z}$ be a uniformly-chosen random vector in $\{-1,1\}^n$. Then, for any $\theta \in \mathbb{R}$ and $t > 0$ it holds that:*

$$\Pr\left[\langle w, \mathbf{z}\rangle \in \theta \pm t \cdot \|w\|_2\right] \leq 2 \cdot (t + \mu) \,.$$

2.2 Restrictions of functions

Given a function $f : \{0,1\}^n \to \{0,1\}$, a restriction of f is a subset $W \subseteq \{0,1\}^n$. We say that a function f simplifies under a restriction W to a function from a class $\mathcal{H} \subseteq \{\{0,1\}^n \to \{0,1\}\}$ if there exists $h \in \mathcal{H}$ such that for every $w \in W$ it holds that $h(w) = f(w)$.

We will be interested in restrictions that are subcubes, and such restrictions can be described by a string $\rho \in \{0,1,\star\}^n$, where the subcube consists of all $x \in \{0,1\}^n$ such that for every $i \in [n]$ for which $\rho_i \neq \star$ it holds that $x_i = \rho_i$. The living variables under ρ are the input bits indexed by the set $\{i \in [n] : \rho_i = \star\}$. We will sometimes describe a restriction by a pair $\rho = (I, z)$, where $I = \{i \in [n] : \rho_i = \star\}$ is the set of living variables and $z = (\rho_i)_{i \in ([n] \setminus I)} \in \{0,1\}^{[n]\setminus I}$ is the sequence of values that ρ assigns to the variables that are fixed. The function f restricted to a subcube ρ is denoted by $f|_\rho : \{0,1\}^n \to \{0,1\}$ such that $f|_\rho(x) = f(y)$, where for every $i \in [n]$ it holds that $y_i = x_i$ if $\rho_i = \star$ and $y_i = \rho_i$ otherwise. We will also consider the composition of restrictions to subcubes, where a composition $\rho = \rho_1 \circ \rho_2$ yields the restricted function $f|_\rho = \left(f|_{\rho_2}\right)|_{\rho_1}$.

We identify strings $r \in \{-1,1\}^{(q+1)\cdot n}$, where $n, q \in \mathbb{N}$, with restrictions $\rho = \rho_r \in \{-1,1,\star\}^n$, as follows: Each variable is assigned a block of $q+1$ bits in the string; the variable remains alive if the first q bits in the block are all 1, and otherwise takes the value of the $(q+1)^{th}$ bit. When we refer to a "block" in the string that corresponds to a restriction, we mean a block of $q+1$ bits that corresponds to some variable. When we say that a restriction is chosen from a distribution $\mathbf{r}$ over $\{-1,1\}^{(q+1)\cdot n}$, we mean that a string is chosen according to $\mathbf{r}$, and interpreted as a restriction. In addition, we will sometimes identify a *pair* of strings $y \in \{-1,1\}^{q\cdot n}$ and $z \in \{-1,1\}^n$ with a restriction $\rho = \rho_{y,z}$. In this case, the restriction $\rho = \rho_{y,z}$ is the restriction ρ_r that is obtained by combining y and z to a string r in the natural way (i.e., appending a bit from z to each block of q bits in y). Note that the string y determines which variables ρ keeps alive, and the string z determinse the values that ρ assigns to the fixed variables.

2.3 Models of computation

Throughout the current text, fix any standard model of uniform computation, for example a multitape Turing Machine or a RAM Turing Machine. In specific places

where our results will be sensitive the the relevant model of computation, we will point this out explicitly.

We now recall standard definitions of other models of computation that will be used in the current thesis: Specifically, non-uniform Boolean circuits (see Section 2.3.1), Merlin-Arthur protocols (see Section 2.3.2) and multivariate polynomials over finite fields (see Section 2.3.3).

2.3.1 Non-uniform circuits

We will consider Boolean circuit families $\{C_n\}_{n\in\mathbb{N}}$ such that C_n gets n input bits and outputs a single bit. Whenever we refer to circuits (without qualifying which type), we mean non-uniform circuit families over the De-Morgan basis (i.e., the gates of the circuit can compute the $\wedge, \vee$, and $\neg$ functions) with fan-in at most two and unlimited fan-out, and without any specific structural restrictions (e.g., without any limitation on their depth). The size of a circuit is the number of its gates, unless specifically indicated otherwise (in Section 3.5 we will measure the size of certain circuits by the number of their wires). Moreover, we consider some fixed standard form of representation for such circuits, where the representation size is quasilinear in the size of the circuit.

2.3.1.1 Constant-depth circuits

The circuit class $\mathcal{AC}^0$ consists of all circuit families over the De-Morgan basis such that the circuit gates have unbounded fan-in and fan-out, and for every $n \in \mathbb{N}$, the size of C_n is at most $\text{poly}(n)$, and the depth of C_n (i.e., the length of the longest path from an input gate to the output gate) is upper bounded by a constant. We also assume that for every $n \in \mathbb{N}$ it holds that C_n has $2 \cdot n$ input gates that correspond to the input literals (i.e., the input bits $x_1, ..., x_n$ and their negations $\neg x_1, ..., \neg x_n$); and that C_n is *layered*, in the sense that in a fixed circuit, for every integer d, all gates at distance d from the input gates are of the same gate-type (i.e., either $\wedge$ or $\vee$).

2.3.1.2 Constant-depth circuits with modular gates

The circuit class $\mathcal{AC}^0[\oplus]$ is defined similarly to $\mathcal{AC}^0$, the only difference being that the basis is extended: The gates can compute the $\wedge, \vee, \neg$, and $\oplus$ functions (rather than only $\wedge, \vee$, and $\neg$). We stress that a $\oplus$-gate can compute either the parity of its input gates, or the negated parity of its input gates. We also assume that all $\mathcal{AC}^0[\oplus]$ circuits are *layered*, in the sense that in a fixed circuit, for every integer d, all gates at distance d from the input gates are of the same gate-type (i.e., either $\wedge$, or $\vee$, or $\oplus$).

For every integer $m \in \mathbb{N}$, the class $\mathcal{AC}^0[m]$ is defined analogously, by extending the basis to $\wedge, \vee, \neg$ and MOD_m, where $\text{MOD}_m \colon \{0,1\}^n \to \{0,1\}$ outputs zero if the sum of its inputs is zero modulo m, and outputs one otherwise. We denote $\mathcal{ACC}^0 = \cup_{m\in\mathbb{N}} \mathcal{AC}^0[m]$.

2.3.1.3 Linear threshold circuits

A linear threshold function (or LTF, in short) $\Phi : \{-1,1\}^n \to \{-1,1\}$ is a function of the form $\Phi(x) = \text{sgn}(\langle x, w\rangle - \theta)$, where $w \in \mathbb{R}^n$ is a vector of real "weights", and $\theta \in \mathbb{R}$ is a real number (the "threshold"), and $\langle x, w\rangle = \sum_{i\in[n]} x_i \cdot w_i$ denotes the standard inner-product over the reals (as already defined above).[1] Indeed, the majority function is the special case where the weights are identical (e.g., $w_i = 1$ for all $i \in [n]$) and the threshold is zero (i.e., $\theta = 0$).

The following are standard definitions (see, e.g., [Ser07; DGJ+10]), which refer to "structural" properties of LTFS and will be useful for us in Section 3.5.

Definition 2.3.1 (regularity)**.** *For $\epsilon > 0$, we say that a vector $w \in \mathbb{R}^n$ is ϵ-*regular *if for every $i \in [n]$ it holds that $|w_i| \leq \epsilon \cdot \|w\|_2$. An LTF $\Phi = (w, \theta)$ is ϵ-regular if w is ϵ-regular.*

Definition 2.3.2 (critical index)**.** *When $w \in \mathbb{R}^n$ satisfies $|w_1| \geq |w_2| \geq ... \geq |w_n|$, the ϵ-*critical index *of w is defined as the smallest $h \in [n]$ such that $w_{>h}$ is ϵ-regular (and $h = \infty$ if no such $h \in [n]$ exists). The critical index of an LTF $\Phi = (w, \theta)$ is the critical index of w', where $w' \in \mathbb{R}^n$ is the vector that is obtained from w by permuting the coordinates in order to have $|w'_1| \geq ... \geq |w'_n|$.*

Definition 2.3.3 (balanced LTF)**.** *For $t \in \mathbb{R}$, we say that an LTF $\Phi = (w, \theta)$ is t-balanced if $|\theta| \leq t \cdot \|w\|_2$; otherwise, we say that Φ is t-imbalanced.*

We will be interested in linear threshold circuits, which are circuits that consist only of LTF gates with unbounded fan-in and fan-out. We assume that linear threshold circuits are *layered*, in the sense that for each gate Φ, all the gates feeding into Φ have the same distance from the inputs. For $n, d, m \in \mathbb{N}$, let $\mathcal{C}_{n,d,m}$ be the class of linear threshold circuits over n input bits of depth $d \geq 1$ and with at most m wires. For some fixed sizes and depths, linear threshold circuits are known to be stronger than circuits with majority gates; however, linear threshold circuits can be simulated by circuits with majority gates with a polynomial size overhead and with one additional layer (see [GHR92; GK98]). Thus, the class $\mathcal{TC}^0$ as a whole equals the class of linear threshold circuits.

Representation of linear threshold circuits One of the algorithms that we present (specifically, the algorithm in Theorem 3.5.1) gets as input an explicit representation of a linear threshold circuit C, where the weights and thresholds of the LTFs in C may be arbitrary real numbers. We will not be specific about how exactly C is represented as an input to the algorithm, since the algorithm works in any reasonable model. In particular, the algorithm only performs *addition, subtraction, and comparison operations* on the weights and thresholds of the LTFs in C.

[1] When dealing with LTFs we can assume, without loss of generality, that $\langle w, x\rangle \neq \theta$ for every $x \in \{-1,1\}^n$ (because for every Boolean function over $\{-1,1\}^n$ that is computable by an LTF there exists an LTF that computes the function such that $\langle w, x\rangle \neq \theta$ for every $x \in \{-1,1\}^n$).

Explicitly considering one convenient model, one may assume that the weights and threshold of each LTF are integers of unbounded magnitude (since the real numbers can be truncated at some finite precision without changing the function). In this case, the circuit C has a binary representation, and the required time to perform addition, subtraction, and comparison on these integers is linear in the representation size.[2]

2.3.2 Merlin-Arthur protocols

We recall the standard definition of Merlin-Arthur protocols (i.e., $\mathcal{MA}$ verifiers) that receive non-uniform advice.

Definition 2.3.4 ($\mathcal{MA}$ verifiers with non-uniform advice)**.** *For $t, \ell : \mathbb{N} \to \mathbb{N}$, a set $S \subseteq \{0,1\}^*$ is in $\mathcal{MATIME}[t]/\ell$ if there exists a probabilistic machine V, called a* verifier*, such that the following holds: The verifier V gets input $x \in \{0,1\}^*$, and a witness $w \in \{0,1\}^*$, and an advice string $a \in \{0,1\}^*$, and runs in time $t(|x|)$; and there exists a sequence $\{a_n\}_{n\in\mathbb{N}}$ of advice such that $|a_n| = \ell(n)$ and:*

1. *For every $x \in S$ there exists $w \in \{0,1\}^{t(|x|)}$ such that $\Pr[V(x, w, a_{|x|}) = 1] \geq 2/3$.*
2. *For every $x \notin S$ and every $w \in \{0,1\}^{t(|x|)}$ it holds that $\Pr[V(x, w, a_{|x|}) = 1] \leq 1/3$.*

If in Item (1) the probability that $V(x, w, a_{|x|}) = 1$ is one (i.e., the verifier has perfect completeness when given "good" advice), then we denote $S \in \mathcal{MATIME}_0[t]/\ell$.

It is common to denote by $\mathcal{MATIME}[t]$ the class $\mathcal{MATIME}[t]/0$ (i.e., when the verifier receives no non-uniform advice). Note that $\mathcal{MA} = \bigcup_{c\in\mathbb{N}} \mathcal{MATIME}[n^c]$.

Any Merlin-Arthur verifier can be modified to have perfect completeness, at the cost of a polynomial overhead in the running time, using the ideas of [Lau83] (see [FGM+89]). Moreover, this statement also holds in the setting when the verifier relies on non-uniform advice. We include a proof of this fact for completeness:

Theorem 2.3.5 ($\mathcal{MA}$ verifiers have perfect completeness, wlog)**.** *There exists a universal constant $c > 1$ such that the following holds. Let $t, \ell : \mathbb{N} \to \mathbb{N}$ such that t is time-constructible, and let $S \subseteq \{0,1\}^*$ such that $S \in \mathcal{MATIME}[t]/\ell$. Then, $S \in \mathcal{MATIME}_0[t^c]/\ell$.*

Proof. Let V be the $\mathcal{MATIME}[t]/\ell$ verifier (with two-sided error) for S, and let $\{a_n\}_{n\in\mathbb{N}}$ be a sequence of "good" advice strings for V. By standard error-reduction, we can assume that V runs in time $t' = \text{poly}(t)$ and (when given the "good" advice) has error probability at most $1/3t'$. Denote by $V(x, w, r, a')$ the decision of V on input x with proof w and randomness r and advice a'.

[2] It is well-known that every LTF over n input bits has a representation with integer weights of magnitude $2^{\tilde{O}(n)}$ (for proof see, e.g., [Hås94]), and therefore the circuit C actually has a representation of size $\text{poly}(n)$. However, we do not know of a polynomial-time algorithm to find such a representation for a given circuit C.

We construct a verifier V' with perfect completeness for S. On input $x \in \{0,1\}^n$ and with advice $a' \in \{0,1\}^{\ell(n)}$, the verifier V' expects to receive as proof both a string $w \in \{0,1\}^{t(n)}$ and $t' = t'(n)$ strings $\bar{s} = s_1, ..., s_{t'} \in \{0,1\}^{t'}$. The verifier V' uniformly chosses $r \in \{0,1\}^{t'}$, and accepts $x \in \{0,1\}^n$ if and only if there exists $i \in [t']$ such that $V(x, w, r \oplus s_i, a') = 1$. Now, when a' equals the "good" advice a_n, the following holds:

- If $x \in S$, then there exists $(w, \bar{s})$ such that for every $r \in \{0,1\}^{t'}$ there exists $i \in [t']$ satisfying $V(x, w, r \oplus s_i, a_n) = 1$; this is because for any w such that $\Pr_r[V(x, w, r, a_n) = 1] \geq 1 - 1/3t'$ we have that

$$\begin{aligned}\Pr_{\bar{s}}[\exists r \in \{0,1\}^{t'} \text{ st } \forall i \in [t'], V(x, w, r \oplus s_i, a_n) = 0] \\ &\leq \sum_{r \in \{0,1\}^{t'}} \Pr_{\bar{s}}[\forall i \in [t'], V(x, w, r \oplus s_i, a_n) = 0] \\ &\leq 2^{t'} \cdot (1/3t')^{t'} ,\end{aligned}$$

  which is less than one.

- If $x \notin S$ then for every $(w, \bar{s})$ that the prover sends it holds that $\Pr_r[\exists i \in [t'] \text{ st } V(x, w, r \oplus s_i, a_n) = 1] \leq \sum_{i \in [t']} \Pr_r[V(x, w, r \oplus s_i, a_n) = 1] \leq 1/3$. ■

2.3.3 Polynomials over finite fields

A multivariate polynomial $p \colon \mathbb{F}^n \to \mathbb{F}$ of degree d over a finite field $\mathbb{F}$ can be viewed as a codeword in the corresponding Reed-Muller code; thus, if p is non-zero, then the relative distance of the corresponding Reed-Muller code, which is stated below, lower bounds the fraction of inputs on which p does not vanish.

Theorem 2.3.6 (distance of the Reed-Muller code; see, e.g., [GRS19, Lem 9.4.1])**.** *For any $d, q \in \mathbb{N}$, let $a = \lfloor d/(q-1) \rfloor$ and $b = d \pmod{q-1}$. The relative distance of the Reed-Muller code of degree d over alphabet q is $\delta_{RM}(d, q) = q^{-a} \cdot (1 - b/q) \geq q^{-d/(q-1)}$.*

The $\mathtt{OR} \colon \mathbb{F}^k \to \mathbb{F}$ function maps any non-zero input $z \in \mathbb{F}^k \setminus \{0^k\}$ to $1 \in \mathbb{F}$, and maps 0^k to zero. We consider a generalization of this function, which we call *multivalued OR*; a multivalued OR function maps any non-zero $z \in \mathbb{F}^k \setminus \{0^k\}$ to *some* non-zero element (i.e., different non-zero inputs may yield different outputs), while still mapping 0^k to zero. That is:

Definition 2.3.7 (multivalued OR functions)**.** *For any finite field $\mathbb{F}$, we say that a polynomial $\mathtt{mvOR} \colon \mathbb{F}^k \to \mathbb{F}$ is a* multivalued OR function *if $\mathtt{mvOR}(0^k) = 0$, but $\mathtt{mvOR}(x) \neq 0$ for every $x \neq 0^k$.*

For a fixed field $\mathbb{F}$ there are many different k-variate multivalued OR functions. Indeed, the standard $\mathtt{OR}$ function is a multivalued OR function, but it has maximal degree $k \cdot (q-1)$ as a polynomial. We will need k-variate multivalued OR functions that are of much lower degree (i.e., degree approximately k); such functions can be constructed relying on well-known techniques in algebraic geometry (see, e.g., [CLO15, Exercise 8] for a reference to the well-known underlying techniques):

Proposition 2.3.8 (low-degree multivalued OR function)**.** *Let $\mathbb{F}$ be a finite field and let $k \in \mathbb{N}$. Then, there exists a multivalued OR function $\texttt{mvOR}\colon \mathbb{F}^k \to \mathbb{F}$ that is computable by a polynomial of degree less than $2k$.*

Proof. For $\mathbb{F} = \mathbb{F}_2$ we can use the polynomial $f(x_1, ..., x_k) = \Pi_{i\in[k]}(1 + x_i) + 1$, which has degree exactly k. For all other fields, let us first assume that k is a power of two. Our construction is based on a known construction of a bivariate quadratic polynomial $f^{(2)} : \mathbb{F}^2 \to \mathbb{F}$ that vanishes only at $(0, 0)$ (see below). Given such a polynomial, for every $k \geq 4$ that is a power of two, we recurse the construction; that is, we define $f^{(k)}(x_1, ..., x_k) = f^{(2)}\left(f^{(k/2)}(x_1, ..., x_{k/2}), f^{(k/2)}(x_{k/2+1}, ..., x_k)\right)$. Observe that $f^{(k)}(x_1, ..., x_k) = 0$ if and only if $x_i = 0$ for every $i \in [k]$, whereas $\deg(f^{(k)}) = k$. Finally, for any k that is not a power of two, we use a padding argument to obtain a polynomial of degree $2^{\lceil \log(k) \rceil} < 2 \cdot k$.

The following construction of the bivariate polynomial $f^{(2)}$ is taken from [BSGH+06, Footnote 28]. Fix any irreducible monic quadratic polynomial $p : \mathbb{F} \to \mathbb{F}$, denoted $p(z) = z^2 + \alpha \cdot z + \beta$; such a polynomial exists since there are $|\mathbb{F}|^2$ monic quadratic polynomials, but only $|\mathbb{F}| + \binom{|\mathbb{F}|}{2} < |\mathbb{F}|^2$ of them have a root.[3] Note that $\beta \neq 0$, since p is irreducible. Then, we let $f^{(2)}(x, y) = x^2 + \alpha \cdot x \cdot y + \beta \cdot y^2$. Note that $f^{(2)}(0, 0) = 0$, and that for any $\sigma \in \mathbb{F} \setminus \{0\}$ it holds that $f^{(2)}(\sigma, 0) = \sigma^2 \neq 0$ and $f^{(2)}(0, \sigma) = \beta \cdot \sigma^2 \neq 0$. Finally, observe that for any $\sigma_1, \sigma_2 \in \mathbb{F}$ such that $\sigma_1 \cdot \sigma_2 \neq 0$ we have that $f^{(2)}(\sigma_1, \sigma_2) = \sigma_2^2 \cdot p(\sigma_1/\sigma_2)$; thus, $f^{(2)}(\sigma_1, \sigma_2) = 0$ if and only if $p(\sigma_1/\sigma_2) = 0$, which cannot happen since p is irreducible. ■

A minor improvement to the proof of Proposition 2.3.8. The proof of Proposition 2.3.8 transforms any irreducible monic quadratic polynomial $p : \mathbb{F} \to \mathbb{F}$ into a corresponding quadratic polynomial $f^{(2)} : \mathbb{F}^2 \to \mathbb{F}$ that vanishes only at $(0, 0)$. In the proof, the construction of p was non-explicit, and relied on a counting argument. We now show two different constructions of p that are more explicit: One construction for fields of odd characteristic, and another construction for fields of characteristic two.

Our construction for the case of odd characteristic follows [BSGH+06, Footnote 28]. Specifically, let $\alpha \in \mathbb{F}$ be a quadratic non-residue (i.e., for every $\sigma \in \mathbb{F}$ it holds that $\sigma^2 \neq \alpha$); there exists such α since the characteristic is odd.[4] Then, the irreducible polynomial is $p(z) = z^2 - \alpha$.

Now, for the case of characteristic two, denote the multiplicative group of $\mathbb{F}$ by $\mathbb{F}^*$. Our construction will use an element $\alpha \in \mathbb{F}$ that is defined in the following claim:

Claim 2.3.9. *Let $\mathbb{F}$ be a field of characteristic two. Then, there exists $\alpha \in \mathbb{F}^*$ such that for every $z \in \mathbb{F}^*$ it holds that $z + z^{-1} \neq \alpha$.*

[3] Because every quadratic polynomial with a root is either of the form $p(z) = (z + \alpha)^2$ or of the form $p(z) = (z + \alpha) \cdot (z + \beta)$, for $\alpha \neq \beta \in \mathbb{F}$.

[4] Recall that over fields of characteristic two the mapping $x \mapsto x^2$ is a bijection: This is since $x^2 = y^2$ if and only if $x = \pm y$, but over fields of characteristic two this means that $x = y$.

Proof. Consider the mapping $\Phi : \mathbb{F}^* \to \mathbb{F}$ such that $\Phi(z) = z + z^{-1}$. Note that $\Phi(1) = 0$, and that for every $z \in \mathbb{F}^* \setminus \{1\}$ it holds that $z + z^{-1} \neq 0$. [5] Hence, Φ maps $1 \mapsto 0$ and $\mathbb{F}^* \setminus \{1\}$ to $\mathbb{F}^*$, which means that there exists $\alpha \in \mathbb{F}^*$ such that $\Phi(z) \neq \alpha$ for all $z \in \mathbb{F}^*$. □

Now, for $\alpha \in \mathbb{F}^*$ as in Claim 2.3.9, the irreducible polynomial is $p(z) = z^2 + \alpha \cdot z + 1$. Indeed, observe that $p(0) \neq 0$, and for $z \neq 0$ it holds that $p(z) = 0$ if and only if $z + z^{-1} = \alpha$, which cannot happen due to our choice of α.

2.4 Derandomization, pseudorandom generators and hitting-set generators

2.4.1 Circuit acceptance probability problem

We now formally define the circuit acceptance probability problem (or CAPP, in short); this well-known problem is also sometimes called Circuit Derandomization, Approx Circuit Average, and GAP-SAT or GAP-UNSAT.

Definition 2.4.1 (CAPP)**.** *The* circuit acceptance probability problem *with parameters $\alpha, \beta \in [0,1]$ such that $\alpha > \beta$ and for size $S : \mathbb{N} \to \mathbb{N}$ (or (α,β)-CAPP$[S]$, in short) is the following promise problem:*

- *The* YES *instances are (representations of) circuits over v input bits of size at most $S(v)$ that accept at least an α fraction of their inputs.*
- *The* NO *instances are (representations of) circuits over v input bits of size at most $S(v)$ that accept at most a β fraction of their inputs.*

We define the CAPP$[S]$ *problem (i.e., omitting α and β) as the* $(2/3, 1/3)$-CAPP *problem. We define* CAPP *to be the problem when there is no restriction on S.*

We typically consider the complexity of CAPP as a function of the input size, which corresponds to the size of the (description of the) circuit. In this context, solving CAPP$[S]$ becomes *easier* as the function S grows *larger*: This is because for a given input size $n = \tilde{O}(S(v))$, if S is large then it means that the number of variables is *small* compared to n. Recall that a naive deterministic algorithm can solve the problem in time $2^v \cdot \text{poly}(S)$, whereas the naive probabilistic algorithm solves the problem in time $v \cdot \text{poly}(m) \leq \text{poly}(S)$.

It is well-known that CAPP is complete for $pr\mathcal{BPP}$ under deterministic polynomial-time reductions; in particular, CAPP can be solved in deterministic polynomial time *if and only if* $pr\mathcal{BPP} = pr\mathcal{P}$.

[5] This is the case because over a field of characteristic two, $z + z^{-1} = 0$ for a non-zero z implies that $z^2 = 1$, but the latter happens only for $z = 1$.

Proposition 2.4.2 (CAPP is complete for $prBPP$; see, e.g., [Vad12, Cor 2.31] or [Gol08, Exer 6.14]). *Any problem in $prBPP$ can be reduced to CAPP in deterministic polynomial time. Consequently, relying on the fact that CAPP $\in prBPP$, we have that CAPP $\in prP$* if and only if *$prBPP = prP$.*

Lastly, some of our results will refer to *non-deterministic* algorithms that solve $(1,1/3)$-CAPP. To avoid confusion we now define precisely what we mean by the latter notion:

Definition 2.4.3 (non-deterministically solving CAPP). *We say that* $(1,1/3)$-CAPP can be solved in non-deterministic time $T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *if there exists a non-deterministic machine that, when given as input a circuit C of size m over v variables, runs in time $T(m,v)$ and satisfies the following: If C has acceptance probability one, then for some non-deterministic choice the machine accepts; and if C has acceptance probability at most $1/3$, then the machine always rejects (regardless of the non-deterministic choices).*

2.4.2 Pseudorandom generators and hitting-set generators

We next recall the standard definitions of Hitting-Set Generators (HSGs) and of Pseudorandom Generators (PRGs). We will present two sets of definitions: One instantiated for Boolean functions (see Section 2.4.2.1) and one instantiated for multivariate polynomials over finite fields (see Section 2.4.2.2).

2.4.2.1 PRGs and HSGs for Boolean functions

We will use the following two standard definitions of pseudorandom generators and of hitting-set generators for Boolean functions.

Definition 2.4.4 (pseudorandom distribution). *For $\epsilon > 0$ and a domain $\mathfrak{D}$, we say that a distribution $\mathbf{w}$ over $\mathfrak{D}$ is ϵ-pseudorandom for a class of functions $\mathcal{F} \subseteq \{\mathfrak{D} \to \{0,1\}\}$ if for every $f \in \mathcal{F}$ it holds that $\Pr_{w \sim \mathbf{w}}[f(w) = 1] \in \Pr_{w \in \mathfrak{D}}[f(w) = 1] \pm \epsilon$. In this case, we say that functions in $\mathcal{F}$ are ϵ-*fooled *by $\mathbf{w}$.*

Definition 2.4.5 (pseudorandom generators). *Let $\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{F}_n$, where for every $n \in \mathbb{N}$ it holds that $\mathcal{F}_n \subseteq \{\{0,1\}^n \to \{0,1\}\}$, and let $\epsilon : \mathbb{N} \to [0,1]$ and $\ell : \mathbb{N} \to \mathbb{N}$. An algorithm G is a* pseudorandom generator for $\mathcal{F}$ with error parameter ϵ and seed length ℓ *if for every $n \in \mathbb{N}$ it holds that $G(1^n, \mathbf{u}_{\ell(n)})$ is a distribution over $\{0,1\}^n$ that is ϵ-pseudorandom for $\mathcal{F}_n$.*

Definition 2.4.6 (hitting-set generators). *Let $\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{F}_n$, where for every $n \in \mathbb{N}$ it holds that $\mathcal{F}_n$ is a set of functions $\{0,1\}^n \to \{0,1\}$, and let $\ell : \mathbb{N} \to \mathbb{N}$. An algorithm G is a* hitting-set generator for $\mathcal{F}$ with seed length ℓ *if for every $n \in \mathbb{N}$, when G is given as input 1^n and a random seed of length $\ell(n)$, it outputs a string in $\{0,1\}^n$ such that for every $f \in \mathcal{F}_n$ it holds that $\Pr_{y \in \{0,1\}^{\ell(n)}}[f(G(1^n, y)) \neq 0] > 0$. For $\epsilon : \mathbb{N} \to (0,1]$, we say that G has* density *ϵ if for every $n \in \mathbb{N}$ and $f \in \mathcal{F}_n$ it holds that $\Pr_{y \in \{0,1\}^{\ell(n)}}[f(G(1^n, y)) \neq 0] \geq \epsilon(n)$.*

We will need the following well-known construction of a pseudorandom generator from a function that is "hard" for non-uniform circuits, by Umans [Uma03] (following the line of works initiated by Nisan [Nis91] and Nisan and Wigderson [NW94]).

Theorem 2.4.7 (Umans' PRG; see [Uma03, Thm 6])**.** *There exists a constant $c > 1$ and an algorithm G such that the following holds. When G is given an n-bit truth-table of a function $f : \{0,1\}^{\log(n)} \to \{0,1\}$ that cannot be computed by circuits of size s, and a random seed of length $\ell(n) = c \cdot \log(n)$, it runs in time n^c, and for $m = s^{1/c}$ outputs an m-bit string that is $(1/m)$-pseudorandom for every size-m circuit over m bits.*

Corollary 2.4.8 (near-optimal non-uniform hardness-to-randomness using Umans' PRG)**.** *There exists a universal constant $\Delta > 1$ such that for every time-computable $S : \mathbb{N} \to \mathbb{N}$ and for $T(n) = 2^{\Delta \cdot S^{-1}(n^\Delta)}$, we have that*

1. *If $\mathcal{E} \not\subset \mathcal{SIZE}[S]$ then $\mathtt{CAPP} \in \mathtt{i.o.}pr\mathcal{DTIME}[T]$.*
2. *If $\mathcal{E} \not\subset \mathtt{i.o.}\mathcal{SIZE}[S]$ then $\mathtt{CAPP} \in pr\mathcal{DTIME}[T]$.*

2.4.2.2 PRGs and HSGs for polynomials

We recall the standard definitions of hitting-set generators and of pseudorandom generators for multivariate polynomials over finite field. Recall that HSGs for a class of polynomials need to produce a set of inputs such that any polynomial from the class evaluates to *non-zero* on some input in the set. That is:

Definition 2.4.9 (hitting-set generator)**.** *Fix a field $\mathbb{F}$, and let $d, n \in \mathbb{N}$. A function $H\colon \{0,1\}^\ell \to \mathbb{F}^n$ is a* hitting-set generator for a set of polynomials *$\mathcal{P} \subseteq \{\mathbb{F}^n \to \mathbb{F}\}$ if for every non-zero function $p \in \mathcal{P}$ there exists $s \in \{0,1\}^\ell$ satisfying $p(H(s)) \neq 0$. In this case, the set $S = \{H(s) : s \in \{0,1\}^\ell\}$ is called a* hitting-set *for $\mathcal{P}$.*

Definition 2.4.10 (explicit hitting-set generators)**.** *Let $\ell, q, d\colon \mathbb{N} \to \mathbb{N}$, let $\{\mathbb{F}_{q(n)}\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ it holds that $\mathbb{F}_{q(n)}$ is a field of size $q(n)$, and let $H = \{H_n\colon \{0,1\}^{\ell(n)} \to \mathbb{F}_{q(n)}^n\}$ such that for every $n \in \mathbb{N}$ it holds that H_n is a hitting-set generator for polynomials of degree $d(n)$. We say that H is* polynomial-time computable *if there exists an algorithm that gets as input $s \in \{0,1\}^\ell$ and outputs $H_n(s)$ in time $\mathrm{poly}(\ell, \log(q), n)$.*

In Definition 2.4.9, the generator G gets a seed from $\{0,1\}^\ell$, rather than from $\mathbb{F}^\ell$ (as is also common in some texts); indeed, the seed length $\ell(n)$ of the generator G might depend on the size of $\mathbb{F}$. This choice was made because it is more general, and because we want to measure the seed length in bits.

The standard definition of PRGs for polynomials in $p\colon \mathbb{F}^n \to \mathbb{F}$ that we will use is as follows. Consider the distribution over $\mathbb{F}$ that is obtained by uniformly choosing $x \in \mathbb{F}^n$ and outputting $p(x)$, and the distribution over $\mathbb{F}$ that is obtained by choosing a seed s for a PRG G and outputting $p(G(s))$. We require that the statistical distance between the two distributions is small. That is:

Definition 2.4.11 (pseudorandom generator). *Fix a field $\mathbb{F}$, let $d, n \in \mathbb{N}$, and let $\rho > 0$. A function $G\colon \{0,1\}^{\ell} \to \mathbb{F}^n$ is a* pseudorandom generator with error ρ for polynomials of degree d *if for every polynomial $p\colon \mathbb{F}^n \to \mathbb{F}$ of degree at most d it holds that*

$$\sum_{\sigma \in \mathbb{F}} \left| \Pr_{s \in \{0,1\}^{\ell}}[p(G(s)) = \sigma] - \Pr_{x \in \mathbb{F}^n}[p(x) = \sigma] \right| \leq \rho \,.$$

An alternative standard definition of PRGs for polynomials requires that the "character distance" $\left|\mathbb{E}_{x \in \mathbb{F}^n}[\mathsf{e}^{p(x)}] - \mathbb{E}_{x}[\mathsf{e}^{p(G(s))}]\right|$ will be small, where e is any (fixed, non-trivial) character of $\mathbb{F}$. The "character distance" and the statistical distance are equivalent, up to a multiplicative factor of $\sqrt{q-1}$ (see [Lov09, Lem 2.4]).

Lastly, we recall the standard lower bound on the size of hitting-sets for polynomials of degree d (for completeness, we include its proof) and state the complementary upper-bound that is obtained by a standard probabilistic argument.

Fact 2.4.12 (lower bound on the size of hitting-sets for linear subspaces). *Let $\mathbb{F}$ be a finite field, let $n \in \mathbb{N}$, and let $\mathcal{C} \subseteq \{\mathbb{F}^n \to \mathbb{F}\}$ be a linear subspace of dimension $D = \dim(\mathcal{C})$. Then, any hitting-set for $\mathcal{C}$ has at least D points. In particular, for any $d < n$, any hitting-set for degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$ has size at least $\binom{n+d}{d}$, and correspondingly the seed length of any hitting-set generator for such polynomials is at least $d \cdot \log(n/d)$.*

Proof. Assume towards a contradiction that $S \subseteq \mathbb{F}^n$ is a hitting-set for $\mathcal{C}$ with $D-1$ points. Consider a generator matrix M for the linear subspace $\mathcal{C}$, which is a full-rank $D \times |\mathbb{F}|^n$ matrix over $\mathbb{F}$ whose D rows span $\mathcal{C}$. Let M_S be the projection of M to the $D-1$ columns corresponding to the points in S.

Since M_S is a $D \times (D-1)$ matrix, there exists a non-trivial linear combination of the rows of M_S that yields the all-zero row. Now, since S is a hitting-set for $\mathcal{C}$, the only function in $\mathcal{C}$ that vanishes on all of S is the all-zero function; in particular, any non-trivial linear combination of the rows of M_S that yields the all-zero row (which induces a corresponding function in $\mathcal{C}$ that vanishes on S) also yields the all-zero row in M. Thus, we obtain a non-trivial linear combination of the rows of M that yields the all-zero row, contradicting the hypothesis that M is full-rank.

The "in particular" part follows since the dimension of the corresponding Reed-Muller code (which is a linear subspace of $\mathbb{F}^n$) is $D = \binom{n+d}{d} > \binom{n}{d} > (n/d)^d$, where we used the hypothesis that $d < n$. ∎

Fact 2.4.13 (upper bound on the size of hitting-sets). *Let $\mathbb{F}$ be a finite field, let $n \in \mathbb{N}$, and let $d < n$. Then, there exists a (non-explicit) hitting-set generator for polynomials $\mathbb{F}^n \to \mathbb{F}$ of degree d with seed length $O(d \cdot \log(n/d) + \log\log(q))$.*

Proof. The number of degree-d polynomials is at most $q^{\binom{n+d}{d}}$, and each of them vanishes on at most $1-\delta$ of its inputs, where $\delta \geq q^{-d/(q-1)}$ is the distance of the corresponding Reed-Muller code. Thus, if we randomly choose

$$O\left((1/\delta) \cdot \binom{n+d}{d} \cdot \log(q)\right) < O\left(q^{d/(q-1)} \cdot \binom{2n}{d} \cdot \log(q)\right)$$

elements in $\mathbb{F}^n$, with high probability we obtain a hitting-set for degree-d polynomials. The number of bits that we need to sample an element from this hitting-set is

$$O\left(\frac{d}{q-1}\cdot\log(q)+d\cdot\log(n/d)+\log\log(q)\right) < O\left(d\cdot\log(n/d)+\log\log(q)\right) . \quad \blacksquare$$

2.4.2.3 PRGs and HSGs for uniform circuits, and average-case derandomization

We now define the notions of "average-case" derandomization of probabilistic algorithms. The first definitions that we need are of circuits that distinguish a distribution from uniform, and of distributions that are pseudorandom for uniform algorithms, as follows:

Definition 2.4.14 (distinguishing distributions from uniform)**.** *For two functions* $\mathtt{str}, \ell : \mathbb{N} \to \mathbb{N}$*, let* G *be an algorithm that gets input* 1^n *and a random seed of length* $\ell(n)$ *and outputs a string of length* $\mathtt{str}(n)$*. Then:*

1. *For* $n \in \mathbb{N}$ *and* $n' \in \mathtt{str}^{-1}(n)$*, we say that* $D_n : \{0,1\}^n \to \{0,1\}$ ϵ-distinguishes $G(1^{n'}, \mathbf{u}_{\ell(n')})$ from uniform *if* $\left|\Pr[D_n(G(1^{n'}, \mathbf{u}_{\ell(n')})) = 1] - \Pr[D_n(\mathbf{u}_n) = 1]\right| > \epsilon$.
2. *For a probabilistic algorithm* A*, an integer* n*, and* $\epsilon > 0$*, we say that* $G(1^n, \mathbf{u}_{\ell(n)})$ is ϵ-pseudorandom for A *if the probability that* $A(1^{\mathtt{str}(n)})$ *outputs a circuit that* ϵ*-distinguishes* $G(1^n, \mathbf{u}_{\ell(n)})$ *from uniform is at most* ϵ*.*

When applying this definition without specifying a function $\mathtt{str}$*, we assume that* $\mathtt{str}$ *is the identity function.*

We now use Definition 2.4.14 to define pseudorandom generators for uniform circuits and hitting-set generators for uniform circuits, which are analogous to the standard definitions of PRGs and HSGs for non-uniform circuits:

Definition 2.4.15 (PRGs for uniform circuits)**.** *For* $\ell : \mathbb{N} \to \mathbb{N}$*, let* G *be an algorithm that gets as input* 1^n *and a random seed of length* $\ell(n)$*, and outputs strings of length* n*. For* $t, a : \mathbb{N} \to \mathbb{N}$ *and* $\epsilon : \mathbb{N} \to (0,1)$*, we say that* G *is an* ϵ-i.o.-PRG for (t,a)-uniform circuits *if for every probabilistic algorithm* A *that runs in time* $t(n)$ *and gets* $a(n)$ *bits of non-uniform advice there exists an infinite set* $S_A \subseteq \mathbb{N}$ *such that for every* $n \in S_A$ *it holds that* $G(1^n, \mathbf{u}_{\ell(n)})$ *is* $\epsilon(n)$*-pseudorandom for* A*. If for every such algorithm* A *there is a set* S_A *as above that contains all but finitely-many inputs, we say that* G *is an* ϵ-PRG for (t,a)-uniform circuits*.*

Definition 2.4.16 (HSGs for uniform circuits)**.** *For* $\ell : \mathbb{N} \to \mathbb{N}$*, let* H *be an algorithm that gets as input* 1^n *and a random seed of length* $\ell(n)$*, and outputs strings of length* n*. For* $t, a : \mathbb{N} \to \mathbb{N}$ *and* $\epsilon : \mathbb{N} \to (0,1)$*, we say that* H *is an* ϵ-HSG for (t,a)-uniform circuits *if the following holds. For every probabilistic algorithm* A *that gets input* 1^n *and* $a(n)$ *bits of non-uniform advice, runs in time* $t(n)$*, and outputs a circuit* $D_n : \{0,1\}^n \to \{0,1\}$*, and every sufficiently large* $n \in \mathbb{N}$*, with probability at least* $1 - \epsilon(n)$ *(over the coin tosses of* A*) at least one of the following two cases holds:*

1. *There exists* $s \in \{0,1\}^{\ell(n)}$ *such that* $D_n(G(1^n, s)) = 1$.

2. *The circuit* D_n *satisfies* $\Pr_{x \in \{0,1\}^n}[D_n(x) = 1] \leq \epsilon(n)$.

It is well-known that PRGs for uniform circuits can be used to derandomize $\mathcal{BPP}$ "on average" (see, e.g., [Gol11, Prop. 4.4]). Analogously, *HSGs* for uniform circuits can be used to derandomize $\mathcal{RP}$ "on average". That is, loosely speaking, if there exists an HSG for uniform circuits, then for any $L \in \mathcal{RP}$ there exists a deterministic algorithm D such that for every efficiently-samplable distribution $\mathcal{X}$, the probability over $x \sim \mathcal{X}$ that $D(x) \neq L(x)$ is small. For simplicity, we prove the foregoing claim for HSGs that are computable in polynomial time and have logarithmic seed length:

Claim 2.4.17 (HSGs for uniform circuits $\Rightarrow$ derandomization of $\mathcal{RP}$ "on average")**.** *For* $\epsilon : \mathbb{N} \to (0,1)$ *such that* $\epsilon(n) \leq 1/3$*, assume that for every* $k \in \mathbb{N}$ *there exists a* ϵ*-HSG for* $(n^k, 0)$*-uniform circuits that is polynomial-time computable and that has logarithmic seed length. Then, for every* $L \in \mathcal{RP}$ *and every* $c \in \mathbb{N}$*, there exists a deterministic polynomial-time algorithm* D *such that for every probabilistic algorithm* F *that runs in time* n^c *and every sufficiently large* $n \in \mathbb{N}$*, the probability (over the internal coin tosses of* F*) that* $F(1^n)$ *outputs a string* $x \in \{0,1\}^n$ *such that* $D(x) \neq L(x)$ *is at most* $\epsilon(n)$*.*

Proof. Let M be an $\mathcal{RP}$ machine that decides L in time $n^{c'}$, for some $c' \in \mathbb{N}$. The deterministic algorithm D gets input $x \in \{0,1\}^n$, enumerates the seeds of the HSG for output length $m = n^{c'}$ and with the parameter $k = O(1 + c/c')$, and accepts x if and only if there exists an output r of the HSG such that M accepts x with random coins r. Note that D never accepts inputs $x \notin L$ (since M is an $\mathcal{RP}$ machine), and thus we only have to prove that for every algorithm F as in the claim's statement, the probability that $x = F(1^n)$ satisfies both $x \in L$ and $D(x) = 0$ is at most $\epsilon(n)$.

To do so, let F be a probabilistic algorithm that runs in time n^c. Consider the probabilistic algorithm A that, on input 1^m, runs the algorithm F on input 1^n to obtain $x \in \{0,1\}^n$, and outputs a circuit $C_{m,x} : \{0,1\}^m \to \{0,1\}$ that computes the decision of M at input x as a function of M's $m = n^{c'}$ random coins. Note that the algorithm A runs in time at most $m^{O(1+c/c')}$, and also note that the only probabilistic choices that A makes are a choice of $x = F(1^n)$. Thus, by Definition 2.4.16 for every sufficiently large m, with probability at least $1 - \epsilon(m) > 1 - \epsilon(n)$ over choice of $x = F(1^n)$ (i.e., over the coin tosses of A), if $D(x) = 0$ then $\Pr_r[C_{m,x}(r) = 1] = \Pr[M(x) = 1] \leq \epsilon(n) \leq 1/3$, which means that $x \notin L$. ■

2.4.2.4 PRGs for linear threshold functions

Pseudorandom generators for the class of linear threshold functions have a nice alternative characterization; this characterization was communicated to us by Rocco Servedio, and is attributed to Li-Yang Tan. Towards presenting this characterization, let us define the notion of a distribution that is ϵ-pseudorandomly concentrated.

Definition 2.4.18 (ϵ-pseudorandomly concentrated distribution)**.** *For $n \in \mathbb{N}$ and $\epsilon > 0$, we say that a distribution $\mathbf{z}$ over $\{-1,1\}^n$ is ϵ-*pseudorandomly concentrated *if the following holds: For every $w \in \mathbb{R}^n$ and every $a < b \in \mathbb{R}$ it holds that $\Pr[\langle w, \mathbf{z}\rangle \in [a,b]] \in \Pr[\langle w, \mathbf{u}_n\rangle \in [a,b]] \pm \epsilon$.*

We now show that PRGs for LTFs are essentially equivalent to algorithms that output pseudorandomly concentrated distributions.

Claim 2.4.19 (being pseudorandomly concentrated is equivalent to being pseudorandom for LTFs)**.** *Let $\mathbf{z}$ be a distribution over $\{-1,1\}^n$. Then,*

1. *If $\mathbf{z}$ is ϵ-pseudorandom for LTFs, then $\mathbf{z}$ is (2ϵ)-pseudorandomly concentrated.*
2. *If $\mathbf{z}$ is ϵ-pseudorandomly concentrated, then $\mathbf{z}$ is ϵ-pseudorandom for LTFs.*

Proof. Let us first prove Item (1). Fix $w \in \mathbb{R}^n$ and $I = [a,b] \subseteq \mathbb{R}$. For any fixed $z \in \{-1,1\}^n$, exactly one of three events happens: Either $\langle w, z\rangle \in I$, or $\langle w, z\rangle < a$, or $\langle w, z\rangle > b$. Since the event $\langle w, z\rangle < a$ can be tested by an LTF (i.e., by the LTF $\Phi(z) = \operatorname{sgn}(a - \langle w, z\rangle)$), this event happens with probability $\Pr_{z\in\{-1,1\}^n}[\langle w, z\rangle < a] \pm \epsilon$ under a choice of $z \sim \mathbf{z}$. Similarly, the event $\langle w, z\rangle > b$ happens with probability $\Pr_{z\in\{-1,1\}^n}[\langle w, z\rangle > b] \pm \epsilon$ under a choice of $z \sim \mathbf{z}$. Thus, the probability under a choice of $z \sim \mathbf{z}$ that $\langle w, z\rangle \in I$ is $\Pr_{z\in\{-1,1\}^n}[\langle w, z\rangle \in I] \pm 2\epsilon$.

To see that Item (2) holds, let $\Phi = (w, \theta)$ be an LTF over n input bits, and let $M = \|w\|_1 = \sum_{i\in[n]} |w_i|$. Then, for every $z \in \{-1,1\}^n$ it holds that $\Phi(z) = -1$ if and only if $z \in [-M, \theta]$. Thus, $\Pr[\Phi(\mathbf{z}) = -1] = \Pr[\mathbf{z} \in [-M,\theta]] \in \Pr[\mathbf{u}_n \in [-M,\theta]] \pm \epsilon = \Pr[\Phi(\mathbf{u}_n) = -1] \pm \epsilon$. ∎

We will rely on the following construction of a pseudorandom generator for LTFs, by Gopalan, Kane, and Meka [GKM15]:

Theorem 2.4.20 (a PRG for LTFs; [GKM15, Cor. 1.2])**.** *For every $\epsilon > 0$, there exists a polynomial-time computable pseudorandom generator for the class of LTFs with seed length $O\left(\log(n/\epsilon) \cdot (\log\log(n/\epsilon))^2\right)$.*

2.5 Dispersers, extractors, and averaging samplers

We now recall the standard definition of dispersers and of extractors $[N] \times \{0,1\}^\ell \to [M]$. We will typically think of N and M as vector spaces $\mathbb{F}^n$ and $\mathbb{F}^m$, and in particular as Boolean hypercubes $\{0,1\}^n$ and $\{0,1\}^m$.

Definition 2.5.1 (disperser)**.** *A function $\mathsf{Disp}\colon [N] \times \{0,1\}^\ell \to [M]$ is a (k,δ)-disperser if for every $T \subseteq [M]$ of size $|T| \geq \delta \cdot |M|$, the probability over $x \in [N]$ that for all $s \in \{0,1\}^\ell$ it holds that $\mathsf{Disp}(x,s) \notin T$ is less than $2^k/|N|$. The value ℓ is the* seed length *of the disperser.*

Dispersers can be defined in an equivalent way using the notion of *min-entropy*; let us recall this notion:

Definition 2.5.2 (min-entropy). *We say that a random variable* $\mathbf{x}$ *has* min-entropy k *if for every* $x \in \mathrm{supp}(\mathbf{x})$ *is holds that* $\Pr[\mathbf{x} = x] \leq 2^{-k}$.

Then, Definition 2.5.1 is equivalent to the following definition: Disp is a (k, δ)-disperser if for any random variable $\mathbf{x}$ over $[N]$ with min-entropy k, the support of $\mathsf{Disp}(\mathbf{x}, \mathbf{u}_\ell)$ covers at least $(1-\delta) \cdot |M|$ elements from $[M]$.

We will be particularly interested in dispersers $\mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ that can be computed by low-degree multivariate polynomials over $\mathbb{F}$. Specifically, we require that for each fixed seed $s \in \{0,1\}^\ell$ and output index $i \in [m]$, the function that maps any $z \in \mathbb{F}^n$ to the i^{th} output of Disp at z with seed s (i.e., $z \mapsto \mathsf{Disp}(z,s)_i$) has low degree as a polynomial $\mathbb{F}^n \to \mathbb{F}$.

Definition 2.5.3 (degree of a disperser). *We say that a disperser* $\mathsf{Disp} : \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ *has* degree d *if for every fixed* $s \in \{0,1\}^\ell$ *and* $i \in [m]$, *the polynomial* $p_{s,i} \colon \mathbb{F}^n \to \mathbb{F}$ *defined by* $p_{s,i}(z) = \mathsf{Disp}(z,s)_i$ *is of degree at most* d. *If* $d = 1$, *then we say the disperser is* linear.

Recall that there are two standard dispersers that are linear: The naive disperser, which treats its input $z \in \mathbb{F}^n$ as a list of samples from $\mathbb{F}^m$ and its seed as an index of a sample in this list; and the subspace sampler, which treats its input as the description of an affine subspace in $\mathbb{F}^m$ and its seed as an index of an element in the subspace. Nevertheless, these dispersers have disadvantages (small output length and large seed length, respectively), and in our results we will use more sophisticated linear dispersers (see Section 3.6.4 for details).

While in dispersers we only care about *covering* almost all of $[M]$, in extractors we want to do it *uniformly*; that is, we require that $\mathtt{Ext}(\mathbf{x}, \mathbf{u}_\ell)$ to be δ-close in statistical distance to the uniform distribution $\mathbf{u}_m$ over $[M]$. Formally:

Definition 2.5.4 (seeded extractor). *A function* $\mathtt{Ext} \colon [N] \times \{0,1\}^\ell \to [M]$ *is a* (k, δ)-*extractor if for every random variable* $\mathbf{x}$ *over* $[N]$ *with min-entropy* k *it holds that* $\mathtt{Ext}(\mathbf{x}, \mathbf{u}_\ell)$ *is* δ-*close in statistical distance to* $\mathbf{u}_m$. *The value* ℓ *is the* seed length *of the extractor.*

As the support size of a distribution which is δ-close to $\mathbf{u}_m$ is at least $(1-\delta) \cdot |M|$, any (k, δ)-extractor is readily a (k, δ)-disperser. We also recall that standard fact that seeded extractors are essentially equivalent to *averaging samplers*; that is:

Definition 2.5.5 (averaging samplers). *A function* $f : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *is an* averaging sampler with accuracy $\epsilon > 0$ and error $\delta > 0$ *(or* (ϵ, δ)-averaging sampler, *in short) if it satisfies the following. For every* $T \subseteq \{0,1\}^m$, *for all but a* δ-*fraction of the strings* $x \in \{0,1\}^n$ *it holds that* $\Pr_{z \in \{0,1\}^t}[f(x,z) \in T] = |T|/2^m \pm \delta$. *We will also identify* f *with a function* $\mathsf{Samp} \colon \{0,1\}^n \to (\{0,1\}^m)^{2^t}$ *in the natural way (i.e.,* $\mathsf{Samp}(x)_i = f(x,i)$*).*

Proposition 2.5.6 (seeded extractors are equivalent to averaging samplers; see, e.g., [Vad12, Cor 6.24]). *Let* $f : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$. *Then, the following two assertions hold:*

1. *If* f *is a* (k, ϵ)-*extractor, then* f *is an averaging sampler with accuracy* ϵ *and error* $\delta = 2^{k-n}$.

2. *If f is an averaging sampler with accuracy ϵ and error δ, then f is an $(n - \log(\epsilon/\delta), 2\epsilon)$-extractor.*

We will use the following well-known construction by Guruswami, Umans, and Vadhan [GUV09].

Theorem 2.5.7 (the near-optimal extractor of [GUV09], instantiated as a sampler and for specific parameters)**.** *Let $\gamma \geq 1$ and $\beta > \alpha > 0$ be constants. Then, there exists a polynomial-time algorithm that for every m computes an $(m^{-\gamma}, 2^{-(\alpha-\beta)\cdot m})$-averaging sampler* $\mathsf{Samp}\colon \{0,1\}^{m'} \to (\{0,1\}^m)^D$*, where $m' = (1+\beta)\cdot m$ and $D = \mathrm{poly}(m)$.*

Chapter 3

Quantified Derandomization

3.1 Introduction

As mentioned in Section 1, the $prBPP = prP$ conjecture holds if and only if a specific "meta-algorithmic" problem, called the Circuit Acceptance Probability Problem (CAPP), can be solved in deterministic polynomial time. Since CAPP will be our starting-point for the current section, let us properly define this problem:

Definition 3.1.1 (CAPP; for a more detailed definition see Definition 2.4.1)**.** *The* Circuit Acceptance Probability Problem*, denoted* CAPP*, is the following promise problem:*

- *The* YES *instances are (representations of) circuits that accept all but at most* $2^v/3$ *of their inputs, where* v *is the number of input gates to the circuit.*
- *The* NO *instances are (representations of) circuits that reject all but at most* $2^v/3$ *of their inputs, where* v *is the number of input gates to the circuit.*

Note that CAPP can be easily solved using randomness, since an algorithm can sample random inputs for the given circuit and estimate, with high probability and up to a small error, the fraction of inputs that the circuit accepts. On the other hand, any problem in $prBPP$ can be reduced in deterministic polynomial time to CAPP, and thus a deterministic deterministic T-time algorithm for CAPP would imply that $prBPP \subseteq prDTIME[T(\text{poly}(n))]$ (see Proposition 2.4.2). Thus, we say that CAPP is complete for $prBPP$ under polynomial-time reductions; informally, we will say that derandomization of $prBPP$ is equivalent to deterministically solving CAPP.

Let us now focus on the maximal *error probability* that we allow probabilistic algorithms in the definition of $prBPP$, which is $1/3$, and on the corresponding number of exceptional inputs that we allow in the definition of CAPP, which is $2^v/3$. It is well-known that the precise constant value $1/3$ is arbitrary: The reason is that the error of any probabilistic algorithm can be reduced from $1/3$ to any small constant, by standard *error-reduction*. In fact, by running the algorithm polynomially-many times with "fresh" randomness each time, and taking a majority vote of the outcomes, the error of the algorithm can be reduced to be exponentially small in the input length.

Moreover, using *randomness-efficient* methods for error-reduction (e.g., as described in Section 2.5), one can modify a polynomial-time algorithm that uses n random coins and errs with probability at most $1/3$ into a polynomial-time algorithm that solves the same problem using $m = \text{poly}(n)$ random coins and having error probability at most $2^{-m+m^{.01}}$ (and indeed, this is just one possible target setting of the parameters). Thus, we could equivalently define $prBPP$ using a tiny error probability such as $2^{-m+m^{.01}}$ instead of $1/3$ (where m is the number of random coins), and polynomial-time derandomization of $prBPP$ (in any of these equivalent definitions) is equivalent to the existence of a deterministic polynomial-time algorithm that solves a corresponding version of CAPP that refers to v-bit circuits that either accept all but at most $2^{v^{.01}}$ of their inputs or reject all but at most $2^{v^{.01}}$ of their inputs.

Since the error probability of probabilistic algorithms (and in the definition of CAPP) can be made so small without losing generality, one might mistakenly suspect that the precise value of the error probability does not matter at all. This is obviously false: If a probabilistic algorithm has no error at all, and correctly computes a Boolean function regardless of its random choices, then derandomizing it is trivial.[1] Therefore, we deduce that for some small values of the error (e.g., the value zero) it is easy to derandomize the corresponding class of probabilistic algorithms, whereas for larger values of the error (e.g., the value $2^{-m+m^{.01}}$), derandomizing the corresponding class of probabilistic algorithms is just as difficult as derandomizing $prBPP$.

In other words, the *precise value* for the error probability *matters a great deal*: It's just that the more interesting values for it – values that do not trivialize the problem, but for which we also do not know if problem is equivalent to the original version – are *very small*. This observation was first put forward by Goldreich and Wigderson [GW14], who introduced the following parametrized version of CAPP, which they named quantified derandomization:

Definition 3.1.2 (quantified derandomization)**.** *For a function $B\colon \mathbb{N} \to \mathbb{N}$, the problem of* quantified derandomization of circuits with B exceptional inputs *is the following promise problem:*

1. *The* YES *instances are (representations of) circuits that accept all but at most $B(v)$ of their inputs, where v is the number of input gates to the circuit.*

2. *The* NO *instances are (representations of) circuits that reject all but at most $B(v)$ of their inputs, where v is the number of input gates to the circuit.*

We say that B quantifies the number of exceptional inputs *to the circuit.*

Observe that solving quantified derandomization of circuits with B exceptional inputs is equivalent to derandomizing probabilistic algorithms that err on at most B

[1] Note that derandomizing algorithms that never err is trivial as long as such algorithms are also forbidden to output "fail" (or "don't know"). In contrast, derandomizing algorithms that never err but output "fail" with small probability (i.e., derandomizing ZPP) does not seem trivial at all.

of their random strings.[2] Indeed, when $B(v) = 2^v/3$ the quantified derandomization problem is identical to CAPP. However, following the discussion above, our interest will typically be in much smaller values of B, such as $B(v) = 2^{o(v)}$ or even $B(v) = 2^{v^{o(1)}}$. Indeed, our hope is that by studying such a potentially-easier version of CAPP we can make progress towards solving the original version of CAPP, and hence also towards proving that $prBPP = prP$.

The current study of CAPP focuses on restricted cases of this problem, most importantly when the input circuit is guaranteed to come from some predetermined "weak" circuit class, such as AC^0. Even for such weak classes, our current knowledge of deterministic algorithms for CAPP is limited, and any progress on constructing such deterministic algorithms is inherently tied to progress in proving new lower bounds for the corresponding circuit class (see Chapter 4 for further details).

Similarly, most of the work on quantified derandomization focuses on cases where the input circuit comes from some "weak" class. Fixing such a "weak" circuit class $\mathcal{C}$, let us consider the problem of quantified derandomization of $\mathcal{C}$-circuits with B exceptional inputs. As a simplisitic mental model, let us think of two values of B in this context: The first value, which we call the feasible value, is the maximal value for which we are able to unconditionally solve the problem by a deterministic polynomial-time algorithm; and the second value, which we call the threshold value, is the minimal value to which we are able to reduce CAPP in polynomial time. This model is simplistic since it fixes or neglects parameters other than B that will be of vital importance in certain settings, such as the running time of our algorithms and potential overheads in the circuit class $\mathcal{C}$.[3] Nevertheless, the model is instructive in outlining a main driving force in this study: Narrowing the gap between the feasible setting of the parameters (i.e., of B, of $\mathcal{C}$, and of the algorithms' running time) and the threshold setting of the parameters, and hopefully eliminating this gap and solving CAPP for $\mathcal{C}$-circuits.

The current chapter focuses on contributions in the study of quantified derandomization. In high-level, we include three main contributions:

- We study quantified derandomization of classes of *constant-depth circuits* that are major frontier in the study of CAPP. In particular, we study quantified derandomization of AC^0, of $AC^0[\oplus]$, of ACC^0, and of TC^0. For each class, we significantly narrow the gap between the feasible setting of the parameters and the threshold setting of parameters; this is done both by constructing new quantified derandomization algorithms, and by constructing new reductions from CAPP.

[2] Specifically, there exists a deterministic T-time algorithm that solves quantified derandomization of circuits with B exceptional inputs if and only if the class of problems solvable by probabilistic algorithms that err on at most B of their random strings is contained in $\cup_c \mathcal{DTIME}[T(n^c)]$. This is analogous to the fact that derandomizing $prBPP$ is equivalent to solving CAPP, and follows the same proof as Proposition 2.4.2.

[3] In more detail, the model fixes the running time of the algorithm for quantified derandomization and of the reduction from CAPP to be polynomial, whereas the question is obviously interesting also when the running time of these algorithms is super-polynomial; and the model also ignores the possibility that the reduction from CAPP to quantified derandomization might incur overheads in the class $\mathcal{C}$ (i.e., might map $\mathcal{C}$-circuits to circuits from a somewhat larger class).

For some of these classes, the remaining gap between the feasible setting and the threshold setting is remarkably small. For example, for the class $\mathcal{AC}^0$ this gap refers to the difference in the value of $B(v)$, and specifically in the precise universal constant $c \in \mathbb{N}$ in the expression $B(v) = 2^{v/(\log v)^{d-c}}$, where d is the depth of the circuit; and for the class of constant-depth linear threshold circuits (i.e., $\mathcal{TC}^0$) the gap refers to a difference in the number of *wires* of the circuit, and specifically in the precise universal constant $c \in \mathbb{N}$ in the bound $n^{1+c^{-d}}$ on the number of wires (where d again is the depth of the circuit).

- We provide a *partial explanation* for the fact that there remains a very small gap between the feasible setting of parameters and the threshold setting of parameters. Specifically, many of the results above are proved using certain "black-box" algorithmic techniques, both for the quantified derandomization algorithm and for the reduction from CAPP. We show that when relying solely on these specific "black-box" techniques for both algorithms, there will *necessarily* be a gap between the feasible setting and the threshold setting.

- We study *hitting-set generators for polynomials* $\mathbb{F}^n \to \mathbb{F}$ *that vanish rarely*. We prove non-explicit results, showing that there *exist* hitting-sets for such polynomials that are of smaller size than the size of any hitting-set for all polynomials of the corresponding degree; and we also construct efficiently-computable hitting-set generators for polynomials that vanish rarely, in settings where no explicit construction of hitting-set generator for general polynomials is known. We complement these results by showing lower bounds on the size of any hitting-set for polynomials that vanish rarely.

Organization and included works. In Section 3.2, which is based on results from [Tel19a], we present several lemmas that will be used throughout the chapter. In Section 3.3, which is based on the work [Tel19a], we present our results for the circuit class $\mathcal{AC}^0$. In Section 3.4, which is also based on [Tel19a], we present our results for the circuit class $\mathcal{AC}^0[\oplus]$. In Section 3.5, which is based on [Tel18b] and on a joint work with Chen [CT19], we present our results for the classes $\mathcal{TC}^0$ and $\mathcal{ACC}^0$. In Section 3.6, which is based on a joint work with Doron and Ta-Shma [DTST19], we present the results for polynomials that vanish rarely. And in Section 3.7, which is based on [Tel17a], we present the limitations of certain "black-box" techniques.

3.2 Randomized tests

In this section we present several lemmas that will be used several times throughout the chapter. We first explain, in high-level, the ideas behind these lemmas (see Section 3.2.1); then we present the lemmas in the context of Boolean functions (see Section 3.2.2) and in the context of polynomials over finite fields (see Section 3.2.3).

3.2.1 Overview

Let $G \subseteq \{0,1\}^n$ be a set of *good* objects, and assume that we want to efficiently and deterministically find some $x \in G$. That is, we want to solve the *search problem* of finding a good object in G. A common technique to do so is to construct a simple deterministic test $T : \{0,1\}^n \to \{0,1\}$ (say, T is a small-depth circuit or a low-degree polynomial) that decides G (i.e., $T(x) = 1$ if and only if $x \in G$). The existence of such a test T is useful, since if T is sufficiently simple such that we are able to construct a hitting-set generator for T, then the generator outputs $x \in G$ with positive probability (because the output distribution of the generator contains $x \in T^{-1}(1) = G$). Indeed, this approach reduces the task of finding $x \in G$ to the task of constructing a test T for G that is sufficiently simple such that we are able to construct a hitting-set generator for T.

Intuitively, the *randomized tests technique* is based on the observation that an argument similar to the one above holds also when we replace the single deterministic test T by a *distribution* $\mathbf{T}$ *over (deterministic) tests* such that, for every fixed $x \in \{0,1\}^n$, it holds that $\mathbf{T}(x)$ computes the indicator function of G, with probability $1 - \epsilon > 1/2$. That is:

Observation 3.2.1. *For a $\epsilon < 1/2$, assume that $\mathbf{T}$ is a distribution as above, and let $\mathbf{w}$ be a distribution over $\{0,1\}^n$ that is a hitting-set with density $1 - \epsilon$ for every $T \in \mathbf{T}$. Then, there exists w in the support of $\mathbf{w}$ such that $w \in G$.*

Proof. Consider the behavior of a random test $T \sim \mathbf{T}$ on a random element from the hitting-set $w \sim \mathbf{w}$. On the one hand, we have that $\Pr[\mathbf{T}(\mathbf{w}) = 1] \geq 1 - \epsilon$ (because for every $T \in \mathbf{T}$ it holds that $\Pr[T(\mathbf{w}) = 1] \geq 1 - \epsilon$). On the other hand, we have that $\Pr[\mathbf{T}(\mathbf{w}) = 0] \geq \Pr[\mathbf{w} \notin G] \cdot \min_{x \notin G}\{\Pr[\mathbf{T}(x) = 0]\} \geq \Pr[\mathbf{w} \notin G] \cdot (1 - \epsilon)$. Combining the two statements, we deduce that $\Pr[\mathbf{w} \notin G] \leq \epsilon/(1 - \epsilon) < 1$. ■

Indeed, this approach reduces the task of finding $x \in G$ to the tasks of constructing a distribution $\mathbf{T}$ over simple tests as above, and of constructing a hitting-set generator with high density for the deterministic tests in the support of $\mathbf{T}$. The main benefit in this approach over the previous one (in which we had a single deterministic test) is that in some cases, *the use of randomness allows us to obtain very simple residual tests, which are simpler than any deterministic test for G*; one appealing example for such a case appears in Section 3.3.2. We stress that when constructing the distribution $\mathbf{T}$ we can be wasteful in the use of randomness, because the existence of $\mathbf{T}$ is only a part of the *analysis*: The actual algorithm for finding $x \in G$ is merely a hitting-set generator, whereas only the proof that the generator outputs $x \in G$ relies on the existence of the distribution $\mathbf{T}$.

Two relaxations of the hypotheses for the argument above can immediately be made, and turn out to be very useful. First, we do not have to assume that $\mathbf{w}$ is a hitting-set with high density for the tests in the support of $\mathbf{T}$, but rather only use the hypothesis that $\Pr[\mathbf{T}(\mathbf{w}) = 1]$ is high. When the set G of good objects is dense (i.e.,

$\Pr_{x\in\{0,1\}^n}[x \in G]$ is high), the latter hypothesis is satisfied when $\mathbf{w}$ is a pseudorandom set for the tests in the support of T (rather than a hitting-set with high density). Secondly, while our argument relies on the hypothesis that $\mathbf{T}(x) = 0$ with high probability for *every* $x \notin G$, it does not rely on the hypothesis that $\mathbf{T}(x) = 1$ with high probability for *every* $x \in G$.

Let us demonstrate one appealing setting in which the two relaxed hypotheses above hold (this example simplifies and abstracts the setting in the proof of Theorem 3.3.3). Assume that the set G of good objects is indeed dense, and moreover that G contains a dense subset $E \subseteq G$ of *excellent* objects that have additional useful properties. (For example, when G is dense, a random object is not only in G, but with high probability it is also a "hard" function, or an expander.) Now assume that we are able to construct a distribution $\mathbf{T}$ over simple tests that only distinguishes between excellent objects and bad ones; that is, $\mathbf{T}$ solves a promise problem with some "gap" between the "yes" instances (i.e., E) and the "no" instances (i.e., $\{0,1\}^n \setminus G$). Denoting the uniform distribution over $\{0,1\}^n$ by $\mathbf{u}_n$, in this case we have that $\Pr[\mathbf{T}(\mathbf{u}_n) = 1]$ is high, whereas $\Pr[\mathbf{T}(x) = 0]$ is high for every $x \notin G$. Thus, in such a setting, in order to find a good object $x \in G$ it suffices to construct a pseudorandom generator for the tests in the support of $\mathbf{T}$ (see Corollary 3.2.3).

3.2.2 Boolean functions

We first prove the following claim, which corresponds to the setting that was described after Observation 3.2.1 and refers to two relaxations of the observation. Specifically, our goal will be to find some $x \in$ G, and we will use a distribution $\mathbf{T}$ that rejects every $x \notin$ G, with high probability, and accepts a random input, with high probability; and a distribution $\mathbf{w}$ that is pseudorandom for the tests in the support of $\mathbf{T}$.

Lemma 3.2.2 (randomized tests, "hitting" a Boolean function)**.** *Let $n \in \mathbb{N}$, let $G \subseteq \{0,1\}^n$, and let $\delta_1, \delta_2, \delta_3, \delta_4 > 0$ be error parameters.*

- *Let $\mathbf{T}$ be a distribution over $\{\{0,1\}^n \to \{0,1\}\}$ such that $\Pr[\mathbf{T}(\mathbf{u}_n) = 1] \geq 1 - \delta_1$ and such that for every $x \notin G$ it holds that $\Pr[\mathbf{T}(x) = 0] \geq 1 - \delta_2$.*
- *Let $\mathbf{w}$ be a distribution that δ_4-fools all but an δ_3-fraction of the tests in $\mathbf{T}$; that is, the probability over $T \sim \mathbf{T}$ that $\left|\Pr[T(\mathbf{u}_n) = 1] - \Pr[T(\mathbf{w}) = 1]\right| > \epsilon_4$ is at most ϵ_3.*

Then, the probability that $\mathbf{w} \in G$ is at least $1 - \sum_{i\in[4]} \delta_i$.

Proof. To upper-bound the probability that $\mathbf{w} \notin G$, we first show that a *random test* $T \sim \mathbf{T}$ accepts a *pseudorandom input* from the distribution $\mathbf{w}$ with high probability. To see this, let $\mathcal{T}$ be the set of tests in the support of $\mathbf{T}$ that are δ_4-fooled by $\mathbf{w}$; that is,

$\mathcal{T} = \left\{ T \in \mathrm{supp}(\mathbf{T}) : \left| \Pr[T(\mathbf{u}_n) = 1] - \Pr[T(\mathbf{w}) = 1] \right| \leq \delta_4 \right\}$. Then, we have that

$$\begin{aligned}
\Pr[\mathbf{T}(\mathbf{w}) = 1] &= \Pr[\mathbf{T} \in \mathcal{T}] \cdot \Pr[\mathbf{T}(\mathbf{w}) = 1 | \mathbf{T} \in \mathcal{T}] \\
&\quad + \Pr[\mathbf{T} \notin \mathcal{T}] \cdot \Pr[\mathbf{T}(\mathbf{w}) = 1 | \mathbf{T} \notin \mathcal{T}] \\
&\geq \Pr[\mathbf{T} \in \mathcal{T}] \cdot (\Pr[\mathbf{T}(\mathbf{u}_n) = 1 | \mathbf{T} \in \mathcal{T}] - \delta_4) \\
&\quad + \Pr[\mathbf{T} \notin \mathcal{T}] \cdot (\Pr[\mathbf{T}(\mathbf{u}_n) = 1 | \mathbf{T} \notin \mathcal{T}] - 1) \\
&\geq \Pr[\mathbf{T}(\mathbf{u}_n) = 1] - (\delta_3 + \delta_4) \,, \qquad (3.2.1)
\end{aligned}$$

and relying on our hypothesis, Eq. (3.2.1) is lower-bounded by $1 - \delta_1 - \delta_3 - \delta_4$.

Now, note that if $\Pr[\mathbf{w} \notin G]$ is high, then there is significant probability that a random test from $\mathbf{T}$ will reject a pseudorandom input from $\mathbf{w}$. Specifically,

$$\Pr[\mathbf{T}(\mathbf{w}) = 0] \geq \Pr[\mathbf{w} \notin G] \cdot \min_{x \notin G} \left\{ \Pr_{T \sim \mathbf{T}}[T(x) = 0] \right\} \geq \Pr[\mathbf{w} \notin G] - \delta_2 \,. \qquad (3.2.2)$$

By combining Eq. (3.2.2) with the fact that $\Pr[\mathbf{T}(\mathbf{w}) = 0] \leq \delta_1 + \delta_3 + \delta_4$, it follows that $\Pr[\mathbf{w} \notin G] \leq \sum_{i \in [4]} \delta_i$. ■

The following result refers to the setting that was described in the end of Section 3.2.1. The result follows as a special case of Lemma 3.2.2.

Corollary 3.2.3 (randomized tests, "hitting" a Boolean function; a special case)**.** *Let $n \in \mathbb{N}$, and let $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5 > 0$ be error parameters.*

- *Let $G \subseteq \{0,1\}^n$ and $E \subseteq G$ such that $\Pr[\mathbf{u}_n \in E] \geq 1 - \epsilon_1$.*
- *Let $\mathbf{T}$ be a distribution over over $\{\{0,1\}^n \to \{0,1\}\}$ such that for every $x \in E$ it holds that $\Pr[\mathbf{T}(x) = 1] \geq 1 - \epsilon_2$ and for every $x \notin G$ it holds that $\Pr[\mathbf{T}(x) = 0] \geq 1 - \epsilon_3$.*
- *Let $\mathbf{w}$ be a distribution that ϵ_5-fools all but an ϵ_4-fraction of the tests in $\mathbf{T}$; that is, the probability over $T \sim \mathbf{T}$ that $\left| \Pr[T(\mathbf{u}_n) = 1] - \Pr[T(\mathbf{w}) = 1] \right| > \epsilon_5$ is at most ϵ_4.*

Then, the probability that $\mathbf{w} \notin G$ is at least $1 - \sum_{i \in [5]} \epsilon_i$.

Proof. Since $\Pr[\mathbf{u}_n \in E] \geq 1 - \epsilon_1$ and for every $x \in E$ it holds that $\Pr[\mathbf{T}(x) = 1] \geq 1 - \epsilon_2$, we have that $\Pr[\mathbf{T}(\mathbf{u}_n) = 1] \geq 1 - (\epsilon_1 + \epsilon_2)$. We thus invoke Lemma 3.2.2 with $\delta_1 = \epsilon_1 + \epsilon_2$ and with $\delta_i = \epsilon_{i+1}$ for $i \in \{2,3,4\}$. ■

In Lemma 3.2.2 and Corollary 3.2.3 our goal was simply to "hit" the (dense) set G. In contrast, in the following lemma our goal will be to *approximate the density* of the set G. Specifically, we think of $G = p^{-1}(1)$ for some function $p \colon \{0,1\}^n \to \{0,1\}$, and we do not assume that G is dense. We fix any distribution $\mathbf{h}$ such that for every $x \in \{0,1\}^n$, with high probability it holds that $\mathbf{h}(x) = p(x)$. Our conclusion is that any distribution $\mathbf{w}$ that is pseudorandom for the functions in the support of $\mathbf{h}$ is also pseudorandom for p.

Lemma 3.2.4 (randomized tests, "fooling" a Boolean function)**.** *Let $n \in \mathbb{N}$, let $\epsilon > 0$, and let $p\colon \{0,1\}^n \to \{0,1\}$ Assume that there exists a distribution $\mathbf{h}$ over functions $\{0,1\}^n \to \{0,1\}$ such that for every fixed $x \in \{0,1\}^n$ it holds that $\Pr[\mathbf{h}(x) = p(x)] \geq 1 - \epsilon$. Finally, let $\mathbf{w}$ be a distribution over $\{0,1\}^n$ such that for every h in the support of $\mathbf{h}$ it holds that $\left|\Pr_{x\in\{0,1\}^n}[h(x) = 1] - \Pr[h(\mathbf{w}) = 1]\right| \leq \epsilon$. Then,*

$$\left|\Pr_{x\in\{0,1\}^n}[p(x) = 1] - \Pr[p(\mathbf{w})]\right| \leq 5\epsilon\ .$$

Lemma 3.2.4 follows as a special case of a more general claim below (see Lemma 3.2.8), and therefore we omit its proof.

3.2.3 Polynomials over finite fields

We now present several results that follow the high-level intuition from Section 3.2.1 and apply to the setting of polynomials over finite fields. We first prove a result that refers to "hitting" a set $G \subseteq \mathbb{F}^n$ and is very similar to Observation 3.2.1.

Lemma 3.2.5 (randomized tests, "hitting" polynomials)**.** *Let $n \in \mathbb{N}$, let $\mathbb{F}$ be any finite field, let $G \subseteq \mathbb{F}^n$, and let $\epsilon_1, \epsilon_2, \epsilon_3 \in [0,1)$ be three parameters. Assume that*

1. *There exists a distribution $\mathbf{p}$ over $\{\mathbb{F}^n \to \mathbb{F}\}$ such that for every $x \notin G$ it holds that $\Pr[\mathbf{p}(x) = 0] \geq 1 - \epsilon_1$.*
2. *There exists a distribution $\mathbf{w}$ over $\mathbb{F}^n$ such that for some set $\mathcal{H} \subseteq \{\mathbb{F}^n \to \mathbb{F}\}$ satisfying $\Pr[\mathbf{p} \in \mathcal{H}] \geq 1 - \epsilon_2$, for every fixed $p \in \mathcal{H}$ it holds that $\Pr[p(\mathbf{w}) \neq 0] > \epsilon_3$.*

Then, $\Pr[\mathbf{w} \in G] > \epsilon_3 \cdot (1 - \epsilon_2) - \epsilon_1$.

Proof. First observe that $\Pr[\mathbf{p}(\mathbf{w}) \neq 0]$ is high; this is the case since

$$\Pr[\mathbf{p}(\mathbf{w}) \neq 0] \geq \Pr[\mathbf{p} \in \mathcal{H}] \cdot \Pr_{p\sim\mathbf{p}}[p(\mathbf{w}) \neq 0 | p \in \mathcal{H}] > \epsilon_3 \cdot (1 - \epsilon_2)\ . \tag{3.2.3}$$

On the other hand, we also have that

$$\begin{aligned}\Pr[\mathbf{p}(\mathbf{w}) \neq 0] &= \mathbb{E}_{x\sim\mathbf{w}}\left[\Pr_{p\sim\mathbf{p}}[p(x) \neq 0]\right] \\ &\leq \Pr_{x\sim\mathbf{w}}[x \in G] + \Pr_{x\sim\mathbf{w}}[x \notin G] \cdot \max_{x\notin G}\left\{\Pr_{p\sim\mathbf{p}}[p(x) \neq 0]\right\} \\ &\leq \Pr[\mathbf{w} \in G] + \epsilon_1\ ,\end{aligned} \tag{3.2.4}$$

and the lemma follows by combining Eqs. (3.2.3) and (3.2.4). ■

The next result, which follows as a corollary of Lemma 3.2.5, refers to the setting in which we assume that the set G is dense, that $\mathbf{p}$ distinguishes between G and $\neg$G, and that $\mathbf{w}$ is a hitting-set for polynomials that *vanish rarely*.

Corollary 3.2.6 (randomized tests, "hitting" polynomials; a special case)**.** *Let $\epsilon, \rho, \mu, \gamma \in [0,1)$ such that $\rho + \epsilon < 1$, and let $G \subseteq \mathbb{F}^n$ be such that $\Pr[\mathbf{u}_n \in G] \geq 1 - \epsilon$. Assume that there exists a distribution $\mathbf{p}$ over polynomials $p\colon \mathbb{F}^n \to \mathbb{F}$ and a distribution $\mathbf{w}$ over $\mathbb{F}^n$ such that:*

1. *For every fixed $x \in G$ it holds that $\Pr[\mathbf{p}(x) \neq 0] \geq 1 - \rho$, and for every fixed $x \notin G$ it holds that $\Pr[\mathbf{p}(x) = 0] \geq 1 - \gamma$.*

2. *For every $p\colon \mathbb{F}^n \to \mathbb{F}$ in the support of $\mathbf{p}$ that vanishes on at most a $\sqrt{\rho + \epsilon}$ fraction of its inputs it holds that $\Pr[p(\mathbf{w}) \neq 0] > \mu$.*

Then, $\Pr[\mathbf{w} \in G] > \mu \cdot (1 - \sqrt{\rho + \epsilon}) - \gamma$.

Proof. Let $\mathcal{H}$ be the set of polynomials in the support of $\mathbf{p}$ that vanish on at most $\sqrt{\rho + \epsilon}$ of the inputs $x \in \mathbb{F}^n$. Note that $\Pr[\mathbf{p} \in \mathcal{H}] > 1 - \sqrt{\rho + \epsilon}$, because

$$\Pr[\mathbf{p}(\mathbf{u}_n) \neq 0] \geq \Pr[\mathbf{u}_n \in G] \cdot \min_{x \in G}\{\Pr[\mathbf{p}(x) \neq 0]\} \geq 1 - (\rho + \epsilon)\,,$$

and using Markov's inquality. We therefore invoke Lemma 3.2.5 with parameters $\epsilon_1 = \gamma$ and $\epsilon_2 = \sqrt{\rho + \epsilon} < 1$ and $\epsilon_3 = \mu$. ■

In the next argument, instead of trying to "hit" a fixed set $G \subseteq \mathbb{F}^n$, we will fix a polynomial $p : \mathbb{F}^n \to \mathbb{F}$, and try to "fool" p (i.e., we want to construct a pseudorandom generator for p). Indeed, we will need to explain exactly what we mean by "fooling" in the context of functions over finite fields. Towards presenting the argument, let us first define the notion of *randomly computing p by a distribution of functions that is typically over simpler functions*.

Definition 3.2.7 (randomly computing a function)**.** *Let $\mathbb{F}$ be a finite field, let $p : \mathbb{F}^n \to \mathbb{F}$, and let $\mathcal{H}$ be a class of functions $\mathbb{F}^n \to \mathbb{F}$. For $\rho, \rho' > 0$, we say that p can be* randomly computed with error ρ by a distribution $\mathbf{h}$ that is $(1 - \rho')$-typically in $\mathcal{H}$*, if:*

1. *For every $x \in \mathbb{F}^n$ it holds that $\Pr[p(x) = \mathbf{h}(x)] \geq 1 - \rho$.*

2. *The probability that $\mathbf{h} \in \mathcal{H}$ is at least $1 - \rho'$.*

The following claim extends an argument that is implicit in the work of Bogdanov and Viola [BV10, Proof of Lemma 23]. Loosely speaking, our claim is the following: If p can be computed with small error by a distribution $\mathbf{h}$ that is typically in $\mathcal{H}$, then any distribution $\mathbf{w}$ over $\mathbb{F}^n$ that "fools" every $h \in \mathcal{H}$ also "fools" p, where "fooling" a function f means that for some (fixed) mapping $\zeta : \mathbb{F} \to \mathcal{C}$ it holds that $\Big|\mathbb{E}[\zeta(f(\mathbf{w}))] - \mathbb{E}[\zeta(f(\mathbf{u}_n))]\Big|$ is small. It is useful to think of $\zeta\colon \mathbb{F} \to \mathcal{C}$ as a fixed non-trivial character $e : \mathbb{F} \to \mathcal{C}$, in which case we have that $\max_{v,w \in \mathbb{F}}\{|\zeta(v) - \zeta(w)|\} = 2$.

Lemma 3.2.8 (randomized tests, "fooling" polynomials; an extension of a claim that is implicit in [BV10]). *Let $n \in \mathbb{N}$, let $\mathbb{F}$ be any finite field, let $\epsilon > 0$. Also, let $\zeta\colon \mathbb{F} \to \mathcal{C}$, and let $\delta = \max_{v,w\in\mathbb{F}}\{|\zeta(v) - \zeta(w)|\}$. Let $p\colon \mathbb{F}^n \to \mathbb{F}$, and assume that there exists a distribution $\mathbf{h}$ over polynomials $\mathbb{F}^n \to \mathbb{F}$ such that for every fixed $x \in \mathbb{F}^n$ it holds that $\Pr[\mathbf{h}(x) = p(x)] \geq 1 - \epsilon$. Finally, let $\mathbf{w}$ be a distribution over $\mathbb{F}^n$ such that for every polynomial h in the support of $\mathbf{h}$ it holds that $\left|\mathbb{E}_{x\in\mathbb{F}^n}[\zeta(h(x))] - \mathbb{E}[\zeta(h(\mathbf{w}))]\right| \leq \epsilon$. Then,*

$$\left|\mathbb{E}_{x\in\mathbb{F}^n}[\zeta(p(x))] - \mathbb{E}[\zeta(p(\mathbf{w}))]\right| \leq (2\delta + 1)\cdot\epsilon\,.$$

Proof. Let $\mathbf{u}_n$ be the uniform distribution over $\mathbb{F}^n$. For simplicity of notation, define $p' = \zeta \circ p\colon \mathbb{F}^n \to \mathcal{C}$, and for every h in the support of $\mathbf{h}$, define $h' = \zeta \circ h\colon \mathbb{F}^n \to \mathcal{C}$. Also denote by $\mathbf{h}'$ the distribution that is obtained by sampling $h \sim \mathbf{h}$ and outputting $h' = \zeta \circ h$. By the triangle inequality,

$$\begin{aligned}\left|\mathbb{E}[p'(\mathbf{u}_n)] - \mathbb{E}[p'(\mathbf{w})]\right| \leq{}& \left|\mathbb{E}[p'(\mathbf{u}_n)] - \mathbb{E}[\mathbf{h}'(\mathbf{u}_n)]\right| \\ &+ \left|\mathbb{E}[\mathbf{h}'(\mathbf{u}_n)] - \mathbb{E}[\mathbf{h}'(\mathbf{w})]\right| \\ &+ \left|\mathbb{E}[\mathbf{h}'(\mathbf{w})] - \mathbb{E}[p'(\mathbf{w})]\right|\,. \end{aligned} \tag{3.2.5}$$

To upper bound the first item in Equation (3.2.5), note that

$$\begin{aligned}\left|\mathbb{E}[p'(\mathbf{u}_n)] - \mathbb{E}[\mathbf{h}'(\mathbf{u}_n)]\right| &\leq \mathbb{E}_{x\sim\mathbb{F}^n,h\sim\mathbf{h}}\left[\left|p'(x) - h'(x)\right|\right] \\ &\leq \mathbb{E}_{x\in\mathbb{F}^n}\left[\Pr_{h\sim\mathbf{h}}[h(x) \neq p(x)]\cdot\max_{v,w\in\mathbb{F}}\{|\zeta(v) - \zeta(w)|\}\right] \\ &\leq \delta\cdot\epsilon\,,\end{aligned}$$

where the last inequality holds because for every fixed $x \in \mathbb{F}^n$ we have that $\Pr_{h\sim\mathbf{h}}[h(x) \neq p(x)] \leq \epsilon$. The third item in Equation (3.2.5) is similarly upper bounded by $\delta\cdot\epsilon$, by replacing the uniform choice of $x \in \mathbb{F}^n$ with a choice of $x \sim \mathbf{w}$.

To upper bound the second item in Equation (3.2.5), note that

$$\left|\mathbb{E}[\mathbf{h}'(\mathbf{u}_n)] - \mathbb{E}[\mathbf{h}'(\mathbf{w})]\right| \leq \mathbb{E}_{h\sim\mathbf{h}}\left[\left|\mathbb{E}[h'(\mathbf{u}_n)] - \mathbb{E}[h'(\mathbf{w})]\right|\right] \leq \epsilon\,,$$

where we used the hypothesis that for every polynomial h in the support of $\mathbf{h}$ it holds that $\left|\mathbb{E}_{x\in\mathbb{F}^n}[\zeta(h(x))] - \mathbb{E}[\zeta(h(\mathbf{w}))]\right| \leq \epsilon$. ■

3.3 Constant-depth circuits

3.3.1 The main results

Let us first state the threshold values for quantified derandomization of $\mathcal{AC}^0$, and then turn to describe our algorithms for quantified derandomization. Goldreich and

Wigderson showed that the value $B(n) = 2^{n/\log^{0.99\cdot D}(n)}$ is a threshold value for quantified derandomization of depth-D circuits. Specifically, they reduced the *standard* derandomization problem of depth-d circuits to the problem of quantified derandomization of circuits of depth $D \gg d$ with $B(n) = 2^{n/\log^{D-O(d)}(n)}$ (see [GW14, Thm 3.4 (full version)]). Since their work, Cheng and Li [CL16] improved the known techniques for error-reduction within $\mathcal{AC}^0$, which allows us to further decrease the threshold value, as follows:

Theorem 3.3.1 (threshold for quantified derandomization of $\mathcal{AC}^0$)**.** *For any $d \geq 2$ and $D > d + 11$, the standard derandomization problem of depth-d circuits reduces in deterministic polynomial-time to the quantified derandomization problem of circuits of depth D that accept all but $B(n) = 2^{n/\log^{D-d-11}(n)}$ of their inputs.*

Our main result for $\mathcal{AC}^0$ circuits is a derandomization of depth-D circuits with $B(n) = 2^{n/\log^{D-2}(n)}$ exceptional inputs; this value of $B(n)$ *is only slightly smaller* than the threshold value in Theorem 3.3.1. The quantified derandomization algorithm runs in time that is significantly faster than the current state-of-the-art for derandomizing $\mathcal{AC}^0$:

Theorem 3.3.2 (quantified derandomization of $\mathcal{AC}^0$ with $B(n) = 2^{n/\log^{D-2}(n)}$)**.** *For any $D \geq 2$, there exists a hitting-set generator with seed length $\tilde{O}(\log^3(n))$ for the class of depth-D circuits over n input bits that accept all but at most $B(n) = 2^{\Omega\left(n/\log^{D-2}(n)\right)}$ of their inputs.*

We stress that the power of the poly-logarithm in the seed length in Theorem 3.3.2 does not depend on the depth D. Any *standard* hitting-set generator for $\mathcal{AC}^0$ (i.e., with $B(n) = 2^n/2$) with such a seed length would be a major breakthrough, and in particular would significantly improve the lower bounds of Håstad for $\mathcal{AC}^0$ [Hås87] (see, e.g., [Vad12, Prob. 7.1] and [TX13, "Barriers to Further Progress"]).

The values of $B(n)$ in Theorems 3.3.1 and 3.3.2 are indeed very close, yet the smaller value allows for derandomization in time $2^{\tilde{O}(\log^3(n))}$ whereas the larger value is a threshold for standard derandomization. This represents a progress towards the goal of the quantified derandomization approach, which is to *close* the gap between the two parameters: That is, to either increase the value of $B(n)$ in Theorem 3.3.2, or decrease the value of $B(n)$ in Theorem 3.3.1, and obtain a standard derandomization of $\mathcal{AC}^0$.

Theorem 3.3.2 is a special case of the following, more general result, which extends the main theorem of Goldreich and Wigderson [GW14]. Their algorithm works with logarithmic seed and $B(n) = 2^{n^{1-\Omega(1)}}$ exceptional inputs. The following result is parametrized (by the parameter t), and can work with more than $2^{n^{1-\Omega(1)}}$ exceptional inputs, at the expense of a longer (i.e., super-logarithmic) seed; Theorem 3.3.2 is the special case where both $B(n)$ and the seed are the largest possible in this result.

Theorem 3.3.3 (quantified derandomization of $\mathcal{AC}^0$: a general trade-off)**.** *For any $D \geq 2$ and $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) \leq O(\log(n))$, there exists a hitting-set generator that uses a*

seed of length $\tilde{O}(t^2 \cdot \log(n))$ for the class of depth-D circuits over n input bits that accept all but at most $B(n) = \exp\left(n^{1-1/\Omega(t)}/t^{d-2}\right)$ of their inputs.

Indeed, the main result in [GW14] is essentially obtained (up to a $\operatorname{poly}\log\log(n)$ factor in the seed length) by setting $t = O(1)$, whereas Theorem 3.3.2 is obtained by setting $t = O(\log(n))$. Theorem 3.3.3 is based on a *new derandomization of Hastad's switching lemma*, which is our main technical contribution in this section.

Theorem 3.3.4 (new width-dependent derandomization of the switching lemma; see Theorem 3.3.14)**.** *Let $m, n \in \mathbb{N}$, let $w \leq O(\log(m))$, and let $\delta > 0$. Then, there exists an algorithm that gets as input a random seed of length $\tilde{O}(w^2 \cdot \log(mn/\delta))$, runs in time $\operatorname{poly}(n)$, and outputs a restriction $\rho \in \{0,1,\star\}^n$ such that for every depth-2 formula $F : \{0,1\}^n \to \{0,1\}$ of size m and width w, with probability $1 - O(\delta)$ the following holds:*

- *The number of variables that are kept alive by ρ is $\Omega(n/w)$.*
- *There exist "lower-sandwiching" and "upper-sandwiching" formulas $F^{\mathtt{low}}$ and $F^{\mathtt{up}}$ for F (i.e., for every $x \in \{0,1\}^n$ it holds that $F^{\mathtt{low}}(x) \leq F(x) \leq F^{\mathtt{up}}(x)$) such that both $F^{\mathtt{low}}{\restriction_\rho}$ and $F^{\mathtt{up}}{\restriction_\rho}$ can be computed by decision trees of depth $O(\log(1/\delta))$, and each of them agrees with $F{\restriction_\rho}$ on $1 - \delta$ of the inputs (in the subcube that corresponds to ρ).*

The actual bound on the seed length of the algorithm from Theorem 3.3.4 in our result is better when $\delta = \omega(1/m)$; see Theorem 3.3.14 for the precise expression. Note that the seed length depends on the width of the formula F; previous derandomizations of the switching lemma can also be adapted to depend on the width, but when the width is $o(\log(n))$ the seed length in Theorem 3.3.4 is significantly shorter than in these adaptations; see Section 3.3.2 for further details.

3.3.2 Proof overviews

Theorem 3.3.2 is a special case of the more general Theorem 3.3.3. However, since there is a simple and more direct way to prove Theorem 3.3.2, we describe this simpler way first, and only then turn to the describe the proof of the more general theorem.

Let C be a depth-D circuit that accepts all but $B(n) = \Omega\left(2^{n/\log^{D-2}(n)}\right)$ of its inputs. The hitting-set generator first uses pseudorandom restrictions to simplify C to a depth-2 circuit, by fixing values for all but $n' = \Omega(n/\log^{D-2}(n))$ of the variables. These pseudorandom restrictions are chosen using an adaptation of the derandomized switching lemma of Trevisan and Xue [TX13] (either Tal's [Tal17] improvement or the adapted version in Proposition 3.3.12), which requires a seed of length $\tilde{O}(\log^3(n))$. At this point, there are $n' \geq \log(B(n)) + 1$ living variables, and therefore the simplified circuit (over n' input bits) has acceptance probability at least $1/2$ (since C has at most $B(n)$ unsatisfying inputs). Hence, we can use any pseudorandom generator for depth-2 circuits with seed length at most $\tilde{O}(\log^3(n))$ (e.g., that of De *et al.* [DET+10]) in order

to fix values for the remaining n' variables, thus finding a satisfying input for C, with high probability.[4]

Turning to the more general Theorem 3.3.3, the high-level structure of its proof is similar to that of the proof of Theorem 3.3.2: We first use a derandomized switching lemma to radically simplify the circuit, while keeping more than $\log(B(n))$ variables alive, and then use a pseudorandom generator for the simplified circuit to find a satisfying input. The key difference from Theorem 3.3.2 is that the first step uses a new derandomization of the switching lemma, which we establish.

The new derandomization of the switching lemma depends on the *width* (i.e., bottom fan-in) of the depth-2 formula that we want the restriction to simplify. Previous known derandomizations of the lemma can also be adapted to depend on the width of the formula: For typical settings of the parameters (i.e., polynomial size and polynomially-small error), the derandomization of Goldreich and Wigderson [GW14] can be adapted to yield a seed length of $\tilde{O}(2^w) \cdot \log(n)$ for formulas of width w (see Proposition 3.3.18), and the derandomization of Trevisan and Xue [TX13] can be adapted (using the pseudorandom generator of Gopalan, Meka, and Reingold [GMR13]) to yield a seed length of $\tilde{O}(w) \cdot \log^2(n)$ (see Proposition 3.3.12). We show a derandomization that requires a seed of length $\tilde{O}(w^2 \cdot \log(n))$ (see Theorem 3.3.14). Indeed, in this new result, the dependency of the seed length on w is exponentially better than in [GW14], and the seed length is shorter than in [TX13] for any $w = o(\log(n))$. The caveat, however, is that we do not show that the formula itself is simplified in the subcube corresponding to the restriction; instead, we show that the formula is *approximated* by a decision tree of bounded depth in this subcube (i.e., there exists such a decision tree that agrees with the formula on almost all inputs in the subcube). This weaker conclusion suffices for our main application (i.e., for Theorem 3.3.3) as well as for all other applications of derandomized switch lemmas that we are aware of.

Our starting point in the proof of this lemma is a result of Gopalan, Meka, and Reingold [GMR13], which asserts that for any depth-2 formula F of width w and any $\beta > 0$, there exists a formula $F^{\mathtt{low}}$ of width at most w and size at most $m' = 2^{\tilde{O}(w) \cdot \log\log(1/\beta)}$ such that $F^{\mathtt{low}}$ is "lower-sandwiching" for F (i.e., $F^{\mathtt{low}}(x) \leq F(x)$ for all $x \in \{0,1\}^n$) and $\Pr_{x \in \{0,1\}^n}[F(x) \neq F^{\mathtt{low}}(x)] \leq \beta$. Now, since $F^{\mathtt{low}}$ is both small (i.e., m' is upper bounded) and of bounded width, we can find a restriction that simplifies it using a relatively short seed; specifically, we can use an adapted version of the lemma of [TX13] (see Proposition 3.3.12), and the required seed length (when we want the probability of error to be $1/\mathrm{poly}(n)$) is only $\tilde{O}(w) \cdot \log(m') \cdot \log(n) = \tilde{O}(w^2) \cdot \log(n) \cdot \log\log(1/\beta)$.

The main challenge that underlies this approach is that, while $F^{\mathtt{low}}$ agrees with F

[4] Actually, there is one minor subtlety in this description: In the derandomizations of [TX13; Tal17], the expected number of living variables is close to $n/\log^{d-2}(n)$, but it is *not* guaranteed that approximately this many variables remain alive with high (or even constant) probability. Nevertheless, the latter does hold when instantiating their generic construction in a specific manner; see the proof of Theorem 3.3.3 for further details.

on most inputs $x \in \{0,1\}^n$, it is not clear that $F^{\mathtt{low}}$ also agrees with F on most inputs *in the subcube that corresponds to* ρ; that is, it is not guaranteed that $F^{\mathtt{low}}|_\rho$ will agree with $F|_\rho$ on most of *their* inputs. To make sure that $F^{\mathtt{low}}|_\rho$ will agree with $F|_\rho$ on most of their inputs, we will choose ρ such that it "fools" additional tests that check whether or not $F^{\mathtt{low}}|_\rho$ and $F|_\rho$ indeed typically agree. To design these tests we use the randomized tests technique: Specifically, a natural randomized test to decide whether or not $F^{\mathtt{low}}|_\rho$ and $F|_\rho$ typically agree is to sample random inputs inside the subcube that corresponds to ρ, and accept if and only if $F^{\mathtt{low}}|_\rho$ and $F|_\rho$ agree on the sampled inputs.

Indeed, *the residual tests under this distribution are simpler (in any reasonable sense) than any deterministic test* that decides whether or not $F^{\mathtt{low}}|_\rho$ and $F|_\rho$ agree on most of their inputs. The remaining task is thus to construct a hitting-set generator with high density for these residual tests. We will now describe how to do so, relying both on the specific details of the construction of $F^{\mathtt{low}}$ from [GMR13], in order to construct circuits with a specific structure that will be convenient for us for each residual test, and on relaxations of the randomized tests technique that follow the ones suggested in the end of Section 3.2.1 (i.e., using Corollary 3.2.3).

We want to use the lemma to simplify polynomially-many depth-2 formulas (i.e., simplify an entire "layer" of a constant-depth circuit). Thus, we want that for every fixed formula F it will hold that $F^{\mathtt{low}}|_\rho$ and $F|_\rho$ agree on an all but an α-fraction of their inputs, where $\alpha = 1/\mathrm{poly}(n)$. We say that a restriction ρ is *good* if $F^{\mathtt{low}}|_\rho$ and $F|_\rho$ agree with probability at least $1-\alpha$. If we start from a formula $F^{\mathtt{low}}$ with the approximation parameter $\beta = \mathrm{poly}(\alpha)$, then almost all restrictions ρ' are *excellent*, in the sense that $F^{\mathtt{low}}|_{\rho'}$ and $F|_{\rho'}$ agree with probability $1-\sqrt{\beta} \gg 1-\alpha$. For each fixed F and $F^{\mathtt{low}}$, to distinguish between excellent restrictions and restrictions that are not good, the distribution $\mathbf{T}$ of tests uniformly samples $\mathrm{poly}(\alpha)$ inputs inside the subcube that corresponds to its input restriction ρ, and accepts ρ if and only if F and $F^{\mathtt{low}}$ agree on the sampled inputs.

The next step is to show that each residual test $T \in \mathbf{T}$ can be computed by a circuit with a convenient structure. To do so, we observe that the construction of $F^{\mathtt{low}}$ in [GMR13] is based on a sequence of *specific syntactic modifications* to F: Each syntactic modification is a simplification of a quasi-sunflower, a notion introduced by Rossman [Ros14] (for more specific details see Section 3.3.4.1). We define the tests $T \in \mathbf{T}$ to accept if and only if the *specific syntactic modifications* used to transform F into $F^{\mathtt{low}}$ did not affect the formula at the relevant inputs. Then, we show that each such test T can be decided by a depth-3 circuit with a top AND gate and bottom fan-in w (relying on the hypothesis that the original formula F has width w; see Claim 3.3.15.3).

Now, since almost all restrictions are excellent, and each excellent restriction is accepted with high probability by $\mathbf{T}$, it follows that almost all tests in $\mathbf{T}$ belong to the subset $\mathbf{T}' \subseteq \mathbf{T}$ of tests that accept almost all of their input restrictions. We will in fact construct a hitting-set generator for the residual tests $T \in \mathbf{T}'$. This can be done relying both on the fact that $T \in \mathbf{T}'$ has very high acceptance probability and on the fact that it can be computed by a depth-3 circuit with a top AND gate and bottom fan-in w (the

latter allows us to use the pseudorandom generator of [GMR13] for formulas of small width; see Claim 3.3.15.4).

To prove Theorem 3.3.3, we will repeat the following step: First reduce the width of the formulas in the next-to-bottom layer by a pseudorandom restriction (see Claim 3.3.16.1), and then use the new switching lemma to approximate the circuit by a circuit in which all the formulas in the next-to-bottom layer are simplified (and thus the latter circuit has smaller depth). Since all our approximations are "lower-sandwiching", any satisfying input for the latter circuit is also satisfying for the former circuit.

3.3.3 Proof of Theorem 3.3.1

Let $c = D - d - 11$. Starting from a depth-d circuit $C : \{0,1\}^n \to \{0,1\}$, we will employ error-reduction within $\mathcal{AC}^0$, by first sampling inputs for C using the seeded extractor of Cheng and Li [CL16], and then taking the disjunction of the evaluation of C on these inputs. The extractor will be of depth $c+10$, and will work for min-entropy $n'/\log^c(n')$, where n' is the number of random bits that it uses. Thus, this construction will yield a circuit $C' : \{0,1\}^{n'} \to \{0,1\}$ of depth $D = d + (c+10) + 1$ that accepts all but $2^{n'/\log^c(n')} = 2^{n'/\log^{D-d-11}(n')}$ of its inputs. Details follow.

Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of depth d. We will rely on the following theorem from [CL16], which we cite with minor changes of notation:

Theorem 3.3.5 (an $\mathcal{AC}^0$-computable seeded extractor [CL16, Thm 1.5])**.** *For any constant $c \in \mathbb{N}$, and $k = \Omega(n'/\log^c(n'))$ and any $\epsilon = 1/\mathrm{poly}(n')$, there exists an explicit construction of a strong (k,ϵ)-extractor $\mathtt{Ext} : \{0,1\}^{n'} \times \{0,1\}^d \to \{0,1\}^n$ that can be computed by an $\mathcal{AC}^0$ circuit of depth $c+10$, where $d = O(\log(n))$, $n = k^{\Omega(1)}$ and the extractor family has locality $O(\log^{c+5}(n))$.*

We will not need the strongness property or the locality property in the current proof. Let $n' = \mathrm{poly}(n)$ such that for $k = \Omega(n'/\log^c(n'))$ it holds that $n = k^{\Omega(1)}$, and let $\mathtt{Ext} : \{0,1\}^{n'} \times \{0,1\}^d \to \{0,1\}^n$ be the seeded extractor from Theorem 3.3.5, instantiated with error parameter $\epsilon = 1/4$. We construct a circuit $C' : \{0,1\}^{n'} \to \{0,1\}$ that first computes the values $\mathtt{Ext}(x,z)$, for each possible seed $z \in \{0,1\}^d$, then evaluates C on each value $E(x,z)$, and finally takes an OR of these evaluations; that is, $C'(x) = \vee_{z\in\{0,1\}^s} C(\mathtt{Ext}(x,z))$.

Note that C' has depth D and size $\mathrm{poly}(n)$. Also note that the number of inputs $x \in \{0,1\}^{n'}$ for which $\Pr_z[C(\mathtt{Ext}(x,z))] < 1/4$ is at most $2^{n'/\log^c(n')}$. [5] In particular, C' accepts all but at most $2^{n'/\log^c(n')}$ of its inputs, and for each satisfying input x for C', we can find a corresponding satisfying input for C among $\{\mathtt{Ext}(x,z)\}_{z\in\{0,1\}^s}$.

3.3.4 Proofs of Theorems 3.3.2 and 3.3.3

The first step towards proving Theorems 3.3.2 and 3.3.3 is to establish a derandomized switching lemma that simplifies depth-2 formulas of *bounded-width*; that is, to prove

[5] Otherwise, the uniform distribution on such inputs yields a source X of min-entropy $n'/\log^c(n')$ such that C distinguishes $\mathtt{Ext}(X)$ from the uniform distribution over $\{0,1\}^n$ with probability $1/4$.

Theorem 3.3.4. After presenting several required definitions in Section 3.3.4.1, we prove the lemma in Section 3.3.4.2. Then, in Section 3.3.4.3, we use the lemma to prove Theorems 3.3.2 and 3.3.3.

3.3.4.1 Preliminary definitions, and results from [GMR13]

For any restriction $\rho \in \{0,1,\star\}^n$, denote by $\mathfrak{C}(\rho)$ the subcube that corresponds to the living variables under ρ; that is, $\mathfrak{C}(\rho) = \{x \in \{0,1\}^n : \forall i \in [n] \text{ s.t. } \rho_i \neq \star \text{ it holds that } x_i = \rho_i\}$. We identify strings $r \in \{0,1\}^{(q+1)\cdot n}$, where $n, q \in \mathbb{N}$, with restrictions $\rho = \rho_r \in \{0,1,\star\}^n$, as follows: Each variable is assigned a block of $q+1$ bits in the string; the variable remains alive if the first q bits in the block are all zeroes, and otherwise takes the value of the $(q+1)^{th}$ bit. When we refer to a "block" in the string that corresponds to a restriction, we mean a block of $q+1$ bits that corresponds to some variable. When we say that a restriction is chosen from a distribution $\mathbf{r}$ over $\{0,1\}^{(q+1)\cdot n}$, we mean that a string is chosen according to $\mathbf{r}$, and interpreted as a restriction. Moreover, when we say that an algorithm "reads bits" in the restriction, we mean that it reads bits in the corresponding string.

In addition, we will sometimes identify a *pair* of strings $y \in \{0,1\}^{q\cdot n}$ and $z \in \{0,1\}^n$ with a restriction $\rho = \rho_{y,z}$ such that the string y determines which variables ρ keeps alive, and the string z determines the values that ρ assigns to the fixed variables. Specifically, the restriction $\rho = \rho_{y,z}$ is the restriction ρ_r that is obtained by combining y and z to a string r by appending a bit from z to each block of q bits in y.

Throughout the section, whenever we consider a depth-2 formula for a function $F : \{0,1\}^n \to \{0,1\}$, we allow the formula to be a redundant representation of F (i.e., not necessarily the most concise representation of F as a formula), and in particular we allow formulas in which some clauses are simply constants. We will identify any clause of a depth-2 formula with the corresponding subset of the literals; the clause is a conjunction of the literals if the formula is a DNF, and otherwise it is a disjunction of the literals. We say that a function $F^{\mathtt{low}} : \{0,1\}^n \to \{0,1\}$ is lower-sandwiching for F if for every $x \in \{0,1\}^n$ it holds that $F^{\mathtt{low}}(x) \leq F(x)$. Similarly, we say that $F^{\mathtt{up}} : \{0,1\}^n \to \{0,1\}$ is upper-sandwiching for F if for every $x \in \{0,1\}^n$ it holds that $F(x) \leq F^{\mathtt{up}}(x)$.

Refinements: Definition and basic facts. We need several definitions that are related to the results of Gopalan, Meka, and Reingold [GMR13]. Their main theorem involves a process of *sparsification* of a depth-2 formula. The sparsification process is iterative: In each iteration, they identify a *quasi-sunflower* in the formula (a notion that was introduced by Rossman [Ros14]), and simplify the quasi-sunflower using one of two operations. The first operation is simply the removal of a clause from the formula; and the second operation is the removal of a set $f_1, ..., f_u$ of $u \geq 2$ clauses, replacing them with a new clause that consists of the set of literals that are shared by all the u clauses (i.e., replacing $f_1, ..., f_u$ with the clause $\bigcap_{j\in[u]} f_j$). The following definition generalizes

this sparsification process.[6]

Definition 3.3.6 (refinements of a depth-2 formula)**.** *Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula with at least two clauses. We define the following three syntactic operations on F, which we call* refinement steps*.*

1. *A* removal step *is simply the removal of a clause from F.*
2. *A* merging step *is the removal of $u \geq 2$ clauses $f_1, ..., f_u$ from F, and the addition of a new clause that consists of the set of literals that appear in all the u clauses (i.e., replacing $f_1, ..., f_u$ with the new clause $\bigcap_{j \in [u]} f_j$). If $\bigcap_{j \in [u]} f_j = \emptyset$, then the new clause computes the constant one function if F is a DNF, and the constant zero function if F is a CNF.*
3. *A* clean-up step *is the removal of one or more clauses that compute the constant zero function from a DNF, or of one or more clauses that compute the constant one function from a CNF.*

We say that a depth-2 formula $F' : \{0,1\}^n \to \{0,1\}$ is a refinement *of another depth-2 formula $F : \{0,1\}^n \to \{0,1\}$ if F' can be obtained from F either by a sequence of removal steps and clean-up steps, or by a sequence of merging steps and clean-up steps.*

We now state some basic facts about refinements, which will be useful for us later on. The following two facts follow from Definition 3.3.6:

Fact 3.3.7 (refinements under negations)**.** *Let $F : \{0,1\}^n \to \{0,1\}$ and $F' : \{0,1\}^n \to \{0,1\}$ be depth-2 formulas. Then, F' is a refinement of F if and only if $\neg(F')$ is a refinement of $\neg F$.*

Fact 3.3.8 (sandwiching refinements)**.** *Let $F : \{0,1\}^n \to \{0,1\}$ be a DNF. Then, any refinement of F that is obtained by a sequence of removal steps and clean-up steps is lower-sandwiching for F, and any refinement of F that is obtained by a sequence of merging steps clean-up steps is upper-sandwiching for F.*

Loosely speaking, the following claim asserts that if F' is a refinement of F, then for any restriction ρ it holds that $(F')\restriction_\rho$ is a refinement of $F\restriction_\rho$. That is, intuitively, restricting both F and F' by ρ does not affect the fact that the latter formula is a refinement of the former.

Claim 3.3.9 (refinements under restrictions)**.** *Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula of width w and size m, and let $F' : \{0,1\}^n \to \{0,1\}$ be a refinement of F. Then, for any restriction $\rho \in \{0,1,\star\}^n$ it holds that $F\restriction_\rho$ can be computed by a depth-2 formula Φ of width w and size m such that $F'\restriction_\rho$ is a refinement of Φ.*

The proof of Claim 3.3.9 relies on an elementary (and tedious) case analysis, so we defer it to Appendix 3.3.5.2.

[6] The reason that we need this generalization is in order to facilitate the proof of Claim 3.3.9; this is also the reason that we allow formulas to have redundant clauses that compute constant functions.

Two theorems from [GMR13]. For $\epsilon > 0$ and two Boolean functions F and F' over a domain $\mathfrak{D}$, we say that F and F' are ϵ-close if $\Pr_{x \in \mathfrak{D}}[F(x) = F'(x)] \geq 1 - \epsilon$. We say that F' is an ϵ-refinement of F if F' is both a refinement of F, and ϵ-close to F. Similarly, we say that F' is an ϵ-lower-sandwiching refinement (resp., ϵ-upper-sandwiching refinement) of F if F' is both ϵ-close to F and a lower-sandwiching (resp., upper-sandwiching) refinement of F. Then, the main result of Gopalan, Meka, and Reingold [GMR13] can be stated as follows:

Theorem 3.3.10 ([GMR13, Thm 1.2]). *Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula of width w, and let $\beta > 0$. Then, there exist β-lower-sandwiching and β-upper-sandwiching refinements of F, denoted by $F^{\mathtt{low}}$ and $F^{\mathtt{up}}$, respectively, such that the size of $F^{\mathtt{low}}$ and of $F^{\mathtt{up}}$ is at most $m' = 2^{\tilde{O}(w) \cdot \log\log(1/\beta)}$, and their width is at most w.*

We will also need a pseudorandom generator construction from [GMR13]. In fact, we will rely on an assertion from the proof of their generator construction.

Theorem 3.3.11 ([GMR13, In the proof of Thm 3.1]). *Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula of width w, and let $\delta_0 > 0$. Then, every δ-almost t-wise independent distribution δ_0-fools F, where $\log(1/\delta) = O(w^2 \cdot \log^2(w) + w \cdot \log(w) \cdot \log(1/\delta_0))$ and $t = O(w^2 \cdot \log(w) + w \cdot \log(1/\delta_0))$.*

3.3.4.2 Width-dependent derandomizations of the switching lemma

In the proposition statements in this section, the letter n denotes the number of input bits for a formula, the number of clauses (i.e., size) is denoted by m, the width is denoted by w, and $\delta > 0$ is an error parameter (which will typically take the value $\delta = 1/\mathrm{poly}(n)$ in our applications). As a first step, we need to adapt the derandomized switching lemma of Trevisan and Xue [TX13] such that it will depend on the width of the depth-2 formula that we wish to "switch". Then, we will state and prove our new derandomized switching lemma, which is the main technical part in this section.

Proposition 3.3.12 (an adaptation of the derandomized switching lemma of [TX13]). *Let $m : \mathbb{N} \to \mathbb{N}$, let $w : \mathbb{N} \to \mathbb{N}$ such that $w(n) \leq O(\log(m(n)))$, and let $\delta : \mathbb{N} \to [0,1)$ such that $\delta(n) \leq 2^{-O(w(n))}$. Let $\mathbf{r}$ be a distribution over $\{0,1\}^{O(\log(w)) \cdot n}$ that is δ'-almost t'-wise independent, where $\log(1/\delta') = O(t') = \tilde{O}(w) \cdot \log(1/\delta) \cdot \log(m) + O(\log(n/\delta))$. Then, for any depth-2 formula $F : \{0,1\}^n \to \{0,1\}$ of width $w = w(n)$ and size $m = m(n)$, with probability at least $1 - 2\delta$ (where $\delta = \delta(n)$) over choice of $\rho \sim \mathbf{r}$ it holds that:*

1. *The restricted formula $F\restriction_\rho$ can be computed by a decision tree of depth $D = O(\log(1/\delta))$.*
2. *The number of variables that are kept alive by ρ is at least $\Omega(n/w)$.*

In particular, a restriction $\rho \sim \mathbf{r}$ can be sampled using a seed of length $\tilde{O}(w) \cdot \log(1/\delta) \cdot \log(m) + O(\log(n/\delta))$.

Proof. Loosely speaking, the main lemma of Trevisan and Xue [TX13] reduces the task of finding a restriction that simplifies F to the task of "fooling" a large number

of auxiliary CNFs. Going through their proof, we observe is that if F has width w, then each of the auxiliary CNFs also has width (roughly) w; that is, their proof can be adapted to show the following:

Lemma 3.3.13 (a variation on [TX13, Lem 7])**.** *Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula of size m and width w. For $q \in \mathbb{N}$ and $p = 2^{-q}$, let $\rho \in \{0,1,\star\}^n$ be a restriction that is chosen according to a distribution over $\{0,1\}^{(q+1)\cdot n}$ that δ_0-fools all CNFs of width $w' = w \cdot (q+1)$. Then, the probability that $F\restriction_\rho$ cannot be computed by a decision tree of depth D is at most $2^{D+w+1} \cdot (5pw)^D + \delta_0 \cdot 2^{(D+1)\cdot(2\cdot w+\log(m))}$.*

The proof of Lemma 3.3.13 is a relatively straightforward adaptation of the original proof in [TX13], so we defer it to Appendix 3.3.5.1. We will use the lemma with the parameters $p = 1/O(w)$ and $\delta_0 = 2^{-O(D\cdot(w+\log(m)))}$, in order to get the probability of error down to δ. Relying on Theorem 3.3.11, the auxiliary CNFs of width w' are δ_0-fooled by $\mathbf{r}$, [7] and therefore with probability $1-\delta$ it holds that $F\restriction_\rho$ can be computed by a decision tree of depth D.

The expected number of variables that the pseudorandom restriction leaves alive is $\Omega(n/w)$ (because the distribution on each block of $O(\log(w))$ bits in $\mathbf{r}$, which corresponds to a variable, is of statistical distance at most δ' from uniform, where $\delta' < 2^{-w}$). Since $\mathbf{r}$ is δ'-almost t'-wise independent, where $\delta' < 1/\text{poly}(n/\delta)$ and $t' > O(\log(w))$, the blocks in $\mathbf{r}$ that correspond to each variable are $\frac{1}{\text{poly}(n/\delta)}$-almost $O(1)$-wise independent. Relying on Fact 2.1.2, the probability that $\Omega(n/w)$ variables remain alive is at least $1-\delta$. ■

We mention that the derandomized switching lemma of Goldreich and Wigderson [GW14, second step of the proof of Lemma 3.3] can also be adapted to depend on the width w of the formula that we want to "switch"; in this case, the required seed length is $\tilde{O}(w) \cdot 2^w \cdot \log(1/\delta)$, where δ is the probability of error (and the target depth of the decision tree is $D = O(\log(1/\delta))$). We provide the details in Appendix 3.3.5.1. We now turn to state the new width-dependent derandomization of the switching lemma and prove it:

Theorem 3.3.14 (a new width-dependent derandomization of the switching lemma)**.** *Let $m : \mathbb{N} \to \mathbb{N}$, let $w : \mathbb{N} \to \mathbb{N}$ such that $w(n) \leq O(\log(m(n)))$, let $\delta : \mathbb{N} \to [0,1)$, and let $\alpha : \mathbb{N} \to [0,1)$. Let $\delta' > 0$ and $t' \in \mathbb{N}$ such that $\log(1/\delta') = O(t') = \tilde{O}(w^2) \cdot \log(1/\delta) \cdot \log\log(m/\alpha\delta) + \tilde{O}(w) \cdot \log(m/\alpha\delta) + O(\log(n/\delta))$. Let $\mathbf{y}$ be a distribution over $\{0,1\}^{O(\log(w))\cdot n}$ that is δ'-almost t'-wise independent, and let $\mathbf{z}$ be a distribution over $\{0,1\}^n$ that is δ'-almost t'-wise independent. Finally, let $\rho = \rho_{\mathbf{y},\mathbf{z}}$ be a restriction that is chosen by using a sample from $\mathbf{y}$ to determine which variables are kept alive, and an independent sample from $\mathbf{z}$ to determine values for the fixed variables.*

[7] This is because according to Theorem 3.3.11, CNFs of width w' are δ_0-fooled by any distribution that is δ''-almost t''-wise independent, where $t'' = O\left((w')^2 \cdot \log(w') + w' \cdot \log(1/\delta_0)\right) = \tilde{O}(w) \cdot \log(1/\delta) \cdot \log(m)$ and $\log(1/\delta'') = O\left((w')^2 \cdot \log^2(w') + w' \cdot \log(w') \cdot \log(1/\delta_0)\right) = \tilde{O}(w) \cdot \log(1/\delta) \cdot \log(m)$.

Then, for any depth-2 formula $F : \{0,1\}^n \to \{0,1\}$ of width $w = w(n)$ with $m = m(n)$ clauses, with probability at least $1 - 4\delta$ (where $\delta = \delta(n)$) over choice of ρ it holds that:

1. *There exists a lower-sandwiching refinement $F^{\mathtt{low}}$ of F such that $F^{\mathtt{low}}\restriction_\rho$ and $F\restriction_\rho$ are α-close (i.e., $\Pr_{x \in \mathfrak{C}(\rho)}[F^{\mathtt{low}}(x) = F(x)] \geq 1 - \alpha$) and such that the restricted refinement $F^{\mathtt{low}}\restriction_\rho$ can be computed by a decision tree of depth $D = O(\log(1/\delta))$.*
2. *There exists an upper-sandwiching refinement $F^{\mathtt{up}}$ of F such that $F^{\mathtt{up}}\restriction_\rho$ and $F\restriction_\rho$ are α-close and such that $F^{\mathtt{up}}\restriction_\rho$ can be computed by a decision tree of depth $D = O(\log(1/\delta))$.*
3. *The number of variables that are kept alive by ρ is at least $\Omega(n/w)$.*

In particular, a restriction ρ can be sampled using a seed of length $\tilde{O}(w^2) \cdot \log(1/\delta) \cdot \log\log(m/\alpha\delta) + \tilde{O}(w) \cdot \log(m/\alpha\delta) + O(\log(n/\delta))$.

Note that when $m = \Theta(1/\delta) = \Theta(1/\alpha) = \mathrm{poly}(n)$, the seed length in Theorem 3.3.14 is $\tilde{O}(w^2 \cdot \log(n))$. As in the overview in Section 3.3.2, our strategy in the proof of Theorem 3.3.14 will be as follows. Let $F^{\mathtt{low}}$ and $F^{\mathtt{up}}$ be the refinements of F from Theorem 3.3.10. Using the fact that $F^{\mathtt{low}}$ and $F^{\mathtt{up}}$ are of width w and of size $2^{\tilde{O}(w)\cdot\log\log(m/\alpha\delta)}$, we will rely on Proposition 3.3.12 to prove that, with high probability, both $F^{\mathtt{low}}\restriction_\rho$ and $F^{\mathtt{up}}\restriction_\rho$ simplify to depth-D decision trees. The main challenge will be to prove that with high probability it holds that $F^{\mathtt{low}}\restriction_\rho$ (resp., $F^{\mathtt{up}}\restriction_\rho$) and $F\restriction_\rho$ are α-close. The following lemma is the key one needed to establish the latter assertion, and after proving the lemma, we will use it to prove Theorem 3.3.14.

Lemma 3.3.15. *Let $m : \mathbb{N} \to \mathbb{N}$, let $w : \mathbb{N} \to \mathbb{N}$ such that $w(n) \leq O(\log(m(n)))$, and let $\delta : \mathbb{N} \to [0,1)$. Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula of size $m = m(n)$ and width $w = w(n)$. For $\alpha > 0$ and $\beta \leq \frac{\alpha^6 \cdot (\delta/4)^4}{m^4 \cdot \log^6(1/\delta)}$, let $F' : \{0,1\}^n \to \{0,1\}$ be a β-refinement of F.*

Fix $I \subseteq [n]$, and let $\mathbf{z}$ be a distribution over $\{0,1\}^n$ that β-fools all DNFs of width w. Let $\rho = \rho_{I,\mathbf{z}} \in \{0,1,\star\}^n$ be the restriction that is obtained by fixing values to the variables indexed by $[n] \setminus I$ according to the corresponding bits of $\mathbf{z}$. Then, with probability at least $1 - \delta$ over choice of $\mathbf{z}$ it holds that $F'\restriction_\rho$ is an α-refinement of a depth-2 formula of size m and width w for $F\restriction_\rho$.

Proof. We will prove the claim assuming that F is a DNF; if F is a CNF, then we can rely on Fact 3.3.7 to deduce that the assertion of the lemma holds for F if and only if it holds for the DNF $\neg F$. Also note that by Claim 3.3.9, for any $\rho \in \{0,1,\star\}^n$ it holds that $F'\restriction_\rho$ is a refinement of a depth-2 formula of size m and width w for $F\restriction_\rho$. Thus, we only need to prove that with probability at least $1 - \delta$ it holds that $F'\restriction_\rho$ is α-close to $F\restriction_\rho$. Recall that $I \subseteq [n]$ is fixed throughout the proof; for brevity of notation, for any $z \in \{0,1\}^n$ denote $\rho_z = \rho_{I,z}$.

In high-level, the proof follows the overview that was presented in Section 3.3.2, and in particular relies on Corollary 3.2.3. We first define a set E of excellent restrictions, which are restrictions ρ such that $F'\restriction_\rho$ is $\sqrt{\beta}$-close to $F\restriction_\rho$, and show that almost all restrictions are excellent. We will then define a set B of bad restrictions, which are

restrictions ρ such that $F'|_\rho$ is not α-close to $F|_\rho$. After defining E and B we will define the distribution $\mathbf{T}$ over tests that accepts, with high probability, every restriction in E, and rejects, with high probability, every restriction in B. Then, we will show that the residual tests $T \in \mathbf{T}$ are relatively "simple", in the sense that they can be computed by depth-3 circuits with a specific structure (i.e., top AND gate and bottom fan-in w). And finally, we will show a hitting-set generator for the set of tests in the support of $\mathbf{T}$ that accept almost all of their input restrictions, and conclude the argument using Corollary 3.2.3.

Excellent restrictions and bad restrictions. For any $\rho \in \{0,1,\star\}^n$, let $\mathtt{err}(\rho) = \Pr_{x \in \mathfrak{C}(\rho)}[F'(x) \neq F(x)]$ be the fraction of inputs in $\mathfrak{C}(\rho)$ on which F and F' disagree. Our goal is to show that $\Pr_{z \sim \mathbf{z}}[\mathtt{err}(\rho_z) \leq \alpha] \geq 1 - \delta$. Consider the following two sets:

Definition 3.3.15.1 (excellent and bad restrictions)**.** *Let* $E = \{z \in \{0,1\}^n : \mathtt{err}(\rho_z) \leq \sqrt{\beta}\}$ *be the set of* excellent *choices of restrictions, and let* $B = \{z \in \{0,1\}^n : \mathtt{err}(\rho_z) > \alpha\}$ *be the set of* bad *choices of restrictions.*

Since F' is β-close to F, a random restriction $\rho_{I,\mathbf{u}_n}$ is excellent with probability at least $1 - \sqrt{\beta}$. [8] We want to show that a pseudorandom restriction $\rho_{\mathbf{z}} = \rho_{I,\mathbf{z}}$ is not bad, with probability at least $1 - \delta$.

A distribution over simple tests. Let $t = O(\log(1/\delta)/\alpha)$. We now define a distribution $\mathbf{T}$ over tests $\{0,1\}^n \to \{0,1\}$, such that the random variable $\mathbf{T}(z)$ will essentially be the result of the following random test: Given $z \in \{0,1\}^n$, the test uniformly samples t inputs in $\mathfrak{C}(\rho_z)$, and accepts z if and only if F and F' agree on all the t inputs.

For $x \in \{0,1\}^{|I|}$ and $z \in \{0,1\}^n$, denote by $x|_z \in \mathfrak{C}(\rho_z)$ the string that is obtained by fixing the variables indexed by I according to x, and the rest of the variables (i.e., the ones indexed by $[n] \setminus I$) according to the corresponding bits from z. For any $x \in \{0,1\}^{|I|}$, let $T_x : \{0,1\}^n \to \{0,1\}$ be the function such that $T_x(z) = 1$ if and only if $F'(x|_z) = F(x|_z)$. Also, for $\bar{x} = x^{(1)}, \ldots, x^{(t)} \in \{0,1\}^{t \cdot |I|}$, let $T_{\bar{x}}$ be the function $T_{\bar{x}}(z) = \wedge_{i=1}^{t} T_{x^{(i)}}(z)$. Finally, let $\mathbf{T}$ be the distribution over tests that is obtained by uniformly choosing $\bar{x} \in \{0,1\}^{t \cdot |I|}$ and outputing $T_{\bar{x}}$. Note that $\mathbf{T}(z)$ is indeed the result of uniformly sampling t inputs in $\mathfrak{C}(\rho_z)$, and accepting z if and only if F' and F agree on all the t sampled inputs.

By our choice of the parameter t, and since β is sufficiently small, the distribution $\mathbf{T}$ indeed distinguishes between E and B:

Fact 3.3.15.2. *For any* $z \in E$ *it holds that* $\Pr_{T \sim \mathbf{T}}[T(z) = 1] \geq (1 - \sqrt{\beta})^t \geq 1 - t \cdot \sqrt{\beta}$*, and for any* $z \in B$ *it holds that* $\Pr_{T \sim \mathbf{T}}[T(z) = 1] < (1 - \alpha)^t < \delta/3$.

For $\eta = \sqrt{t+1} \cdot \beta^{1/4}$, let $\mathbf{T}'$ be the set of tests $T_{\bar{x}} \in \mathbf{T}$ that accept at least $1 - \eta$ of their inputs (i.e., $\mathbf{T}' = \{T_{\bar{x}} : \Pr_{z \in \{0,1\}^n}[T_{\bar{x}}(z) = 1] \geq 1 - \eta\}$). We will abuse the

[8] Because $\mathbb{E}[\mathtt{err}(\rho_{I,\mathbf{u}_n})] = \Pr_{x \in \{0,1\}^n}[F'(x) \neq F(x)] \leq \beta$, which implies that $\Pr[\mathtt{err}(\rho_{I,\mathbf{u}_n}) > \sqrt{\beta}] < \sqrt{\beta}$.

notations $\mathbf{T}$ and $\mathbf{T}'$, by using them both to denote sets and to denote the uniform distribution over the corresponding set. To see that the set $\mathbf{T}'$ is dense in $\mathbf{T}$, note that

$$\mathbb{E}_{T_{\bar{x}}\in\mathbf{T}}\left[\Pr_{z\in\{0,1\}^n}[T_{\bar{x}}(z)=1]\right] = \mathbb{E}_{z\in\{0,1\}^n}\left[\Pr_{T_{\bar{x}}\in\mathbf{T}}[T_{\bar{x}}(z)=1]\right]$$
$$\geq \Pr_{z\in\{0,1\}^n}[z\in E]\cdot\min_{z\in E}\left\{\Pr_{T_{\bar{x}}\in\mathbf{T}}[T_{\bar{x}}(z)=1]\right\},$$

which is at least $1-\sqrt{\beta}-t\cdot\sqrt{\beta}=1-\eta^2$. Therefore, the probability over $T_{\bar{x}}\in\mathbf{T}$ that $T_{\bar{x}}$ rejects more than η of its input restrictions is at most η.

A hitting-set generator for $\mathbf{T}'$. Towards designing a hitting-set generator with high density for every $T_{\bar{x}}\in\mathbf{T}'$, we first show that each $T_{\bar{x}}\in\mathbf{T}$ can be computed by a depth-3 circuit with a top AND gate and small bottom fan-in. To do so, we first show that for a single $x\in\{0,1\}^{|I|}$ (rather than for $\bar{x}=x^{(1)},...,x^{(t)}$) it holds that T_x can be computed by a depth-3 circuit with a top AND gate and small bottom fan-in.

Claim 3.3.15.3. *For every fixed $x\in\{0,1\}^{|I|}$, the function $T_x:\{0,1\}^n\to\{0,1\}$ can be computed by a depth-3 circuit with a top AND gate of fan-in at most m such that the bottom fan-in of the circuit is at most w.*

Proof. Denote the number of refinement steps that were applied to F to obtain F' by $k\leq m$. For any $i\in[k]$, let $F^{(i)}$ be the formula in the beginning of the i^{th} refinement step in the transformation of F to F', and let $F^{(k+1)}=F'$. Note that $T_x(z)=1$ if and only if for every $i\in[k]$ it holds that $F^{(i)}(x\restriction_z)=F^{(i+1)}(x\restriction_z)$ (one direction is immediate, whereas the other direction follows by the monotonicity of the sequence $F^{(1)}(x\restriction_z),...,F^{(k+1)}(x\restriction_z)$ [9]).

For every $i\in[k]$, let $T_{x,i}$ be the function such that $T_{x,i}(z)=1$ if and only if $F^{(i)}(x\restriction_z)=F^{(i+1)}(x\restriction_z)$. We will show that each $T_{x,i}$ can be computed by a DNF of width w. This claim suffices to conclude the proof, since it implies that T_x can be computed by a circuit with a top AND gate that is connected to $k\leq m$ DNFs of width w. To prove the claim, fix $i\in[k]$, and let us conduct a case analysis:

- If the i^{th} refinement step was a clean-up step, then $T_{x,i}\equiv 1$.
- If the i^{th} step was a removal step, then let $f^{(i)}$ be the clause that was removed from $F^{(i)}$ in the i^{th} step, and let $F^{(i+1)}=\left(F^{(i)}\setminus f^{(i)}\right)$ be the formula that is obtained by dropping the clause $f^{(i)}$ from $F^{(i)}$. Note that $F^{(i+1)}(x\restriction_z)=F^{(i)}(x\restriction_z)$ if and only if either $f^{(i)}(x\restriction_z)=0$ or $\left(F^{(i)}\setminus f^{(i)}\right)(x\restriction_z)=1$. The latter event is a disjunction of at most m events (because $\left(F^{(i)}\setminus f^{(i)}\right)$ is a DNF of size at most $m-1$), each of which depends on the values of at most w bits in $x\restriction_z$. Thus, each

[9] If F' was obtained by merging steps and clean-up steps, then $F^{(1)}(x\restriction_z)\leq...\leq F^{(k+1)}(x\restriction_z)$, whereas if F' was obtained by removal steps and clean-up steps, then $F^{(1)}(x\restriction_z)\geq...\geq F^{(k+1)}(x\restriction_z)$.

of the (at most m) events depends on at most w bits in z, and can therefore be decided by a DNF of width w. It follows that $T_{x,i}$ is the disjunction of width-w DNFs, which is a width-w DNF.

- If the i^{th} refinement step in the transformation of F to F' was a merging step, denote the $u \geq 2$ clauses that were removed from $F^{(i)}$ in the step by $f_1^{(i)}, ..., f_u^{(i)}$, and the new clause that was added in their stead by $h^{(i)}$. Note that $F^{(i+1)}(x|_z) = F^{(i)}(x|_z)$ if and only if either $h^{(i)}(x|_z) = 0$ or $F^{(i)}(x|_z) = 1$. This is a disjunction of at most $m+1$ events, each of which depends on at most w bits in $x|_z$ (and thus on at most w bits in z). Thus, in this case too it holds that $T_{x,i}$ can be computed by a DNF of width w. □

For a fixed $\bar{x} = x^{(1)}, ..., x^{(t)} \in \{0,1\}^{t \cdot |I|}$, we can compute $T_{\bar{x}}$ by taking a conjunction of t circuits for the corresponding T_x's (i.e., $\wedge_{i \in [t]} T_{x^{(i)}}$), which is a depth-3 circuit with bottom fan-in at most w and top fan-in at most $t \cdot m$. We are now ready to prove that $\mathbf{z}$ is a hitting-set generator with density $1 - \delta/3$ for every $T_{\bar{x}} \in \mathbf{T}'$:

Claim 3.3.15.4. *For every $T_{\bar{x}} \in \mathbf{T}'$ it holds that $\Pr[T(\mathbf{z}) = 1] \geq 1 - \delta/3$.*

Proof. Fix $T_{\bar{x}} \in \mathbf{T}'$, and recall that by the definition of $\mathbf{T}'$ it holds that $T_{\bar{x}}$ accepts at least $1 - \eta$ of its inputs. Thus, each of the DNFs in the middle layer of the circuit that we constructed for $T_{\bar{x}}$ accepts $1 - \eta$ of the inputs. It follows that when using the distribution $\mathbf{z}$, which is β-pseudorandom for such DNFs, each of these DNFs accepts with probability at least $1 - \eta - \beta$. By a union-bound, it follows that

$$\begin{aligned}
\Pr_{z \sim \mathbf{z}}[T_{\bar{x}}(z) = 1] &\geq 1 - (\eta + \beta) \cdot (t \cdot m) \\
&> 1 - (2 \cdot t \cdot m) \cdot \eta \\
&= 1 - O\left((\log(1/\delta)/\alpha)^{3/2} \cdot m \cdot \beta^{1/4} \right) ,
\end{aligned}$$

which is larger than $1 - \delta/3$ by the hypothesis that β is sufficiently small. □

Invoking Corollary 3.2.3. We now conclude the argument by invoking Corollary 3.2.3. Let E be as in Definition 3.3.15.1, and let $G = \{0,1\}^n \setminus B$; recall that for $\epsilon_1 = \sqrt{\beta}$ it holds that $\Pr_{z \in \{0,1\}^n}[z \in E] \geq 1 - \epsilon_1$. Denoting $\epsilon_2 = t \cdot \sqrt{\beta}$ and $\epsilon_3 = \delta/3$, according to Fact 3.3.15.2, for any $z \in E$ it holds that $\Pr_{T_{\bar{x}} \sim \mathbf{T}}[T_{\bar{x}}(z) = 1] \geq 1 - \epsilon_2$ and for any $z \notin G$ it holds that $\Pr_{T_{\bar{x}} \sim \mathbf{T}}[T_{\bar{x}}(z) = 0] \geq 1 - \epsilon_3$.

Finally, for $\epsilon_4 = \eta$ it holds that the set $\mathbf{T}'$ is of density at least $1 - \epsilon_4$ in $\mathbf{T}$, and for every $T_{\bar{x}} \in \mathbf{T}'$, by Claim 3.3.15.4 it holds that $\mathbf{z}$ fools $T_{\bar{x}}$ with error at most $\epsilon_5 = \delta/3$ (because $\Pr_{z \in \{0,1\}^n}[T_{\bar{x}}(z) = 1] \geq 1 - \eta \geq 1 - \delta/3$ and $\Pr_{z \sim \mathbf{z}}[T_{\bar{x}}(z) = 1] \geq 1 - \delta/3$). Relying on Corollary 3.2.3, the probability that $\mathbf{z} \notin G$ is at most

$$\sqrt{\beta} + t \cdot \sqrt{\beta} + \delta/3 + 2 \cdot \eta + \delta/3 = 2\delta/3 + \eta^2 + 2 \cdot \eta < \delta ,$$

where the inequality relied on the fact that β (and hence also η) is sufficiently small. ■

We are now ready to prove Theorem 3.3.14.

Proof of Theorem 3.3.14. Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula of width w and size m. Let $F^{\mathtt{low}} : \{0,1\}^n \to \{0,1\}$ and $F^{\mathtt{up}} : \{0,1\}^n \to \{0,1\}$ be the β-lower-sandwiching and the β-upper-sandwiching formulas for F from Theorem 3.3.10, respectively, where $\beta = \frac{\alpha^6 \cdot (\delta/4)^4}{m^4 \cdot \log^6(1/\delta)}$. Note that the width of $F^{\mathtt{low}}$ and of $F^{\mathtt{up}}$ is at most w, and that their size is at most $2^{\tilde{O}(w) \cdot \log\log(m/\alpha\delta)}$.

According to Fact 2.1.3, the distribution of strings $\mathbf{r}$ over $\{0,1\}^{O(\log(w)) \cdot n}$, which is obtained by combining $\mathbf{y}$ and $\mathbf{z}$ and represents the pseudorandom restriction $\rho = \rho_{\mathbf{y},\mathbf{z}}$, is $(2 \cdot \delta')$-almost t'-wise independent. Hence, relying on Proposition 3.3.12, with probability at least $1 - 2\delta$ it holds both that $F^{\mathtt{low}}\restriction_\rho$ and $F^{\mathtt{up}}\restriction_\rho$ can be computed by decision trees of depth D, and that ρ keeps at least $\Omega(n/w)$ variables alive.

According to Theorem 3.3.11, all DNFs of width w are β-fooled by the distribution $\mathbf{z}$. [10] Therefore, relying on Lemma 3.3.15, for any fixed choice of $y \sim \mathbf{y}$, with probability at least $1 - 2\delta$ over $z \sim \mathbf{z}$ it holds that both $F^{\mathtt{low}}\restriction_\rho$ and $F^{\mathtt{up}}\restriction_\rho$ are α-close to $F\restriction_\rho$. Thus, the probability over choice of both $\mathbf{y}$ and $\mathbf{z}$ that $F^{\mathtt{low}}\restriction_\rho$ and $F^{\mathtt{up}}\restriction_\rho$ are α-close to $F\restriction_\rho$ is at least $1 - 2\delta$. ■

3.3.4.3 Proofs of Theorems 3.3.2 and 3.3.3

We are now ready to prove Theorem 3.3.3. Recall that Theorem 3.3.3 asserts the existence of a hitting-set generator that is parametrized by a parameter $t > 0$.

Theorem 3.3.16 (Theorem 3.3.3, restated)**.** *Let $d \geq 2$, let $m : \mathbb{N} \to \mathbb{N}$ such that $m(n) \leq \mathrm{poly}(n)$, and let $t : \mathbb{N} \to \mathbb{N}$ such that $c_0 \leq t(n) \leq 2 \cdot \log(m(n))$, where c_0 is a sufficiently large constant. For every $n \in \mathbb{N}$, let $\mathcal{C}_n$ be the class of circuits $C : \{0,1\}^n \to \{0,1\}$ of size $m = m(n)$ and of depth at most d that accept all but at most $B(n)$ of their inputs, where $\log(B(n)) = \Omega\left(n^{1-1/\Omega(t)}/t^{d-2}\right)$ and $t = t(n)$. Then, there exists a hitting-set generator for $\mathcal{C} = \cup_{n\in\mathbb{N}}\mathcal{C}_n$ with seed length $\ell = \ell(n) = \tilde{O}\left(t^2 \cdot \log(n)\right)$.*

Theorem 3.3.2 follows as a corollary of Theorem 3.3.16, by using the specific parameter value $t = 2 \cdot \log(m)$, in which case $B(n) = 2^{\Omega\left(n/\log^{d-2}(n)\right)}$ and the seed length is $\tilde{O}\left(\log^3(n)\right)$.

Proof. Given input 1^n and a random seed in $\{0,1\}^\ell$, the hitting-set generator works in two steps. In the first step, the generator outputs a restriction $\bar{\rho} \in \{0,1,\star\}^n$ such that for any circuit C over n input bits of depth d and size $m = m(n)$, with high probability it holds that there exists a depth-2 formula C' of size $\mathrm{poly}(n)$ and width t that is both $(1/2)$-close to $C\restriction_{\bar{\rho}}$ and lower-sandwiching for $C\restriction_{\bar{\rho}}$. Moreover, with high probability the restriction $\bar{\rho}$ keeps at least $\log(B(n)) + 2$ variables alive.

[10] Theorem 3.3.11 requires that the distribution $\mathbf{z}$ will be δ''-almost t''-wise independent, where $t'' = O(w^2 \cdot \log(w) + w \cdot \log(1/\beta)) = \tilde{O}(w) \cdot \log(m/\alpha\delta) < t'$ and $\log(1/\delta'') = O(w^2 \cdot \log^2(w) + w \cdot \log(w) \cdot \log(1/\beta)) = \tilde{O}(w) \cdot \log(m/\alpha\delta) < \log(1/\delta')$.

Since the subcube $\mathfrak{C}(\bar{\rho})$ contains at least $4 \cdot B(n)$ inputs, the acceptance probability of $C|_{\bar{\rho}}$ is at least $3/4$. Hence, the acceptance probability of C' is at least $1/4$ (because C' is $(1/2)$-close to $C|_{\bar{\rho}}$), and every satisfying input for C' is also satisfying for C (because C' is lower-sandwiching for $C|_{\bar{\rho}}$). Thus, in the `second step`, we use a pseudorandom generator for depth-2 circuits to "fool" C': The pseudorandom generator outputs a satisfying input for C' in $\mathfrak{C}(\bar{\rho})$ with positive probability, and any such input yields a satisfying input for C.

Parameter settings. Let $\epsilon > 0$ be a sufficiently small constant, and let $\delta = (\epsilon/m)$. Let $D = O(\log(1/\delta)) > 2 \cdot \log(2m/\delta)$, and let $m' = m \cdot 2^D = \text{poly}(n)$. Let $\beta = \left(\frac{\delta}{2dm}\right)^{10^{2d}}$; we will use β as the approximation parameter whenever using Theorem 3.3.10. Let $\delta' > 0$ and $t' \in \mathbb{N}$ such that $\log(1/\delta') = O(t') = \tilde{O}\left(t^2 \cdot \log(n)\right)$.

The pseudorandom choice of restrictions. The algorithm that we will describe below constructs a sequence of restrictions. We mention in advance that when describing the algorithm, whenever we will say that we choose a restriction with a parameter $p = 2^{-q}$, the pseudorandom choice of restriction is the following:

- Let $\mathbf{y}$ be a distribution over $\{0,1\}^{\log(1/p)\cdot n}$ that is δ'-almost t'-wise independent.
- Let $\mathbf{z}$ be a distribution over $\{0,1\}^n$ that is δ'-almost t'-wise independent.
- The restriction $\rho = \rho_{\mathbf{y},\mathbf{z}}$ is chosen by sampling $y \sim \mathbf{y}$ in order to determine which variables are kept alive, and independently sampling $z \sim \mathbf{z}$ in order to determine values for the fixed variables.

Note that such a restriction keeps every variable alive with probability approximately p (i.e., with probability $p \pm \delta'$). The above process yields a distribution $\mathbf{r}$ over $\{0,1\}^{(\log(1/p)+1)\cdot n}$, which is obtained by combining $\mathbf{y}$ and $\mathbf{z}$ as detailed in the beginning of Section 3.3.4.1; according to Fact 2.1.3, the distribution $\mathbf{r}$ is $(2 \cdot \delta')$-almost t'-wise independent.

The first step. The generator constructs the restriction $\bar{\rho}$ as the composition of $2d-2$ retrictions $\bar{\rho} = \rho^{(2d-3)} \circ \rho^{(2d-4)} \circ ... \circ \rho^{(1)} \circ \rho^{(0)}$. The initial restriction $\rho^{(0)}$ is chosen with parameter $p = 1/O(1)$, and with probability $1-\epsilon$ it reduces the bottom fan-in of the circuit to $D = O(\log(1/\delta))$. [11] The next $2 \cdot (d-2)$ restrictions are applied in $d-2$ iterations. Loosely speaking, in each iteration, we apply a restriction that reduces the bottom fan-in to t, then define an approximating circuit (by replacing the formulas

[11] To see that such a restriction indeed reduces the bottom fan-in, fix a gate in the bottom layer of fan-in more than $2 \cdot \log(2m/\epsilon)$. The probability under a uniformly-chosen restriction with $p = 1/4$ that none of the lexicographically-first $2 \cdot \log(2m/\epsilon)$ variables feeding into the gate is fixed to a satisfying value is $\left(\frac{1+p}{2}\right)^{2\cdot\log(2m/\epsilon)} < \epsilon/2m$. Since this event depends only on the values that the restriction assigns to $2 \cdot \log(2m/\epsilon)$ variables, and the value for each variable depends on $\log(1/p) = O(1)$ bits, the event depends on at most $O(\log(m/\epsilon))$ bits of the restriction. Thus, the event happens with probability at most ϵ/m when the restriction is chosen from a $1/\text{poly}(m/\epsilon)$-biased set.

in the next-to-bottom layer, which have small width at this point, with small lower-sandwiching refinements, using Theorem 3.3.10), and finally apply a second restriction in order to "switch" the formulas in the next-to-bottom layer of the approximating circuit, and reduce the depth of the circuit.

Let $C^{(0)} = C\restriction_{\rho^{(0)}}$ be the circuit in the beginning of the first iteration, and note that $C^{(0)}$ is of depth d, size at most $m < m'$, and bottom fan-in at most D. For $i \in [d-2]$, let us describe the i^{th} iteration. Assuming all previous iterations were successful, in the beginning of the i^{th} iteration we start with a circuit $C^{(i-1)}$ of depth at most $d-(i-1)$, bottom fan-in at most D, and with at most $m' = m \cdot 2^D$ gates in its bottom layer. We will produce two restrictions, denoted $\rho^{(2i-1)}$ and $\rho^{(2i)}$, and define a circuit $C^{(i)}$ whose domain is $\mathfrak{C}(\rho^{(2i)} \circ \rho^{(2i-1)} \circ ... \circ \rho^{(0)})$ such that with probability $1 - O(\epsilon)$ it holds that $C^{(i)}$ is of depth at most $d-i$, bottom fan-in D, and the number of gates in its bottom layer is at most m'. (After we finish the description of a single iteration, we will also prove that for any $i \in [d-2]$ it holds that $C^{(i)}\restriction_{\bar{\rho}}$ is close to $C^{(i-1)}\restriction_{\bar{\rho}}$; see Claim 3.3.16.2 below.)

The first restriction in iteration i, denoted $\rho^{(2i-1)}$, is chosen with the parameter $p = \left(\epsilon / \left(m \cdot 2^{2D+1}\right)\right)^{1/t} = n^{-1/\Omega(t)}$. We now show that with probability at least $1 - O(\epsilon)$ the bottom fan-in of the circuit $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$ is less than t. To do so, first note the following:

Claim 3.3.16.1. *Let S be a fixed set of at most D variables. Then, with probability at least $1 - \epsilon/m'$ it holds that less than t variables in S are kept alive by $\rho^{(2i-1)}$.*

Proof. Recall that the restriction $\rho^{(2i-1)}$ is chosen such that the distribution $\mathbf{y}$ over $\{0,1\}^{\log(1/p)\cdot n}$, which determines which variables will be kept alive, is δ'-almost t'-wise independent. We will only need the fact that the blocks of size $\lceil \log(1/p) \rceil$ in $\mathbf{y}$ are (p^t)-almost t-wise independent; this holds because $t \cdot \lceil \log(1/p) \rceil < O(\log(m/\epsilon)) < t'$, and $\delta' < p^t = 1/\mathrm{poly}(n)$.

For any fixed set of t variables in S, the probability that all variables in the set remain alive after applying a uniformly-chosen restriction with the parameter p is p^t. Since the blocks of size $\lceil \log(1/p) \rceil$ in $\mathbf{y}$ are (p^t)-almost t-wise independent, the probability that $\rho^{(2i-1)}$ keeps all t variables alive is at most $2 \cdot (p^t)$. Thus, the probability that $\rho^{(2i-1)}$ keeps t variables in S alive is at most $\binom{|S|}{t} \cdot 2 \cdot p^t < 2^{D+1} \cdot p^t < \epsilon/m'$. □

Recall that the number of gates in the bottom layer of $C^{(i-1)}$ is at most m', and that each of them is of fan-in at most D. Using Claim 3.3.16.1 and a union-bound, with probability at least $1 - \epsilon$ it holds that the bottom fan-in of $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$ is less than t.

Assuming that the bottom fan-in of $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$ is indeed less than t, we now use Theorem 3.3.10 to replace each formula F in the next-to-bottom layer of $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$ with a β-lower-sandwiching refinement $F^{\mathtt{low}}$ such that the size of $F^{\mathtt{low}}$ is at most $2^{\tilde{O}(t)\cdot \log\log(1/\beta)}$. Let $\widetilde{C^{(i-1)}}\restriction_{\rho^{(2i-1)}}$ be the circuit that is obtained by replacing all the formulas in the next-to-bottom layer of $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$ in this manner.

The final step in the i^{th} iteration is to apply a restriction $\rho^{(2i)}$ with parameter $p = 1/O(t)$ that is intended to simplify each formula $F^{\mathtt{low}}$ in the next-to-bottom layer of $\widetilde{C^{(i-1)}\restriction_{\rho^{(2i-1)}}}$ to a decision tree of depth at most D. Let $C^{(i)} = \left(\widetilde{C^{(i-1)}\restriction_{\rho^{(2i-1)}}}\right)\restriction_{\rho^{(2i)}}$. Relying on Proposition 3.3.12, the restriction $\rho^{(2i)}$ is successful with probability at least $1 - O(\epsilon)$, and in this case the circuit $C^{(i)}$ is of depth at most $d - i$, and the bottom layer of $C^{(i)}$ has at most $m' = m \cdot 2^D$ gates, each of fan-in at most D. [12]

We now apply one final restriction $\rho^{(2d-3)}$, with parameter $p = \left(\epsilon / \left(m \cdot 2^{2D+1}\right)\right)^{1/t}$, in order to reduce the bottom fan-in of $C^{(d-2)}$ to t. Using Claim 3.3.16.1 and a union-bound, with probability at least $1 - O(\epsilon)$ it holds that the width of $C^{(d-2)}\restriction_{\rho^{(2d-3)}}$ is at most t. For convenience, in Table 3.1 we summarize the restrictions that were applied in the first step.

	Value of p	**Goal of the restriction**
$\rho^{(0)}$	$1/O(1)$	Reduce the bottom fan-in to D
$i = 1, ..., d-2$:		
$\rho^{(2i-1)}$	$n^{-1/\Omega(t)}$	Reduce the bottom fan-in to t
$\rho^{(2i)}$	$1/O(t)$	"Switch" the width-t formulas at the next-to-bottom-layer
$\rho^{(2d-3)}$	$n^{-1/\Omega(t)}$	Reduce the bottom fan-in to t

Table 3.1: Summary of the restrictions that are applied in the first step.

Let $C^{(d-1)} = C^{(d-2)}\restriction_{\rho^{(2d-3)}}$, and recall that $\bar{\rho} = \rho^{(2d-3)} \circ \rho^{(2d-2)} \circ ... \circ \rho^{(0)}$. The above shows that if all the iterations are successful, then $C^{(d-1)}$ is a formula of depth 2, size at most m', and width t. Also note that if all the iterations are successful, then $C^{(d-1)}$ is lower-sandwiching for $C\restriction_{\bar{\rho}}$. This is because for every $i \in [d-2]$ it holds that $\widetilde{C^{(i-1)}\restriction_{\rho^{(2i-1)}}}$ is lower-sandwiching for $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$ (since $\widetilde{C^{(i-1)}\restriction_{\rho^{(2i-1)}}}$ is obtained by replacing every formula F in the next-to-bottom-layer of $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$ with a lower-sandwiching refinement $F^{\mathtt{low}}$), which implies that $C^{(i)}\restriction_{\bar{\rho}} = \left(\widetilde{C^{(i-1)}\restriction_{\rho^{(2i-1)}}}\right)\restriction_{\bar{\rho}}$ is lower-sandwiching for $C^{(i-1)}\restriction_{\bar{\rho}}$.

The main thing that is left to prove in the analysis of the first step is that with probability at least $1 - O(\epsilon)$ it holds that $C^{(d-1)}$ is $(1/2)$-close $C\restriction_{\bar{\rho}}$. To do so, we will

[12] Specifically, we rely on Proposition 3.3.12 with width parameter t, error parameter δ, size parameter $2^{\tilde{O}(t)\cdot\log\log(1/\beta)}$, and depth bound D for the decision trees. Proposition 3.3.12 requires that the distribution $\mathbf{r}$ of restrictions will be δ''-almost t''-wise independent, where $\log(1/\delta'') = O(t'') = \tilde{O}(t^2) \cdot \log(1/\delta) \cdot \log\log(1/\beta) = \tilde{O}(t^2 \cdot \log(n))$. The latter holds by our choice of δ' and t'.

show that with probability at least $1-O(\epsilon)$, for every $i \in [d-2]$ it holds that $C^{(i-1)}\restriction_{\bar{\rho}}$ is $(1/2d)$-close to $C^{(i)}\restriction_{\bar{\rho}}$. Assuming that the latter holds, we can deduce that $C\restriction_{\bar{\rho}} = C^{(0)}\restriction_{\bar{\rho}}$ is $1/2$-close to $C^{(d-1)} = C^{(d-2)}\restriction_{\bar{\rho}}$. Thus, it suffices to prove the following claim:

Claim 3.3.16.2. *For any $i \in [d-2]$, with probability at least $1-O(\epsilon)$ it holds that $C^{(i)}\restriction_{\bar{\rho}}$ is $(1/2d)$-close to $C^{(i-1)}\restriction_{\bar{\rho}}$.*

Proof. Let $i \in [d-2]$, let F be a formula in the next-to-bottom layer of $C^{(i-1)}\restriction_{\rho^{(2i-1)}}$, and let $F^{\mathtt{low}}$ be a β-refinement of F. We will prove that with probability $1-O(\delta)$ it holds that $F^{\mathtt{low}}\restriction_{\bar{\rho}}$ is $(1/2dm)$-close to $F\restriction_{\bar{\rho}}$. This suffices to prove Claim 3.3.16.2, since by a union-bound over m formulas it follows that with probability at least $1-O(\epsilon)$ it holds that the circuit $\left(\widetilde{C^{(i-1)}\restriction_{\rho^{(2i-1)}}}\right)\restriction_{\bar{\rho}} = C^{(i)}\restriction_{\bar{\rho}}$ is $(1/2d)$-close to $\left(C^{(i-1)}\restriction_{\rho^{(2i-1)}}\right)\restriction_{\bar{\rho}} = C^{(i-1)}\restriction_{\bar{\rho}}$.

For every $j \in \{2i, ..., 2d-3\}$, let $\rho^{(2i,...,j)}$ be the composed restriction $\rho^{(2i,...,j)} = \rho^{(j)} \circ ... \circ \rho^{(2i)}$, and let $\beta_j = (\delta/2dm)^{10^{2d-3-j}}$. We will prove the following statement: For every $j \in \{2i, ..., 2d-3\}$, with probability at least $1-O(\delta)$ it holds that $F^{\mathtt{low}}\restriction_{\rho^{(2i,...,j)}}$ is a β_j-refinement of a depth-2 formula of size m' and width t for $F\restriction_{\rho^{(2i,...,j)}}$. Invoking this statement with $j = 2d-3$, we can deduce that with probability at least $1-O(\delta)$ it holds that $F^{\mathtt{low}}\restriction_{\bar{\rho}}$ is β_{2d-3}-close to $F\restriction_{\bar{\rho}}$, where $\beta_{2d-3} < 1/2dm$.

We prove the aforementioned statement by induction on j. For the base case $j = 2i$, we start with a formula F of size m' and width t, and a β-refinement $F^{\mathtt{low}}$ of F, where $\beta < \beta_0 \leq \beta_{j-1}$. Now, $\rho^{(j)}$ is chosen according to a distribution such that for every fixed choice of variables to keep alive (i.e., every fixed $y \sim \mathbf{y}$), the choice of values for the fixed variables (i.e., $z \sim \mathbf{z}$) is δ'-almost t'-wise independent. Relying on Theorem 3.3.11 and on our choice of δ' and t', the distribution distribution $\mathbf{z}$ β-fools all DNFs of width w. We can therefore rely on Lemma 3.3.15 to deduce that with probability at least $1-O(\delta)$ it holds that $F^{\mathtt{low}}\restriction_{\rho^{(j)}}$ is a β_j-refinement of $F\restriction_{\rho^{(j)}}$. [13]

The induction step, for $j \geq 2i+1$, is very similar to the base case. By the induction hypothesis, with probability at least $1-O(\delta)$ it holds that $F^{\mathtt{low}}\restriction_{\rho^{(2i,...,j-1)}}$ is a (β_{j-1})-refinement of a size m' and width w' depth-2 formula for $F\restriction_{\rho^{(2i,...,j-1)}}$. We can then use Theorem 3.3.11 and Lemma 3.3.15 similarly to the base case. □

To conclude the analysis of the first step, note that with probability at least $1-O(\epsilon)$ it holds that at least $\log(B(n))+2 = \Omega\left(n^{1-1/\Omega(t)}/t^{d-2}\right)$ variables remain alive. To see that this is the case, recall that $\bar{\rho}$ is comprised of one restriction with parameter $p_0 = 1/O(1)$, and $d-1$ restrictions with parameter $p_1 = n^{-1/\Omega(t)}$, and $d-2$

[13] We invoke Lemma 3.3.15 with width parameter t, size bound m', and error parameter δ. We know that $F^{\mathtt{low}}$ is a β_{j-1}-refinement of F, and we want to deduce that with probability at least $1-O(\delta)$ it holds that $F^{\mathtt{low}}\restriction_{\rho^{(j)}}$ is an α-refinement of $F\restriction_{\rho^{(j)}}$, where $\alpha = \beta^j$. The lemma requires that the distribution $\mathbf{z}$ will (β_{j-1})-fool all DNFs of width t, and that $\beta_{j-1} \leq \frac{\beta_j^6 \cdot (\delta/4)^4}{m^4 \cdot \log^6(1/\delta)}$, both of which indeed hold.

restrictions with parameter $p_2 = 1/O(t)$. Let $\bar{p} = p_0 \cdot p_1^{d-1} \cdot p_2^{d-2} \cdot n$, and note that $\bar{p} = \Omega\left(n^{1-1/\Omega(t)}/t^{d-2}\right)$.

The expected number of living variables under $\bar{\rho}$ is $\Theta(\bar{p})$ (because in each restriction with parameter p, every variable is kept alive with probability $p \pm O(\delta') \in p \pm (p/2)$). Since all the choices of variables to keep alive are according to distributions that are δ'-almost t'-wise independent, we can use Fact 2.1.2 to deduce that with probability at least $1 - O(\epsilon)$ it holds that at least $\Omega(\bar{p}) = \Omega\left(n^{1-1/\Omega(t)}/t^{d-2}\right) > \log(B(n)) + 2$ variables remain alive after the first step. (When using Fact 2.1.2, we relied on the fact that t is larger than a sufficiently large constant c_0 to deduce that $n^{1-1/\Omega(t)}/t^{d-2} > n^{\Omega(1)}$).

The second step. We now invoke the pseudorandom generator from Theorem 3.3.11 for depth-2 circuits of width t, instantiated with error parameter $1/8$, and output the string that the generator outputs, completed to a string of length n according to $\bar{\rho}$. The generator requires a seed of length $O(t^2 \cdot \log^2(t)) = \tilde{O}(t^2)$.

Let us now prove this yields a satisfying input for C, with positive probability. If the first step was successful, then $\bar{\rho}$ kept more than $\log(B(n)) + 2$ live variables, and hence the acceptance probability of $C|_{\bar{\rho}}$ is at least $3/4$. Since $C^{(d-1)}$ is $1/2$-close to $C|_{\bar{\rho}}$, it follows that $\Pr_{x \in \mathfrak{C}(\bar{\rho})}[C^{(d-1)}(x) = 1] \geq 1/4$. Thus, the generator outputs a satisfying input for $C^{(d-1)}$, with positive probability, and this input (when completed to a string of length n according to $\bar{\rho}$) is satisfying for C, because $C^{(d-1)}$ is lower-sandwiching for $C|_{\bar{\rho}}$. ■

3.3.5 Appendices for Section 3.3

3.3.5.1 Known width-dependent derandomizations of the switching lemma

We prove two claims from Section 3.3.4.2 (i.e., Lemma 3.3.13 and a generalization of the switching lemma of [GW14]). Lemma 3.3.13 is an adaptation of the main lemma of Trevisan and Xue [TX13]. Let us now recall the statement of Lemma 3.3.13, and prove the lemma.

Lemma 3.3.17. *(Lemma 3.3.13, restated). Let F be a CNF over n inputs with m clauses, each clause of width at most w. For a positive parameter $p = 2^{-q}$, where $q \in \mathbb{N}$, let $\rho \in \{0,1,\star\}^n$ be a restriction that is chosen according to a distribution over $\{0,1\}^{(q+1)\cdot n}$ that δ_0-fools all CNFs of width $w' = w \cdot (q+1)$. Then, the probability that $F|_\rho$ cannot be computed by a decision tree of depth D is at most $2^{D+w+1} \cdot (5pw)^D + \delta_0 \cdot 2^{(D+1)\cdot(2\cdot w + \log(m))}$.*

Proof sketch. We rely on the proof of Lemma 7 in [TX13], and in particular use the same definitions of canonical decision tree, path, and segment. The proof in [TX13] reduces the task of finding a restriction ρ such that $F|_\rho$ can be computed by a shallow decision tree to the task of "fooling" less than $2^{(D+1)\cdot(2w+\log(m))}$ tests: For each path of length $D+1$ (i.e., a sequence of $D+1$ segments), there is a corresponding test

$T_P : \{0,1\}^{(q+1)\cdot n} \to \{0,1\}$ that gets as input a restriction $\rho \in \{0,1\}^{(q+1)\cdot n}$, and accepts ρ if and only if the canonical decision tree for $F\restriction_\rho$ contains the path P. Indeed, if all the tests reject ρ, it means that no path of length $D+1$ exists in the canonical decision tree for $F\restriction_\rho$, which implies that the canonical decision tree for $F\restriction_\rho$ is of depth D.

The key claim in the proof is Claim 8, which asserts that for each path P, the test T_P can be computed by a CNF. The goal in [TX13] is to show that the CNF for T_P has few clauses; we focus on showing that the CNF for T_P has small width. To see that this holds, note that T_P is constructed as a conjunction of conditions, where each condition depends only on the assignment that ρ gives to the variables of a single clause of F (either a clause that belongs to a segment in the path, or a clause whose index is between the indices of clauses that belong to segments in the path). Thus, each condition depends only on the assignment that ρ gives to w variables, which means that each condition depends only on $w' = w \cdot (q+1)$ bits of ρ. Hence, each condition can be decided by a CNF of width w', and T_P (which is their conjunction) can also be decided by a CNF of width w'. ■

Let us now formally state the generalization of the switching lemma of Goldreich and Wigderson [GW14] and prove it.

Proposition 3.3.18. *(a generalization of the derandomized switching lemma of [GW14]). Let $m : \mathbb{N} \to \mathbb{N}$, let $w : \mathbb{N} \to \mathbb{N}$, and let $\delta : \mathbb{N} \to [0,1)$. Let $\mathbf{z}$ be a distribution over $\{0,1\}^{O(\log(w))\cdot n}$ that is δ'-almost t'-wise independent, where $\log(1/\delta') = O(t') = \tilde{O}(w) \cdot 2^w \cdot \log(1/\delta)$.*

Then, for any depth-2 formula $F : \{0,1\}^n \to \{0,1\}$ of width $w = w(n)$ with $m = m(n)$ clauses, with probability at least $1 - 4\delta$ (where $\delta = \delta(n)$) over choice of $\rho \sim \mathbf{z}$ it holds that the restricted formula $F\restriction_\rho$ can be computed by a decision tree of depth $D = O(\log(1/\delta))$.

Proof. Let $\delta_0 = \delta \cdot 2^{-D} = \mathrm{poly}(\delta)$, and fix a depth-2 formula $F : \{0,1\}^n \to \{0,1\}$; without loss of generality, assume that F is a CNF.[14] Consider a uniformly-chosen restriction ρ that keeps each variable alive with probability $p = 1/O(w)$; Hastad's switching lemma asserts that with probability at least $1 - 2^{-O(D)} \geq 1 - \delta_0$, the *canonical decision tree* of $F\restriction_\rho$ is of depth $D = O(\log(1/\delta))$ (the canonical decision tree is the decision tree that is constructed by the algorithm in Hastad's original proof; for a definition see, e.g., [TX13, Def. 4]).

Given a restriction ρ, we consider the following way to decide whether the canonical decision tree of $F\restriction_\rho$ is of depth D. Associate each string $P \in \{0,1\}^D$ with a potential *positional path* of depth D in the canonical decision tree of F; that is, the string P induces a path from the root to a specific node of depth D in a full binary tree of depth D or more. For each $P \in \{0,1\}^D$, we consider a corresponding test T_P that gets ρ as input, and tests whether or not one of the nodes in the path induced by P along the canonical decision tree of $F\restriction_\rho$ is a leaf node (i.e., whether or not the path ends at depth at most D); if there is indeed a leaf then T_P accepts ρ, and otherwise (i.e., if the path

[14] This is without loss of generality since if F is a DNF, then $F\restriction_\rho$ can be computed by a depth-D decision tree if and only if $(\neg F)\restriction_\rho$ can be computed by such a tree.

continues to depth $D+1$) then T_P rejects ρ. We will describe T_P in detail in a moment, but for now observe that the canonical decision tree of $F\restriction_\rho$ is of depth D if and only if for each $P \in \{0,1\}^D$ it holds that $T_P(\rho) = 1$.

To describe how each T_P works, fix $P \in \{0,1\}^D$, and let T_P be the following recursive algorithm. The algorithm gets as input a CNF F', a restriction ρ' and a string P' (in the first recursive call $F' = F$, $\rho' = \rho$, and $P' = P$). If the CNF is empty (i.e., has no clauses), then the algorithm accepts; otherwise, the algorithm examines the values that ρ' assigns to the variables in the first clause of F':

- If the first clause is unsatisfied by ρ' (i.e., all variables are fixed to unsatisfying values) then the algorithm accepts and halts.
- If the first clause is satisfied by ρ' (i.e., one or more variables are assigned to satisfying values), then the algorithm simplifies F' by omitting the first clause, and by simplifying the other clauses according to the values that ρ' assigned to the variables in the first clause. Then, the algorithm recurses with with the simplified CNF and with the same restriction ρ' and string P'.
- Otherwise, the first clause is undetermined by ρ'. If the number of living variables in the clause, denoted by k, is greater than the length of P', then the algorithm rejects.[15] If $k \leq |P'|$, let ρ'' be the restriction that fixes the k variables to values according to the k-prefix of P'. The algorithm simplifies F' according to the composition $\rho'' \circ \rho'$, and recurses with the simplified CNF, with the restriction $\rho'' \circ \rho'$, and with the string obtained from P' by omitting its first k bits.

The main point to note in the above description is that in each recursive call, the test T_P needs to read at most w blocks of $\lceil \log(1/p) \rceil = O(\log(w))$ bits in the restriction, corresponding to the (at most w) variables in the clause that it examines. The key observation in [GW14, Lemma 3.3], which we now state in a more general form, is that for each $P \in \{0,1\}^D$, with high probability it holds that T_P makes at most $D' = O\left(2^w \cdot \log\left(1/\delta_0\right)\right))$ recursive calls; that is, with high probability T_P examines the values that ρ assigns to variables of at most D' clauses. This is the case because for each recursive call, the probability that the clause that is examined is *unsatisfied* is at least 2^{-w}; thus, the probability that after D' recursive calls the algorithm encountered an unsatisfied clause, and thus stopped, is more than $1 - \left(1 - 2^{-w}\right)^{D'} \geq 1 - \delta_0$. It follows that for each $P \in \{0,1\}^D$, with probability at least $1 - 2\delta_0$ over a uniformly-chosen restriction ρ it holds that T_P accepts ρ without making more than D' recursive calls.

Now, consider "truncated" versions of these tests: For each $P \in \{0,1\}^D$, consider a modified version T'_P of T_P that, in addition to the description above, rejects ρ if the depth of the recursion exceeds D'. According to previous paragraph, the test T'_P

[15] This event means that the path induced by P in the canonical decision tree of $F\restriction_\rho$ is of depth more than $|P| = D$. Recall that by the definition of the canonical decision tree, whenever the algorithm that constructs the canonical decision tree encounters an undetermined clause, it adds the full sub-tree that corresponds to all living variables in the clause to the canonical decision tree.

accepts a uniformly-chosen restriction with probability at least $1 - 2\delta_0$. Since each T'_P reads at most $D'' = O(D' \cdot w \cdot \log(w)) = \tilde{O}(w) \cdot (2^w \cdot \log(1/\delta))$ bits in the restriction, if instead of the uniform distribution we choose a restriction from the distribution $\mathbf{z}$, which is δ'-almost t'-wise independent, where $\delta' < \left(\delta_0 \cdot 2^{-D''}\right)$ and $t' \geq D''$, then the probability that T'_P will accept is at least $1 - 3\delta_0$. [16] Thus, the probability that all the tests accept (i.e., $\wedge_{P \in \{0,1\}^D} T_P(\rho) = 1$) is at least $1 - 3\delta$. ■

3.3.5.2 Proof of a technical claim from Section 3.3.4.1

We now prove a technical claim from Section 3.3.4.1 (i.e., Claim 3.3.9). As mentioned in Section 3.3.4.1, the proof is elementary and relies on a case-analysis.

Claim 3.3.19. *(Claim 3.3.9, restated). Let $F : \{0,1\}^n \to \{0,1\}$ be a depth-2 formula of width w and size m, and let $F' : \{0,1\}^n \to \{0,1\}$ be a refinement of F. Then, for any restriction $\rho \in \{0,1,\star\}^n$ it holds that $F\restriction_\rho$ can be computed by a depth-2 formula Φ of width w and size m such that $F'\restriction_\rho$ is a refinement of Φ.*

Proof. We prove the claim for the case where F is a DNF; the proof for the case where F is a CNF follows by reduction to the DNF $\neg F$, relying on Fact 3.3.7. Let Φ be the DNF for $F\restriction_\rho$ that is obtained by fixing the variables in each clause of F according to ρ, without omitting any clause from the formula (even if a clause becomes a constant function).

When F' was obtained by a sequence of removal steps and clean-up steps, then F' is simply a sub-formula of F. In this case, we can apply the same sequence of removal steps and clean-up steps to Φ, to obtain a corresponding sub-formula of Φ that computes $F'\restriction_\rho$. [17] We thus focus on proving the claim when F' was obtained by a sequence of $k \leq m$ merging steps and clean-up steps.

For every $i \in [k]$, let $F^{(i)}$ be the formula in the beginning of the i^{th} refinement step in the transformation of F to F', and let $F^{(k+1)} = F'$. We will show a sequence of k merging steps and clean-up steps that, when applied to Φ, induce a corresponding sequence of formulas $\Phi = \Phi^{(1)}, ..., \Phi^{(k+1)}$, such that the following holds: For every $i \in [k]$ there exists a bijection between the clauses of $\Phi^{(i)}$ and the clauses of $F^{(i)}\restriction_\rho$ such that every clause φ of the former is mapped to a clause f of the latter such that φ computes the function $f\restriction_\rho$. In particular, this claim implies that for every $i \in [k]$ it holds that $\Phi^{(i)} \equiv F^{(i)}\restriction_\rho$, and therefore $F'\restriction_\rho \equiv \Phi^{(k+1)}$ is a refinement of $\Phi = \Phi^{(1)}$.

[16] The reason that we use the error parameter $\delta_0 \cdot 2^{-D''}$ instead of the more natural parameter δ_0 is that the tests that we are trying to "fool" are *adaptive*; that is, for each $P \in \{0,1\}^D$, the test T_P does not examine a fixed set of D'' bits in ρ, but rather adaptively chooses which bits to read according to the values of the bits that it read so far. We rely on the fact that any distribution that is $\left(\delta_0 \cdot 2^{-D''}\right)$-almost D''-wise independent also δ_0-fools adaptive tests that only read D'' bits (see, e.g., [Gol17, Exer. 7.4]).

[17] That is, let $F = \vee_{i=1}^m f_i$, and assume that $F' = \vee_{i=k+1}^m f_i$ was obtained from F by removing the clauses $f_1, ..., f_k$. Then it holds that $\Phi = \vee_{i=1}^m (f_i\restriction_\rho)$ and $F'\restriction_\rho = \vee_{i=k+1}^m (f_i\restriction_\rho)$, which implies that we can apply k removal steps to Φ in order to obtain $F'\restriction_\rho$.

The claim is proved by induction on i. The base case $i = 1$ follows immediately from the definition of $\Phi^{(1)} = \Phi$. For the induction step, assume that there is a bijection as above between the clauses of $\Phi^{(i)}$ and the clauses of $F^{(i)}\restriction_\rho$, and let us define the i^{th} refinement step that is applied to $\Phi^{(i)}$. If the i^{th} refinement step of $F^{(i)}$ was a clean-up step, then we can apply an analogous clean-up step to $\Phi^{(i)}$. [18] Otherwise, if the i^{th} refinement step of $F^{(i)}$ was a merging step, let $f_1^{(i)}, ..., f_u^{(i)}$ be the set of clauses that were removed in this step, and let $h^{(i)}$ be the new clause that was added in their stead. For every $j \in [u]$, let $\varphi_j^{(i)}$ be the clause in $\Phi^{(i)}$ that computes $f_j^{(i)}\restriction_\rho$ and exists by the induction hypothesis. We show how apply a single refinement step to $\Phi^{(i)}$ that replaces the clauses $\varphi_1^{(i)}, ..., \varphi_u^{(i)}$ with a new clause $\varphi^{(i)}$ that computes the function $h^{(i)}\restriction_\rho$. This is proved by a case analysis:

1. If $h^{(i)}\restriction_\rho$ is not a constant function, then it follows that $\bigcap_{j\in[u]}(f_j^{(i)}\restriction_\rho) = \bigcap_{j\in[u]}\varphi_j^{(i)} \neq \emptyset$. In this case, we apply a merging step to the clauses $\varphi_1^{(i)}, ..., \varphi_u^{(i)}$ in $\Phi^{(i)}$, and they are replaced with the non-constant clause $\varphi^{(i)} = \bigcap_{j\in[u]}\varphi_j^{(i)} = \bigcap_{j\in[u]}(f_j^{(i)}\restriction_\rho) = h^{(i)}\restriction_\rho$.

2. If $h^{(i)}\restriction_\rho \equiv 0$, then for every $j \in [u]$ it holds that $f_j^{(i)}\restriction_\rho \equiv \varphi_j^{(i)} \equiv 0$. This is the case because $\bigcap_{j\in[u]} f_j^{(i)} \neq \emptyset$ (otherwise $h^{(i)} \equiv 1$ and also $h^{(i)}\restriction_\rho \equiv 1$), whereas $\left(\bigcap_{j\in[u]} f_j^{(i)}\right)\restriction_\rho \equiv 0$, which implies that for every $j \in [u]$ there exists a literal in $f_j^{(i)}$ that is fixed by ρ to an unsatisfying value. Therefore, in this case we can apply a clean-up step to $\Phi^{(i)}$ to remove all but a single constant zero clause among the $f_j^{(i)}$'s.

3. If $h^{(i)}\restriction_\rho \equiv 1$, then it holds that $\bigcap_{j\in[u]}\varphi_j^{(i)} = \emptyset$. To see that this is the case, note that if $\bigcap_{j\in[u]} f_j^{(i)} = \emptyset$ then the latter assertion holds immediately; and otherwise (i.e., $\bigcap_{j\in[u]} f_j^{(i)} \neq \emptyset$), it follows by the assumption that $h^{(i)}\restriction_\rho \equiv 1$ that ρ fixes all the literals that are shared by all the u clauses $f_1^{(i)}, ..., f_u^{(i)}$ to satisfying values, which indeed implies that $\bigcap_{j\in[u]}\varphi_j^{(i)} = \emptyset$. Thus, we can apply a merging step to $\varphi_1^{(i)}, ..., \varphi_u^{(i)}$ to obtain the constant one function. ■

[18] Specifically, denote by $f_1^{(i)}, ..., f_u^{(i)}$ the constant zero clauses that were removed from $F^{(i)}$ in the i^{th} step. For every $j \in [u]$, let $\varphi_j^{(i)}$ be the clause in $\Phi^{(i)}$ that computes $f_j^{(i)}\restriction_\rho \equiv 0$ and exists by the induction hypothesis. Then, the i^{th} refinement step of $\Phi^{(i)}$ is a clean-up step that removes the constant zero clauses $\varphi_1^{(i)}, ..., \varphi_u^{(i)}$.

3.4 Constant-depth circuits with parity gates

3.4.1 The main results

We now study constant-depth circuits that also have gates computing the parity function or the negated parity function; that is, we study $\mathcal{AC}^0[\oplus]$. Specifically, we consider $\mathcal{AC}^0[\oplus]$ circuits that are *layered*, in the sense that all gates at a particular distance from the input gates are of the same gate-type.

We first observe that the standard derandomization problem of CNFs can be reduced to the problem of derandomizing layered $\mathcal{AC}^0[\oplus]$ circuits of *depth four* with $B(n) = 2^{n^c}$ exceptional inputs, which yields a "threshold" at depth four with such a $B(n)$. This improves on a similar result of [GW14] that refers to depth five.

Theorem 3.4.1 (a threshold for quantified derandomization of $\mathcal{AC}^0[\oplus]$ at depth four)**.** *Assume that, for some $c > 0$, there exists a polynomial-time algorithm A such that, when A is given as input a layered depth-four $\mathcal{AC}^0[\oplus]$ circuit C over n input bits that accepts all but $B(n) = 2^{n^c}$ of its inputs, then A finds a satisfying input for C. Then, there exists a polynomial-time algorithm A' that, when given as input a polynomial-size CNF that accepts at least a $1/\mathrm{poly}(n)$ fraction of its inputs, then A' finds a satisfying input for the CNF.*

An appealing way to approach this "threshold" at depth four (with $B(n) = 2^{n^c}$) is to derandomize $\mathcal{AC}^0[\oplus]$ *circuits of depth three* with $B(n) = 2^{n^c}$. Goldreich and Wigderson derandomized most types of layered depth-3 $\mathcal{AC}^0[\oplus]$ circuits with $B(n) = 2^{n^c}$, for any $c < 1$, with the exception of circuits of the form $\oplus \wedge \oplus$ (i.e., top $\oplus$ gate, middle layer of $\wedge$ gates, and a bottom layer of $\oplus$ gates), which they left as an open problem.

Our main result in this section is an algorithm that makes significant progress on this problem, by derandomizing $\oplus \wedge \oplus$ circuits with $B(n) = 2^{n^c}$ under various subquadratic upper bounds on the circuit size, where some of these bounds refer to each layer separately.

Theorem 3.4.2 (hitting biased $\oplus \wedge \oplus$ circuits)**.** *Let $\epsilon > 0$ be an arbitrary constant. Let $\mathcal{C}$ be the class of circuits of depth three with a top $\oplus$ gate, a middle layer of $\wedge$ gates, and a bottom layer of $\oplus$ gates, such that every $C \in \mathcal{C}$ over n input bits satsifies (at least) one of the following:*

1. *The size of C is $O(n)$.*
2. *The number of $\wedge$-gates is at most $n^{2-\epsilon}$, and the number of $\oplus$-gates is at most $n + n^{\epsilon/2}$.*
3. *The number of $\oplus$-gates is at most $n^{1+\epsilon}$, and the number of $\wedge$-gates is at most $\frac{1}{5} \cdot n^{1-\epsilon}$.*

Then, for some $c = c(\epsilon) > 0$, there exists a polynomial-time algorithm that, when given a circuit $C \in \mathcal{C}$ that accepts all but $B(n) = 2^{n^c}$ of its inputs, outputs a satisfying input for C.

We stress that the algorithm from Theorem 3.4.2 makes essential use of the specific circuit C that is given to the algorithm as input.

Lastly, we affirm a conjecture of Goldreich and Wigderson by showing, loosely speaking, that "hitting" $\oplus \wedge \oplus$ circuits reduces to "hitting" *polynomials that vanish*

rarely.[19] The latter problem will be presented in detail in Section 3.6; for now, let us confine our interest to degree-d polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish on at most $\epsilon > 0$ of their inputs, where $d = d(n)$ might be large. If $\epsilon < 2^{-d}$ then this problem is trivial (because the only such polynomial is the constant one function), and solutions for the case of $\epsilon = c \cdot 2^{-d}$, where $c \in \mathbb{N}$ is constant, are known (see Section 3.6 for details).

The following result asserts that to "hit" n-bit $\oplus \wedge \oplus$ circuits of size m and with $B(n) = \Omega(2^n)$ exceptional inputs, it suffices to construct, for some $d = d(n)$ (i.e., any d would do), a sufficiently dense hitting-set generator for degree-d polynomials with $\epsilon = m \cdot 2^{-d}$. For example, for $\oplus \wedge \oplus$ circuits of size $m = m(n) = \text{poly}(n)$, it suffices to construct a hitting-set generator for polynomials of degree $d = O(\log(n))$ that vanish on at most $1/p(n)$ of their inputs, where $p(n) > 2^{d(n)}$ is a sufficiently large polynomial.

Theorem 3.4.3 (reducing hitting $\oplus \wedge \oplus$ circuits to hitting biased polynomials of bounded degree)**.** *Let $\mathcal{C}$ be the class of $\oplus \wedge \oplus$ circuits over n input bits with $m = m(n)$ $\wedge$-gates that accept all but $B(n) = \epsilon \cdot 2^n$ of their inputs, where $m(n) = o(2^n)$ and $\epsilon = \epsilon(n) \leq 1/8$. Let $\mathcal{P}$ be the class of polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ of degree $d = \lfloor \log(m(n)) + \log(1/\epsilon) \rfloor$ that accept all but a $b(n) = (4 \cdot m(n)) \cdot 2^{-d} = 4 \cdot \epsilon$ fraction of their inputs. Then, any hitting-set generator with density $1/2 + 2 \cdot \epsilon$ for $\mathcal{P}$ is also a hitting-set generator for $\mathcal{C}$.*

3.4.2 Proof overviews

Let us now describe the high-level strategy of the algorithms of Theorem 3.4.2. First observe that any $\oplus \wedge \oplus$ circuit C computes an n-variate polynomial over $\mathbb{F}_2$, and that the total degree of this polynomial equals the maximal fan-in of $\wedge$-gates in the circuit. Our approach will be to find an *affine subspace* W of dimension more than $\log(B(n))$ such that when C is restricted to the affine subspace, the fan-in of all $\wedge$-gates becomes constant. Thus, when restricted to W, the circuit C becomes a non-zero polynomial of constant degree, which means that we can then hit it using a pseudorandom generator for polynomials of constant degree (i.e., Viola's [Vio09b]).

In order to find the affine subspace W, we use *affine restrictions*, which are obtained by fixing values to some of the bottom $\oplus$-gates. These are analogous to standard "bit-fixing" restrictions, but in contrast to the latter, we cannot consider *any* sequence of fixed values to the bottom $\oplus$-gates: This is the case because the bottom $\oplus$-gates might not be linearly independent (and thus the values of some $\oplus$-gates might depend on the values of other $\oplus$-gates). In particular, this means that we cannot use random (or pseudorandom) restrictions in which the value of each $\oplus$-gate is chosen obliviously of the $\oplus$-gates of the circuit.

Our algorithm circumvents this problem by constructing a restriction that corresponds to the *specific* $\oplus \wedge \oplus$ circuit that is given to the algorithm as input. For concreteness, let us now describe the construction of Item (2) of Theorem 3.4.2, and let

[19] In [GW14, Sec. 6 (full version)] it is suggested to prove this result by modifying any $\oplus \wedge \oplus$ circuit to a bounded-degree polynomial, where the modification amounts to the removal of all $\wedge$-gates with high fan-in. However, as explained in Section 3.4.2, since the top gate is a $\oplus$-gate, we cannot simply remove $\wedge$-gates with high fan-in (or remove some of the wires that feed into them).

us also fix specific parameter values to work with: We assume, for simplicity, that the number of bottom $\oplus$-gates is *exactly* n; and we assume that the number of $\wedge$-gates is $n^{1.1}$, and that the circuit accepts all but $\Omega\left(2^{n^{1/3}}\right)$ of its inputs.

First assume, for a moment, that the fan-in of each $\wedge$-gate in the middle layer of the circuit is upper bounded by $\sqrt{n}$. In this case we can restrict the $\oplus$-gates as follows. Consider a random restriction process in which each bottom $\oplus$-gate is fixed independently with probability $1 - p = 1 - n^{-2/3}$, and the *values* for the fixed gates are chosen afterwards, in an *arbitrary consistent manner*. With high probability, the restriction will yield a subspace of dimension approximately $p \cdot n = n^{1/3} > \log(B(n))$. Also, since each $\wedge$-gate g has fan-in at most $w = \sqrt{n}$, and $p = 1/w^{1+\Omega(1)}$, with high probability, all but $O(1)$ of the gates that feed into g are fixed by this process.[20] In fact, the above two statements hold even if we choose the restriction according to an $O(1)$-independent distribution, rather than uniformly.

Needless to say, we cannot actually assume that the fan-in of $\wedge$-gates is bounded by $\sqrt{n}$. Thus, our strategy will be to first *mildy* reduce the fan-in of $\wedge$-gates (from n to $\sqrt{n}$), and then invoke the restriction process described above. A standard approach to mildly reduce the fan-in of $\wedge$-gates is to simply remove some of the incoming wires to each $\wedge$-gate. However, this approach *does not* work in our setting, since the top gate is a $\oplus$-gate, which means that such a modification might turn unsatisfying inputs into satisfying ones (and thus hitting the modified circuit might not yield a satisfying input to the original circuit).

To reduce the fan-in of $\wedge$-gates to $\sqrt{n}$, we follow Kopparty and Srinivasan [KS12] in adapting the approach of Chaudhuri and Radhakrishnan [CR96] to the setting of $\oplus \wedge \oplus$ circuits.[21] Specifically, we first iteratively fix each $\oplus$-gate that has *fan-out* more than $n^{1/4}$ to a *non-accepting* value; note that such an action also fixes $n^{1/4}$ $\wedge$-gates in the middle layer, and hence in this step we fix values for at most $n^{1.1}/n^{1/4} = o(n)$ bottom $\oplus$-gates (because afterwards there are no more living $\wedge$-gates). At this point, the number of wires feeding the middle layer is at most $n \cdot n^{1/4} = n^{1.25}$. Now, for each $\wedge$-gate g with *fan-in* more than $\sqrt{n}$, we fix a $\oplus$-gate that feeds into g to a *non-accepting* value, thereby also fixing g; each such action eliminates $\sqrt{n}$ wires that feed into the middle layer, and therefore in this step we fix at most $n^{1.25}/\sqrt{n} = o(n)$ bottom $\oplus$-gates. Overall, the fan-in of each $\wedge$-gate has been reduced to $\sqrt{n}$, and we imposed at most $o(n)$ affine conditions.

To see that the final subspace W is of dimension more than $\log(B(n))$, note that the dimension of W equals the number of living $\oplus$-gates (because we assumed that the initial number of $\oplus$-gates is exactly n). After the first step of the algorithm (i.e., reducing the fan-in of $\wedge$-gates to $\sqrt{n}$), we are left with $(1 - o(1)) \cdot n$ living $\oplus$-gates,

[20] For any $\wedge$-gate g with initial fan-in $d_\wedge$, the probability that there exists a set of size c of $\oplus$-gates that feed into g that are all unfixed is at most $\binom{d_\wedge}{c} \cdot p^c = 1/\mathrm{poly}(n)$, for a sufficiently large $c = O(1)$.

[21] Originally, [CR96] applied their approach to $\mathcal{AC}^0$ circuits, and [KS12] later adapted this approach to $\mathcal{AC}^0[\oplus]$ circuits. Our adaptation is slightly different technically than in [KS12], to suit the specific circuit structure $\oplus \wedge \oplus$; but more importantly, while both [CR96; KS12] use the approach as part of the analysis (to prove lower bounds), we use this approach as a (non-black-box) algorithm for derandomization.

and the second step (i.e., the pseudorandom restriction) leaves a fraction of $p = n^{-2/3}$ of them alive. Thus, the expected dimension of W is $\Omega(p \cdot n) = \Omega\left(n^{1/3}\right) > \log(B(n))$.

The approach above actually works for a broader range of parameters, and in particular when the number of $\wedge$-gates is $n^{2-\epsilon}$, for any constant $\epsilon > 0$, and when the number of $\oplus$-gates is $n + n^c$, for any $c < \epsilon$ (see details in Section 3.4.4.3). In Items (1) and (3), we consider circuits in which the number of $\oplus$-gates is significantly larger than n, namely $O(n)$ and $O\left(n^{1+\epsilon}\right)$, respectively. The proofs of both these items use algorithms that are variations of the first step of the algorithm described above, and these proofs are detailed in Sections 3.4.4.2 and 3.4.4.4, respectively.

3.4.3 Proof of Theorem 3.4.1

The proof is similar to the proof of Theorem 3.3.1, and is a variation on [GW14, Thm 4.2 and Remark 4.4]. Starting from a CNF C that accepts $1/\text{poly}(n)$ of its inputs, we will employ error-reduction within $\mathcal{AC}^0[\oplus]$, by first sampling inputs for C using Trevisan's extractor [Tre01], and then taking the disjunction of the evaluation of C on these inputs (rather than an approximate majority, as in [GW14]). This will yield a layered circuit of the form $\vee \wedge \vee \oplus$ that accepts all but 2^{n^c} of its inputs, for any desired $c > 0$. Details follow.

Let $C : \{0,1\}^n \to \{0,1\}$ be a CNF that accepts an $\epsilon = 1/\text{poly}(n)$ fraction of its inputs. For $n' = n^{(1/c)+1}$ and $s = O(\log(n))$, let $E : \{0,1\}^{n'} \times \{0,1\}^s \to \{0,1\}^n$ be Trevisan's extractor instantiated for min-entropy $(n')^c = n^{1+\Omega(1)}$ and error parameter $\epsilon/2 = 1/\text{poly}(n)$. We construct a circuit $C' : \{0,1\}^{n'} \to \{0,1\}$ that first computes the values $E(x,z)$, for each possible seed $z \in \{0,1\}^s$, then evaluates C on each value $E(x,z)$, and finally takes an OR of these evaluations; that is, $C'(x) = \vee_{z \in \{0,1\}^s} C\left(E(x,z)\right)$.

Note that C' is a layered depth-4 circuit of the form $\vee \wedge \vee \oplus$, since for each seed $z \in \{0,1\}^s$, the residual function $E_z(x) = E(x,z)$ is just a linear transformation of x. Also note that the number of inputs $x \in \{0,1\}^{n'}$ for which $\Pr_z[C(E(x,z))] < \epsilon/2$ is at most $2^{(n')^c}$. In particular, C' accepts all but at most $2^{(n')^c}$ of its inputs, and for each satisfying input x for C', we can find a corresponding satisfying input for C among $\{E(x,z)\}_{z \in \{0,1\}^s}$.

3.4.4 Proof of Theorem 3.4.2

The current section is organized as follows. In Section 3.4.4.1 we present two algorithmic tools that will be used in the proof: An adaptation of the approach of Chaudhuri and Radhakrishnan [CR96] to the setting of $\oplus \wedge \oplus$ circuits, and an adaptation of Viola's pseudorandom generator [Vio09b] to polynomials that are defined over an affine subspace. Then, in the next three sections, we prove the corresponding three items of Theorem 3.4.2.

We rely on the notion of *affine restrictions*. A restriction of a circuit $C : \{0,1\}^n \to \{0,1\}$ to an affine subspace $W \subseteq \{0,1\}^n$ will be constructed by accumulating a list of (independent) affine conditions that defines W. That is, each of the various algorithms

will construct a full-rank matrix A and a vector b such that $W = \{x : Ax = b\}$. For an affine function g, when we say that an algorithm "adds $g = 0$ to the list of affine conditions", we mean that it extends A by adding the linear part of g as an additional row to A, and extends b by adding the constant term of g as an additional bit to b (i.e., if $g(x) = \sum_{i=1}^{n} c_i x_i + c_0$ then the row $c = (c_1, ..., c_n)$ is added to A and c_0 is added to b). After each addition of a condition, we will say that the algorithm "simplifies the circuit accordingly"; by this we mean that for any $\oplus$-gate g' in the bottom layer whose linear function is dependent on the rows of A, the algorithm fixes g' to the appropriate value determined by A and b, and, if g' was fixed to zero, then the algorithm removes all the $\wedge$-gates that g' feeds into.

3.4.4.1 Two algorithmic tools

Let us first adapt the approach of Chaudhuri and Radhakrishnan [CR96], which was originally used to construct "bit-fixing" restrictions for $\mathcal{AC}^0$ circuits, to the setting of $\oplus \wedge \oplus$ circuits and affine restrictions.

Proposition 3.4.4 (whitebox affine restrictions for $\oplus \wedge \oplus$ circuits)**.** *For two integers $m_\wedge$ and $m_\oplus$, let $\mathcal{C}$ be the class of $\oplus \wedge \oplus$ circuits over n input bits with $m_\wedge$ gates in the middle layer and $m_\oplus$ gates in the bottom layer. Then, for any two integers $d_\oplus$ and $d_\wedge$, there exists a polynomial-time algorithm that, when given as input a circuit $C \in \mathcal{C}$, outputs an affine subspace $W \subseteq \{0,1\}^n$ such that:*

1. *In the restriction of C to W, each $\wedge$-gate in the middle layer has fan-in at most $d_\wedge$.*
2. *The subspace W is of co-dimension at most $\frac{m_\wedge}{d_\oplus} + \frac{d_\oplus \cdot m_\oplus}{d_\wedge}$.*

Proof. The algorithm operates in two steps. In the first step, as long as there exists a $\oplus$-gate g in the bottom layer with *fan-out* at least $d_\oplus$, the algorithm adds the condition $g = 0$ to the list of affine conditions, and simplifies the circuit accordingly. Note that each addition of a condition as above fixes at least $d_\oplus$ of the $\wedge$-gates in the middle layer, and thus at most $m_\wedge / d_\oplus$ conditions are added (or else the entire circuit simplifies to a constant). Hence, after the first step concludes, the fan-out of each $\oplus$-gate in the bottom layer is $d_\oplus$, and at most $m_\wedge / d_\oplus$ affine conditions have been accumulated.

In the second step, as long as there exists an $\wedge$-gate g in the middle layer with *fan-in* at least $d_\wedge$, the algorithm (arbitrarily) chooses one $\oplus$-gate g' that feeds into g, adds the condition $g' = 0$ to the list of affine conditions, and simplifies the circuit accordingly. Note that, in the beginning of the second step, the number of wires feeding the middle layer is at most $d_\oplus \cdot m_\oplus$ (since there are at most $m_\oplus$ gates in the bottom layer, each of them with fan-out at most $d_\oplus$). Now, note that each addition of an affine condition in the second step eliminates at least $d_\wedge$ wires; thus, the algorithm adds at most $\frac{d_\oplus}{d_\wedge} \cdot m_\oplus$ conditions in the second step. After the second step is complete, each $\wedge$-gate in the middle layer has fan-in at most $d_\wedge$, and the list of affine conditions contains at most $m_\wedge / d_\oplus + \frac{d_\oplus}{d_\wedge} \cdot m_\oplus$ conditions. ■

We now verify that we can use Viola's pseudorandom generator [Vio09b] in order to "fool" $\oplus \wedge \oplus$ circuits that, when restricted to an affine subspace, have a constant maximal fan-in of the $\wedge$-gates.

Proposition 3.4.5 (invoking Viola's PRG in an affine subspace)**.** *There exists an algorithm G that, for every $n \in \mathbb{N}$, when G is given as input an integer D, a seed of $\ell = O(\log(n))$ bits, and a basis for an affine subspace $W \subseteq \{0,1\}^n$, then G runs in time $\mathrm{poly}(n)$ and satisfies the following: For every $\oplus \wedge \oplus$ circuit C over n input bits such that C simplifies under the restriction W to a $\oplus \wedge \oplus$ circuit in which the maximal fan-in of $\wedge$-gates is D and such that $C\restriction_W \not\equiv 0$, it holds that $\Pr[C(G(\mathbf{u}_\ell)) = 1] > 0$.*

Proof. Denote the dimension of W by $m = \dim(W)$. The algorithm G first finds a full-rank $n \times m$ matrix B and $s \in \{0,1\}^n$ such that $x \mapsto Bx + s$ maps $\{0,1\}^m$ to W. Then, the algorithm G uses its random seed to invoke Viola's pseudorandom generator for polynomials $\mathbb{F}_2^m \to \mathbb{F}_2$ of degree D, with error parameter $2^{-(D+1)}$, thus obtaining a string $x \in \{0,1\}^m$. Finally, the algorithm G outputs the string $Bx + s$.

Now, let C be $\oplus \wedge \oplus$ circuit as in the hypothesis, and consider the polynomial $p : \mathbb{F}_2^m \to \mathbb{F}_2$ such that $p(x) = C(Bx + s)$. Note that p is of degree D, because C computes an sum of monomials of degree D over $\mathbb{F}_2$, and the affine transformation does not increase the degree. Also, using our hypothesis that p is non-zero, it follows that the acceptance probability of p is at least 2^{-D}. Thus, the probability that Viola's generator will output x such that $p(x) = 1$ is at least $2^{-(D+1)} > 0$, and each such x yields a string $y = Bx + s$ such that $C(y) = 1$. ∎

3.4.4.2 Linear-sized circuits with $B(n) = 2^{-\Omega(n)}$

We prove the first item of Theorem 3.4.2 by invoking the whitebox algorithm from Proposition 3.4.4 with appropriate parameters $d_\wedge, d_\oplus = O(1)$, and then using the generator from Proposition 3.4.5.

Proposition 3.4.6 (Theorem 3.4.2, Item (1): hitting biased linear-sized $\oplus \wedge \oplus$ circuits)**.** *Let $\epsilon > 0$ be an arbitrarily small constant, and let $c > 0$ be an arbitrarily large constant. Let $\mathcal{C}$ be the class of $\oplus \wedge \oplus$ circuits such that any circuit $C \in \mathcal{C}$ over n input bits has at most $c \cdot n$ gates and accepts all but at most $2^{(1-\epsilon)\cdot n}$ of its inputs. Then, there exists a polynomial-time algorithm that, when given any circuit $C \in \mathcal{C}$, finds a satisfying input for C.*

Proof. The algorithm first invokes the algorithm from Proposition 3.4.4 with parameters $d_\oplus = \frac{4 \cdot c}{\epsilon}$ and $d_\wedge = d_\oplus^2$, to obtain an affine subspace W of co-dimension at most

$$\frac{m_\wedge}{d_\oplus} + \frac{d_\oplus \cdot m_\oplus}{d_\wedge} < 2 \cdot \frac{c \cdot n}{(4 \cdot c)/\epsilon} = \frac{\epsilon}{2} \cdot n$$

such that in the restriction of C to W, every $\wedge$-gate in the middle layer has fan-in at most $d_\wedge = O(1)$. Since the circuit C has at most $2^{(1-\epsilon)\cdot n}$ unsatisfying inputs, it follows that $\Pr_{w \in W}[C(w) = 1] \geq 1 - 2^{-(\epsilon/2)\cdot n}$. Thus, the algorithm concludes by invoking the algorithm from Proposition 3.4.5. ∎

3.4.4.3 Sub-quadratic circuits with $(1+o(1))\cdot n$ bottom $\oplus$-gates and $B(n)=2^{n^c}$

We now prove the second item of Theorem 3.4.2.

Proposition 3.4.7 (Theorem 3.4.2, Item (2): hitting biased sub-quadratic $\oplus\wedge\oplus$ circuits)**.** *Let $\epsilon>0$ and let $0<c<\epsilon$. Let $\mathcal{C}$ be the class of $\oplus\wedge\oplus$ circuits such that any $C\in\mathcal{C}$ over n input bits has at most $n+n^c$ bottom $\oplus$-gates, and at most $n^{2-\epsilon}$ middle $\wedge$-gates, and accepts all but $B(n)=2^{n^c}$ of its inputs. Then, there exists a polynomial-time algorithm that, when given any circuit $C\in\mathcal{C}$, finds a satisfying input for C.*

Proof. Recall that a high-level overview of the proof, which used the parameter values $m_\wedge=n^{1.1}$ and $m_\oplus=n$, appeared in Section 3.4.2. Let us first explain, in high-level, how to handle the setting of $m_\wedge\leq n^{2-\epsilon}$; for the moment, we are still assuming that $m_\oplus=n$. As in the overview in Section 3.4.2, the algorithm works in two steps. In the first step, we use Proposition 3.4.4 to fix $o(m_\oplus)$ of the $\oplus$-gates such that after the restriction, the fan-in of the $\wedge$-gates is bounded by $w=n^{1-\alpha\cdot\epsilon}$, where $\alpha<1$ is a constant slightly smaller than 1; this is possible because $m_\wedge\leq n^{2-\epsilon}$ (see the proof details below). In the second step, we restrict the $\oplus$-gates using an $O(1)$-independent distribution, keeping each $\oplus$-gate alive with probability $p=n^{-(1-\beta\cdot\epsilon)}$, where $\beta<\alpha$ (and recall that we choose arbitrary consistent values for the gates that are fixed). The crucial point is the following: On the one hand, since $p\leq 1/w^{1+\Omega(1)}$, after the second step the fan-in of the $\wedge$-gates is upper-bounded by a constant (as explained in Section 3.4.2); and on the other hand, the number of living $\oplus$-gates after the second step is approximately $p\cdot(1-o(1))\cdot n=\Omega\left(n^{\beta\cdot\epsilon}\right)>n^c=\log(B(n))$, where the inequality holds if we choose $\beta>c/\epsilon$ (which is possible if we initially choose $\alpha\in(c/\epsilon,1)$).

To see how we handle the setting of $m_\oplus\leq n+n^c$ (rather than $m_\oplus=n$), note that the overall number of affine conditions that the algorithm imposes is $m_\oplus-\Omega(p\cdot m_\oplus)$. Since $m_\oplus\leq n+o(p\cdot n)$, the number of affine conditions is at most $n-\Omega(p\cdot n)$, which means that the affine subspace W is of dimension $\Omega(p\cdot n)>\log(B(n))$.

Let us now provide the full details for the proof. Assume, without loss of generality, that $m_\oplus\geq n$ (we can add dummy gates if necessary). We first invoke the algorithm from Proposition 3.4.4 with parameters $d_\wedge=n^{1-\alpha\cdot\epsilon}$, where $\alpha=\frac{(c/\epsilon)+1}{2}$, and $d_\oplus=n^{1-\alpha'\cdot\epsilon}$, where $\alpha'=(c/\epsilon)+(2/3)\cdot(1-c/\epsilon)>\alpha$. The algorithm outputs an affine subspace of co-dimension at most

$$\begin{aligned}\frac{m_\wedge}{d_\oplus}+\frac{d_\oplus\cdot m_\oplus}{d_\wedge}&\leq n^{2-\epsilon-(1-\alpha'\cdot\epsilon)}+n^{1-\alpha'\cdot\epsilon-(1-\alpha\cdot\epsilon)}\cdot m_\oplus\\&=n^{1-(1-\alpha')\cdot\epsilon}+n^{-(\alpha'-\alpha)\cdot\epsilon}\cdot m_\oplus\,,\end{aligned}$$

which is $o(m_\oplus)$, such that in the restriction of C to the subspace, every $\wedge$-gate in the middle layer has fan-in at most $d_\wedge=n^{1-\alpha\cdot\epsilon}$.

Denote the number of $\oplus$-gates that were not fixed in the previous step by m', and consider the following pseudorandom restriction process. For a sufficiently large constant $\gamma>1$ (which will be determined later), we use a γ-wise independent distribution

over $[1/p]^{n'}$, where $p = n^{-(1-\beta\cdot\epsilon)}$ and $\beta = (c/\epsilon) + (1/3)\cdot(1-c/\epsilon) < \alpha$. [22] Denote the random variable that is the output string of this distribution by $\rho \in [1/p]^{n'}$. For every $\oplus$-gate that has not been restricted by the algorithm from Proposition 3.4.4, the algorithm now marks the gate as "alive" if and only if the corresponding element in the string ρ equals zero; otherwise, it marks the gate as "fixed".

For any $\wedge$-gate g in the middle-layer, the probability that at least γ gates that feed into g are marked "alive" is at most

$$\binom{d_\wedge}{\gamma}\cdot p^\gamma < n^{(1-\alpha\cdot\epsilon)\cdot\gamma}\cdot n^{-(1-\beta\cdot\epsilon)\cdot\gamma} = n^{-(\alpha-\beta)\cdot\epsilon\cdot\gamma}\ ,$$

which can be made less than $1/m_\wedge = n^{-(2-\epsilon)}$ by an appropriate choice of γ (i.e., $\gamma > \frac{2-\epsilon}{(\alpha-\beta)\cdot\epsilon}$). After union-bounding over all $\wedge$-gates, we have that with probability at least 0.99, each $\wedge$-gate is fed by less than γ of the "alive" $\oplus$-gates. Also note that with probability at least 0.99, the number of $\oplus$-gates that were marked as "alive" is at least $(p\cdot m')/2$; this is because the distribution is γ-wise independent (so we can use Fact 2.1.1). The algorithm and finds a choice of ρ, denoted by ρ_0, that meets both these conditions (by enumerating the outputs of the γ-wise independent distribution). Then, the algorithm iteratively fixes values for the $\oplus$-gates that are marked as "fixed" by ρ_0. Specifically, as long as there is a $\oplus$-gate g that is marked as "fixed" by ρ_0, the algorithm adds the condition $g = 0$ to the list of affine conditions that defines W, and simplifies the circuit accordingly.

Let us now count the number of affine conditions that the algorithm imposed (i.e., the co-dimension of W). After all the restrictions, the number of living variables is at least $(p/2)\cdot m' \geq (p/2)\cdot(1-o(1))\cdot m_\oplus \geq (p/3)\cdot m_\oplus$, which implies that the number of affine conditions is at most $m_\oplus - (p/3)\cdot m_\oplus$. Since $m_\oplus \leq n + n^c$, we have that

$$\begin{aligned} m_\oplus - (p/3)\cdot m_\oplus &< n + n^c - (p/3)\cdot n \\ &= n + n^c - \frac{1}{3}\cdot n^{\beta\cdot\epsilon}\ , \end{aligned}$$

which is less than $n - n^c$, because $n^c = o(n^{\beta\cdot\epsilon})$ (since $\beta\cdot\epsilon = c + \Omega(1)$).

Thus, the algorithm is left with a subspace W of dimension more than $n^c = \log(B(n))$ such that when the circuit C is restricted to the subspace W, the fan-in of every $\wedge$-gate in the middle layer is at most $\gamma = O(1)$. Hence, at this point the algorithm can invoke the algorithm from Proposition 3.4.5, and find a satisfying input for C in W. ∎

3.4.4.4 Circuits with a slightly super-linear number of bottom $\oplus$-gates and slightly sub-linear number of $\wedge$-gates

We now prove the third item of Theorem 3.4.2. The crucial observation here is that after invoking the algorithm from Proposition 3.4.4, the number of $\oplus$-gates is at most

[22] We will actually use the value $p = 2^{-\lceil(1-\beta\cdot\epsilon)\cdot\log(n)\rceil}$, such that $1/p$ is a power of 2, but the difference between this value and $n^{-(1-\beta\cdot\epsilon)}$ is insignificant in what follows.

$m_\wedge \cdot d_\wedge$, since this is the number of wires that feed into the middle layer.

Proposition 3.4.8 (Theorem 3.4.2, Item (3): hitting biased $\oplus \wedge \oplus$ circuits with a super–linear number of $\oplus$-gates)**.** *For any constant $\epsilon > 0$, let $\mathcal{C}$ be the class of $\oplus \wedge \oplus$ circuits such that any circuit $C \in \mathcal{C}$ over n input bits has at most $n^{1+\epsilon}$ gates in the bottom layer and at most $(1/5) \cdot n^{1-\epsilon}$ gates in the middle layer, and accepts all but at most $B(n) = 2^{n/15}$ of its inputs. Then, there exists a polynomial-time algorithm that, when given any circuit $C \in \mathcal{C}$, finds a satisfying input for C.*

Proof. We first invoke the algorithm from Proposition 3.4.4 with parameters $d_\oplus = 1$ and $d_\wedge = (5/2) \cdot n^\epsilon$. The algorithm outputs an affine subspace W' of co-dimension at most

$$\frac{m_\wedge}{d_\oplus} + \frac{d_\oplus \cdot m_\oplus}{d_\wedge} \leq (1/5) \cdot n^{1-\epsilon} + (2/5) \cdot n$$

such that in the restriction of C to W', every $\wedge$-gate in the middle layer has fan-in at most $d_\wedge = (5/2) \cdot n^\epsilon$. Since there are at most $m_\wedge = (1/5) \cdot n^{1-\epsilon}$ gates in the middle layer, it follows that there are at most $m_\wedge \cdot d_\wedge = n/2$ bottom $\oplus$-gates that influence the output of $C\restriction_{W'}$. By fixing values for these gates, we obtain a subspace W of dimension at least $(1/2 - (2/5) - o(1)) \cdot n > n/15$ such that $C\restriction_W$ is constant. Since $B(n) = 2^{n/15}$, it follows that $C\restriction_W \equiv 1$, and thus we can output any $w \in W$. ∎

3.4.5 Proof of Theorem 3.4.3

We now prove a more general version of Theorem 3.4.3, which depends on additional parameters; after stating this general version, we will spell out the parameter choices that yield Theorem 3.4.3. The proof relies on Lemma 3.2.5.

Proposition 3.4.9 (Theorem 3.4.3, parametrized version)**.** *For $m : \mathbb{N} \to \mathbb{N}$ and $b : \mathbb{N} \to [0, \frac{1}{2}]$, let $\mathcal{C}$ be the class of $\oplus \wedge \oplus$ circuits over n input bits with $m = m(n)$ $\wedge$-gates that accept all but a $b(n)$ fraction of their inputs. For any $d \geq 2$ and $c' \leq 2^d/m$, let $\mathcal{P}_d^{c'}$ be the class of polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ of degree d that accept all but a $c' \cdot \left(m \cdot 2^{-d}\right)$ fraction of their inputs.*

Let d be an integer such that $\log(m) < d \leq \min\{\log(m) + \log(1/b(n)), n\}$, and let $2 < c' \leq 2^d/m$ be a real number. Assume that there exists a hitting-set generator G with density more than $(2/c') + m \cdot 2^{-d}$ for $\mathcal{P}_d^{c'}$. Then, G is a hitting-set generator for $\mathcal{C}$.

To obtain parameters as in Theorem 3.4.3, let $\epsilon = \epsilon(n)$ such that $2^{-n/2} \leq \epsilon \leq 1/8$, and let $m = m(n) \leq 2^{n/2}$. For $d = \lfloor \log(m) + \log(1/\epsilon) \rfloor \leq n$ and $c' = 4 \leq 2^d/m$, assume that there exists a hitting-set generator G for the class $\mathcal{P}_d^{c'}$ with density $1/2 + 2 \cdot \epsilon \geq (2/c') + m \cdot 2^{-d}$. Then, Proposition 3.4.9 asserts that G is a hitting-set generator for the class of $\oplus \wedge \oplus$ circuits with m $\wedge$-gates that accept all but $\epsilon \cdot 2^n$ of their inputs.

Proof. Let $C : \{0,1\}^n \to \{0,1\}$ be a $\oplus \wedge \oplus$ circuit with m $\wedge$-gates that accepts all but a $b(n)$ fraction of its inputs. We will show how to randomly compute C by a

distribution that is typically in the class $\mathcal{P}_d^{c'}$, and then rely on Lemma 3.2.5 to deduce that any sufficiently dense hitting-set generator for $\mathcal{P}_d^{c'}$ also hits C.

The distribution over polynomials is obtained using Razborov's approximating polynomials method [Raz87]. Our goal is to randomly replace each $\wedge$-gate g that has fan-in more than d with a polynomial $g' : \{0,1\}^n \to \{0,1\}$ of degree d such that for every fixed input $x \in \{0,1\}^n$ it holds that $g(x) = g'(x)$ with probability at least $1 - 2^{-d}$. To this purpose, given $g(x) = \wedge_{j=1}^{k} L_j(x)$, where $k > d$ and the L_j's are linear functions, we randomly choose d subsets $S_1, ..., S_d \subseteq [k]$, and replace g with the $\mathbb{F}_2$-polynomial $g'(x) = \Pi_{i=1}^{d} \left(1 + \sum_{j \in S_i} (L_i(x) + 1)\right)$. [23]

The above yields a random polynomial $p : \mathbb{F}_2^n \to \mathbb{F}_2$ of degree at most d such that for every fixed $x \in \{0,1\}^n$ it holds that $\Pr[p(x) = C(x)] \geq 1 - m \cdot 2^{-d}$. The expected fraction of unsatisfying inputs for p is at most $2m \cdot 2^{-d}$; this is because

$$\begin{aligned}
\mathbb{E}_p \left[\Pr_x [p(x) = 0] \right] &= \mathbb{E}_x \left[\Pr_p [p(x) = 0] \right] \\
&\leq \Pr_x[C(x) = 0] + \Pr_x[C(x) = 1] \cdot \max_x \left\{ \Pr_p[p(x) \neq C(x)] \right\} \\
&\leq b(n) + m \cdot 2^{-d} \,,
\end{aligned}$$

and since $d \leq \log(m) + \log(1/b(n))$ we have that $m \cdot 2^{-d} \geq b(n)$. Thus, the probability that the fraction of unsatisfying inputs for p is more than $c' \cdot \left(m \cdot 2^{-d}\right)$ is at most $2/c'$.

Thus, there exists a distribution that is $(1 - 2/c')$-typically in $\mathcal{P}_d^{c'}$ and that rejects every $x \notin C^{-1}(1)$ with probability at least $1 - m \cdot 2^{-d}$. Now, let $\mathbf{w}$ be the output distribution of a hitting-set generator with density more than $\delta \stackrel{\text{def}}{=\!=} (2/c') + m \cdot 2^{-d}$ for $\mathcal{P}_d^{c'}$. Relying on Lemma 3.2.5 with $G = C^{-1}(1)$ and $\epsilon_1 = m \cdot 2^{-d}$ and $\epsilon_2 = 2/c'$ and $\epsilon_3 = \delta$, we deduce that

$$\Pr[C(\mathbf{w}) = 1] > \delta - m \cdot 2^{-d} - (2/c') = 0 \,,$$

which concludes the proof. ■

3.5 Linear threshold circuits and $\mathcal{ACC}^0$

3.5.1 The main results

In this section we focus on the circuit classes $\mathcal{TC}^0$ and $\mathcal{ACC}^0$. Recall that $\mathcal{ACC}^0$ is the class of constant-depth circuit families with polynomially-many gates that can compute the $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}$ functions as well as the MOD_m function, for any fixed integer $m \in \mathbb{N}$ (i.e., the Boolean function that evaluates to one if and only if the sum of

[23] Using the standard analysis, if $g(x) = 1$, then $L_j(x) = 1$ for all $j \in [k]$, which implies that $g'(x) = 1$ with probability one; and if $g(x) = 0$, then for every $i \in [d]$, with probability $1/2$ over choice of S_i it holds that $\sum_{j \in S_i} (L_i(x) + 1) = 1$, which implies that $g'(x) = 0$ with probability $1 - 2^{-d}$.

its inputs is zero modulo m). Also recall that $\mathcal{TC}^0$ is the class of constant-depth circuit families with polynomially-many gates that can compute the majority function,[24] which extends $\mathcal{ACC}^0$. (The fact that $\mathcal{ACC}^0 \subseteq \mathcal{TC}^0$ follows since $\mathcal{TC}^0$ can compute any symmetric function, because majority can be used to compute the indicator function for any particular Hamming weight of its input.)

A conjecture of Barrington [Bar89, Sec. 7] asserts that $\mathcal{ACC}^0 \neq \mathcal{TC}^0$; that is, the conjecture asserts that MAJ requires $\mathcal{ACC}^0$ circuits of super-polynomial size. Nevertheless, as far as we are aware of, there are currently no known super-linear lower bounds for computing MAJ by $\mathcal{ACC}^0$ circuits. The best known lower bounds for $\mathcal{ACC}^0$ circuits are against functions in non-deterministic quasipolynomial time (see [Wil11; MW18; Che19; CR20]). Inspired by the latter lower bounds, one of the prominent current challenges in complexity theory is the attempt to prove similar lower bounds for $\mathcal{TC}^0$. Even after extensive efforts during the last few decades (and with renewed vigor recently), the best known lower bounds for $\mathcal{TC}^0$ assert the existence of functions in $\mathcal{P}$ that require $\mathcal{TC}^0$ circuits with a *slightly super-linear* number of wires, or with a *linear* number of gates.

Since derandomization algorithms imply lower bounds in general, an appealing approach to prove lower bounds for $\mathcal{TC}^0$ is to construct derandomization algorithms for this class. Accordingly, an intensive recent effort has been devoted to constructing deterministic algorithms either for satisfiability of $\mathcal{TC}^0$ or for derandomization of $\mathcal{TC}^0$.[25] Satisfiability algorithms with non-trivial running time have been constructed for $\mathcal{TC}^0$ circuits of depth two, and for certain "structured subclasses" of $\mathcal{TC}^0$ (see [IPS13; Wil14b; AS15; SST+16; Tam16]). In addition, following an intensive effort to construct pseudorandom generators for a single *linear threshold function* [DGJ+10; RS10; GOW+10; KRS12; MZ13; Kan11; Kan14; KM15; GKM15] (i.e., a single "gate"), a first step towards derandomizing $\mathcal{TC}^0$ *circuits* was very recently undertaken by Servedio and Tan [ST17b], who considered the problem of derandomizing $\mathcal{TC}^0$ circuits of *depth two*.[26]

3.5.1.1 The main results for $\mathcal{TC}^0$

Loosely speaking, we show two complementary results: The first is a quantified derandomization algorithm for $\mathcal{TC}^0$ circuits of depth d with $n^{1+c^{-d}}$ wires and a subexponential $B(n)$, where $c > 1$ is some large constant; and the second results asserts that such derandomization with a *smaller value of the constant* $c > 1$ (i.e., derandomization of depth-d $\mathcal{TC}^0$ circuits with $n^{1+(c')^{-d}}$ wires and a subexponential $B(n)$, for some c'

[24] An alternative common definition for $\mathcal{TC}^0$ allows the gates to only compute any linear threshold function; see Section 2.3.1.3 for details.

[25] To get lower bounds it suffices to construct algorithms for derandomization with *one-sided error* (see, e.g., [Wil13]), which are weaker than satisfiability algorithms.

[26] Their manuscript is currently unpublished. For every $\epsilon > 0$, they constructed a pseudorandom generator that $1/\mathrm{poly}(n)$-fools any *depth-2 linear threshold circuit with at most* $n^{2-\epsilon}$ *wires,* using a seed of length $n^{1-\delta}$, where $\delta = \delta_\epsilon > 0$ is a small constant that depends on ϵ. This yields a derandomization of depth-2 linear threshold circuits with $n^{2-\epsilon}$ wires in time $2^{n^{1-\Omega(1)}}$.

that is smaller than the constant c in the first result) would suffice to solve CAPP for $\mathcal{TC}^0$, and hence deduce that $\mathcal{NEXP} \not\subset \mathcal{TC}^0$. Thus, intuitively, the gap between what we unconditionally know and what would suffice to solve CAPP for $\mathcal{TC}^0$ (and deduce corresponding lower bounds) boils down to the precise constant $c > 1$ in the bound $n^{1+c^{-d}}$ on the number of wires. Details follow.

Our first result is a *quantified derandomization algorithm* for $\mathcal{TC}^0$ circuits with a slightly super-linear number of wires. In fact, our algorithm works not only for $\mathcal{TC}^0$, but also for the class of linear threshold circuits: While in $\mathcal{TC}^0$ circuits each gate computes the majority function, in linear threshold circuits each gate computes a linear threshold function (i.e., a function of the form $g(x) = \operatorname{sgn}\left(\sum_{i\in[n]} w_i \cdot x_i - \theta\right)$, for $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$; see Section 2.3.1.3 for definitions). Towards stating this first result, denote by $\mathcal{C}_{n,d,w}$ the class of linear threshold circuits over n input bits of depth d and with at most w wires.

Theorem 3.5.1 (quantified derandomization of linear threshold circuits)**.** *There exists a deterministic algorithm that, when given as input a circuit $C \in \mathcal{C}_{n,d,n^{1+2^{-10d}}}$, runs in time $n^{O(\log\log(n))^2}$, and satisfies the following:*

1. *If C accepts all but at most $B(n) = 2^{n^{1-1/5d}}$ of its inputs, then the algorithm accepts C.*
2. *If C rejects all but at most $B(n) = 2^{n^{1-1/5d}}$ of its inputs, then the algorithm rejects C.*

Observe that as d grows larger, the algorithm in Theorem 3.5.1 solves a more difficult derandomization task (since $B(n)$ is larger), but only has to handle circuits with fewer wires (i.e., $n^{1+\exp(-d)}$). Also note that the algorithm in Theorem 3.5.1 is "*white-box*": That is, the algorithm gets as input an *explicit description* of a *specific* linear threshold circuit C, and uses this description when estimating the acceptance probability of C. [27] The actual algorithm that we construct works for a more general parameter regime, which exhibits a trade-off between the number $B(n) = 2^{n^{1-\delta}}$ of exceptional inputs for C and the number $n^{1+\delta\cdot\exp(-d)}$ of wires of C (see Theorem 3.5.7 for a precise statement).

The limitation on the number of wires of C in Theorem 3.5.1 (i.e., $n^{1+\exp(-d)}$) essentially *matches the best-known lower bounds for linear threshold circuits*, which were proved by Impagliazzo, Paturi, and Saks [IPS97]. This is no coincidence: Our algorithm construction follows a common theme in the design of circuit-analysis algorithms (e.g., derandomization algorithms or algorithms for satisfiability), which is the conversion of techniques that underlie lower bound proofs into algorithmic techniques. Specifically, we observe that certain proof techniques for *average-case lower bounds* for a circuit class $\mathcal{C}$ can be used to obtain algorithmic techniques for *quantified derandomization* of $\mathcal{C}$. To construct the algorithm in Theorem 3.5.1, we leverage the techniques underlying the recent proof of Chen, Santhanam, and Srinivasan [CSS16] of average-case

[27] The algorithm in Theorem 3.5.1 works in any reasonable model of explicitly representing linear threshold circuits; see Section 2.3.1.3 for a brief discussion.

lower bounds for linear threshold circuits. A high-level description of our algorithm appears in Section 3.5.2.1.

Loosely speaking, our second main result is that a quantified derandomization algorithm for $\mathcal{TC}^0$ circuits of depth d and size $n^{1+c^{-d}}$ with a sub-exponential $B(n)$ would yield a corresponding algorithm for standard derandomization of all $\mathcal{TC}^0$, and hence imply that $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$. Moreover, the hypothesis in the foregoing statement refers to a *fixed* constant $c \approx 1.61$; that is, the hypothesis refers to $\mathcal{TC}^0$ circuits of a specific size but nevertheless implies lower bounds for all of $\mathcal{TC}^0$.

Being more specific, the size of the circuits that the hypothesis refers to is "just beyond" the size required to compute the *parity* function, against which our best known $\mathcal{TC}^0$ lower bounds hold. Specifically, let $\phi \geq \frac{1+\sqrt{5}}{2}$ such that for any $d \geq 2$, parity can be computed by $\mathcal{TC}^0$ circuits of depth d and size $n^{1+O(\phi^{-d})}$ (the bound on ϕ is due to the construction of Paturi and Saks [PS94], following [BBL92]). Then, the hypothesis in the following theorem refers to $\mathcal{TC}^0$ circuits of depth d and $n^{1+c^{-d}}$ wires, where $c > 1$ can be any fixed constant smaller than ϕ.

Theorem 3.5.2 (a bootstrapping result for derandomization of $\mathcal{TC}^0$)**.** *Let $c > 1$ be any fixed constant smaller than ϕ (e.g., $c = 1.61$). Assume that for every sufficiently large $d \in \mathbb{N}$ there exists an algorithm for quantified derandomization of $\mathcal{TC}^0$ circuits with $n^{1+c^{-d}}$ wires and with $B(n) = 2^{n^{1-c^{-d}}}$ that runs in time $2^{n^{o(1)}}$. Then, there exists an algorithm for standard derandomization of $\mathcal{TC}^0$ that runs in time $2^{n^{o(1)}}$.*

Using known results that show that standard derandomization of $\mathcal{TC}^0$ implies lower bounds for $\mathcal{TC}^0$ (i.e., Williams' "algorithmic method", applied to the special case of $\mathcal{TC}^0$; see [Wil13; SW13; BSV14]), we deduce that quantified derandomization as in the hypothesis of Theorem 3.5.2 implies lower bounds for $\mathcal{TC}^0$.

Corollary 3.5.3 (quantified derandomization of sparse $\mathcal{TC}^0$ implies lower bounds for $\mathcal{TC}^0$)**.** *Assume that there exists an algorithm as in the hypothesis of Theorem 3.5.2. Then, $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$.*

The algorithm for quantified derandomization from Theorem 3.5.1 works when both the size of the circuit and the number of exceptional inputs are slightly smaller than in the hypothesis in Theorem 3.5.2 and Corollary 3.5.3. Nevertheless, the running time of the algorithm is $n^{\text{poly}\log\log(n)}$, whereas the running time of the algorithm in the hypothesis in Corollary 3.5.3 may be much larger (i.e., $2^{n^{o(1)}}$). Moreover, we can further relax the requirements from the algorithm in the hypothesis in Corollary 3.5.3, relying on known relaxations of Williams' algorithmic method (see Corollary 3.5.34 for details).

A main technical result that underlies Theorem 3.5.2 is a construction of uniform $\mathcal{TC}^0$ circuits of depth $d \geq 7$ and size $n^{1+\exp(-d)}$ that compute all the outputs of a seeded extractor on a given input. Specifically, we prove that there exists an extractor $\texttt{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ for min-entropy $k = n^{1-\exp(-d)}$ with $m = n^{\exp(-d)}$ output bits and seed length $t = (1+\exp(-d)) \cdot \log(n)$ such that the mapping of

$x \in \{0,1\}^n$ to $\{\mathtt{Ext}(x,z)\}_{z\in\{0,1\}^t}$ is computable by a uniform depth-d circuit of size $n^{1+\exp(-d)}$ (see Theorem 3.5.4). Note that the latter circuit has $2^t \cdot m = n^{1+\exp(-d)}$ output bits and only $n^{1+\exp(-d)}$ wires; thus, the circuit essentially performs an efficient "batch computation" of all the outputs of the extractor (i.e., the outputs corresponding to all seeds) on a given input $x \in \{0,1\}^n$. This construction relies, in turn, on a construction of uniform $\mathcal{TC}^0$ circuits of depth d and size $n^{1+\exp(-d)}$ that compute a code with constant relative distance and a slightly sub-constant rate (see Theorem 3.5.6).

The special case of depth-2 $\mathcal{TC}^0$ circuits We also construct an alternative quantified derandomization algorithm for the special case of linear threshold circuits of *depth two*. Specifically, we construct a pseudorandom generator with seed length $\tilde{O}(\log(n))$ for the class of depth-2 linear threshold circuits with $n^{3/2-\Omega(1)}$ wires that either accept all but $B(n) = 2^{n^{\Omega(1)}}$ of their inputs or reject all but $B(n)$ of their inputs. This result is not a corollary of Theorem 3.5.1, and is incomparable to the pseudorandom generator of Servedio and Tan [ST17b].

The precise result statement and proof appear in Section 3.5.5. The generator construction is obtained by leveraging the techniques of Kane and Williams [KW16] for average-case lower bounds for linear threshold circuits of depth two.

3.5.1.2 The main results for $\mathcal{ACC}^0$

We consider the problem of derandomization of $\mathcal{ACC}^0$ with "one-sided error"; that is, the problem of deterministically distinguishing $\mathcal{ACC}^0$ circuits with acceptance probability one from $\mathcal{ACC}^0$ circuits with acceptance probability at most half. The best currently-known algorithm for this problem is the *satisfiability* algorithm of Williams [Wil11], which runs in time $2^{n-n^{\Omega(1)}}$ (and distinguishes circuits with acceptance probability one from circuits with acceptance probability less than one).

Nevertheless, we do have much faster algorithms for *quantified* derandomization of limited subclasses of $\mathcal{ACC}^0$. Specifically, we currently have *polynomial-time* algorithms for quantified derandomization of "structured subclasses" of $\mathcal{AC}^0[\oplus]$ of depth three and small super-linear size (e.g., size $O(n)$ or $n + n^{\Omega(1)}$) with a subexponential $B(n)$; for precise details see [GW13, Sec. 6] and Section 3.4. We show that improving these results to quantified derandomization of $\mathcal{ACC}^0$ circuits of depth $d \geq 5$ and size $n^{1+\gamma_d}$ with a subexponential $B(n) \approx 2^{n^{1-\gamma_d/8}}$, where $\gamma_d > 0$ can be any sufficiently small constant, would imply corresponding algorithms for standard derandomization of $\mathcal{ACC}^0$ with "one-sided error". See Section 3.5.4.4 for further details.

3.5.2 Proof overviews

3.5.2.1 Proof overview for Theorem 3.5.1

The high-level strategy of the quantified derandomization algorithm follows the strategy suggested by Goldreich and Wigderson [GW14]. Specifically, given a circuit

$C : \{-1,1\}^n \to \{-1,1\}$, the algorithm deterministically finds a set $S \subseteq \{-1,1\}^n$ of size $|S| \gg B(n)$ on which the circuit C simplifies; that is, C agrees with a function from some "simple" class of functions on almost all points in S. If C accepts all but $B(n)$ of its inputs, then the acceptance probability of $C\restriction_S$ will be very high, and similarly, if C rejects all but $B(n)$ of its inputs, then the acceptance probability of $C\restriction_S$ will be very low. The algorithm then distinguishes between the two cases, by enumerating the seeds of a pseudorandom generator for the "simple" class of functions.[28]

Our starting point in order to construct a deterministic algorithm that finds a suitable set S is the recent proof of average-case lower bounds for sparse linear threshold circuits by Chen, Santhanam, and Srinivasan [CSS16]. Their proof is based on a *randomized* "whitebox" algorithm that gets as input a linear threshold circuit with depth d and $n^{1+\epsilon}$ wires, and restricts all but $n^{1-\epsilon\cdot\exp(d)}$ of the variables such that the restricted circuit can be approximated by a single linear threshold function. Thus, if we are able to modify their algorithm to a deterministic one, we will obtain a quantified derandomization algorithm with the parameters asserted in Theorem 3.5.1 (i.e., if $\epsilon = \exp(-d)$, then $B(n) \approx |S|/10 > 2^{n^{1-1/5d}}$). [29]

Converting the randomized restriction algorithm into a deterministic algorithm poses several challenges, which will be our focus in this overview. Let us first describe the original algorithm, in high-level. The algorithm iteratively reduces the depth of the circuit. In each iteration it applies a random restriction that keeps every variable alive with probability $p = n^{-\Omega(1)}$, and otherwise assigns a random value to the variable. The main structural lemma of [CSS16] asserts that such a random restriction turns any LTF to be very biased (i.e., $\exp(-n^{\Omega(1)})$-close to a constant function), with probability $1 - n^{-\Omega(1)}$. Hence, after applying the restriction, most gates in the bottom layer of the circuit become very biased, and the fan-in of the rest of the gates in the bottom layer significantly decreases (i.e., we expect it to reduce by a factor of $p = n^{-\Omega(1)}$). The algorithm replaces the very biased gates with the corresponding constants, thereby obtaining a circuit that *approximates* the original circuit (i.e., the two circuits agree on all but $2^{-n^{\Omega(1)}}$ of the inputs); and in [CSS16] it is shown that the algorithm can afterwards fix relatively few variables such that the fan-in of each gate that did not become very biased decreases to be at most one (such a gate can be replaced by a variable or a constant). Thus, if the circuit C_i in the beginning of the iteration was of depth i, we obtain a circuit C_{i-1} of depth $i-1$ that approximates C_i.

One obvious challenge in converting the randomized restriction algorithm into a deterministic algorithm is "derandomizing" the main structural lemma; that is, we need to construct a pseudorandom distribution of restrictions that turns any LTF to be very biased, with high probability. The second challenge is more subtle, and arises

[28] The actual algorithm that we construct finds a *collection* of sets S such that most sets in the collection are both large and "simplify" C (i.e., $C\restriction_S$ is "simple"); for simplicity, in the overview we ignore this point.

[29] This approach follows the well-known theme of "leveraging" techniques from lower bound proofs to algorithmic techniques, and in particular to techniques for constructing circuit-analysis algorithms; see, e.g., [LMN93; San10; Bra10; IMZ12; ST12; IMP12; BIS12; GMR13; TX13; CKK+15; ST17b; ST17a]. We also mention that in [CSS16, Sec. 5] their randomized restriction algorithm is used to construct a *randomized* algorithm for *satisfiability* of sparse linear threshold circuits.

when applying pseudorandom restrictions several times, sequentially, while replacing biased gates by constants each time. In the following two subsections we describe these two challenges are our solutions, respectively.

Derandomizing the main structural lemma of [CSS16]. Let $\Phi = (w, \theta)$ be an LTF over n input bits, and consider a random restriction ρ that keeps each variable alive with probability $p = n^{-\Omega(1)}$. Peres' theorem implies that the expected distance of $\Phi\restriction_\rho$ from a constant function is approximately $\sqrt{p}$ (see, e.g., [O'D14, Sec. 5.5]).[30] A natural question is whether we can prove a concentration of measure for this distribution. As an illustrative example, consider the majority function $MAJ(x) = \mathrm{sgn}(\sum_{i\in[n]} x_i)$; for any $t \geq 1$, with probability roughly $1 - t \cdot \sqrt{p}$ it holds that $MAJ\restriction_\rho$ is $\exp(-t^2)$-close to a constant function (see Fact 3.5.9). The main structural lemma in [CSS16] asserts that a similar statement indeed holds for *any LTF* Φ; specifically, they showed that with probability at least $1 - p^{\Omega(1)}$ it holds that $\Phi\restriction_\rho$ is $\exp(-p^{-\Omega(1)})$-close to a constant function.

We construct a distribution over restrictions that can be efficiently sampled using $\tilde{O}(\log(n))$ random bits such that for any LTF Φ and any $t \geq p^{-1/8}$, with probability at least $1 - \tilde{O}(t^2) \cdot \sqrt{p}$ it holds that $\Phi\restriction_\rho$ is $\exp(-t^2)$-close to a constant function. (The actual statement that we prove is more general; see Proposition 3.5.14 for precise details.) Indeed, this is both an "almost-full derandomization" of the lemma of [CSS16] as well as a refinement of the quantitative bound in the lemma.

The original proof of [CSS16] relies on a technical case analysis that is reminiscent of other proofs that concern LTFs, and is based on the notion of a *critical index* of a vector $w \in \mathbb{R}^n$ (they refer to the ideas underlying such analyses as "the structural theory of linear threshold functions"; see, e.g., [Ser07; DGJ+10], and Definitions 2.3.1 and 2.3.2). In each case, the main technical tools that are used are concentration and anti-concentration theorems for random weighted sums (i.e., Hoeffding's inequality and the Berry-Esséen theorem, respectively), which are used to bound the probability that several specific random weighted sums that are related to the restricted function $\Phi\restriction_\rho$ fall in certain intervals.

To derandomize the original proof, an initial useful observation is the following. We say that a distribution $\mathbf{z}$ over $\{-1,1\}^n$ is ϵ-pseudorandomly concentrated if for any $w \in \mathbb{R}^n$ and any interval $J \subseteq \mathbb{R}$, the probability that $\langle w, \mathbf{z}\rangle$ falls in J is ϵ-close to the probability that $\langle w, \mathbf{u}_n\rangle$ falls in J (where $\mathbf{u}_n$ is the uniform distribution over $\{-1,1\}^n$). In particular, the Berry-Esséen theorem and Hoeffding's inequality approximately hold for pseudorandom sums $\langle w, \mathbf{z}\rangle$ when $\mathbf{z}$ is pseudorandomly concentrated. The observation is that being ϵ-pseudorandomly concentrated is essentially equivalent to being ϵ-pseudorandom for LTFs (see Claim 2.4.19).[31] In particular, if a distribution $\mathbf{z}$ over $\{-1,1\}^n$ is chosen using the pseudorandom generator of Gopalan,

[30] Peres' theorem is usually phrased in terms of the noise sensitivity of Φ, but the latter is proportional to its expected bias under a random restriction; for further details see [CSS16, Prop. 8].

[31] This observation was communicated to us by Rocco Servedio, and is attributed to Li-Yang Tan.

Kane, and Meka [GKM15] for LTFs, which has seed length $\tilde{O}(\log(n/\epsilon))$, then $\mathbf{z}$ is ϵ-pseudorandomly concentrated.

The main part in the proof of the derandomized lemma is a (non-trivial) modification of the original case analysis, in order to obtain an analysis in which all claims hold under a suitably-chosen pseudorandom distribution of restrictions. Since this part of the proof is quite technical and low-level, we defer its detailed description to Section 3.5.3.1. However, let us mention that our pseudorandom distribution itself is relatively simple: We first choose the variables to keep alive such that each variable is kept alive with probability approximately $p = n^{-\Omega(1)}$, and the choices are $O(1)$-wise independent; and then we independently choose values for the fixed variables, using the generator of [GKM15] with error parameter $\epsilon = 1/\mathrm{poly}(n)$. We also note that it is suprising that in our setting the case analysis can be modified in order to obtain an "almost-full derandomization" (i.e., seed length $\tilde{O}(\log(n))$), since previous derandomizations of similar case analyses regarding LTFs for different settings required much larger seed for error $\epsilon = n^{-\Omega(1)}$ (see [DGJ+10]).

Preserving the closeness of the circuit to its approximations. Recall that in order to simplify a linear threshold circuit into a single LTF we will iteratively apply the pseudorandom restrictions that were described in the previous section, in order to reduce the depth of the circuit. Specifically, in each iteration we will replace the "current" circuit C_i by a circuit C_{i-1} that agrees with C_i on almost all inputs in the subcube of the n living variables (i.e., the circuits disagree on at most $2^{n-n^{\Omega(1)}}$ inputs). The main "approximation" step in constructing C_{i-1} from C_i is replacing very biased gates by corresponding constants.

The potential problem that is our current focus arises from the fact that in subsequent iterations we will fix almost all of these n living variables, such that only $n^{1-\Omega(1)}$ variables will remain alive. Thus, we have no guarantee that C_i and C_{i-1} will remain close after additional restrictions in subsequent iterations; in particular, C_i and C_{i-1} might disagree on *all* of the inputs in the subcube of living variables in the end of the entire process. Of course, this is very unlikely to happen when values for fixed variables are chosen uniformly, but we need to construct a *pseudorandom* distribution of restrictions such that the approximation of each C_i by C_{i-1} is likely to be maintained throughout the process.

We will in fact choose each restriction ρ such that the following holds: For each gate Φ that was replaced by a constant $\sigma \in \{-1, 1\}$, with probability $1 - \frac{1}{\mathrm{poly}(n)}$ over choice of restriction ρ it holds that $\Phi\restriction_\rho$ is still $\frac{1}{\mathrm{poly}(n)}$-close to σ (i.e., $\Pr_x[\Phi\restriction_\rho(x) \neq \sigma] \leq \frac{1}{\mathrm{poly}(n)}$). Specifically, we prove that if an LTF Φ is, say, n^{-20}-close to a constant σ, and a restriction ρ is chosen such that the distribution of values for the fixed variables is n^{-10}-*pseudorandom for LTFs*, then with probability $1 - n^{-10}$ it holds that $\Phi\restriction_\rho$ is n^{-10}-close to σ (see Lemma 3.5.16).[32]

[32]Since each gate is initially $\exp(-n^{\Omega(1)})$-close to a constant, we can afford a constant number of losses in the polynomial power in the "closeness" parameter throughout the execution of the restriction

Accordingly, whenever we fix variables in our algorithm, we will choose the values for the fixed variables according to a distribution that is $(1/\mathrm{poly}(n))$-pseudorandom for LTFs. By the statement above, after each such fixing, every gate that was close to a constant (and was replaced, at some point, by that constant) remains close to the corresponding constant even in the subcube of living variables, with high probability. By union-bounding over all gates (in all the $O(1)$ circuits obtained in each of the iterations), with high probability the final circuit is close to the initial circuit in the final subcube of living variables.

In Section 3.5.3.2 we include a simple proof of the fact that biased LTFs remain biased when variables are fixed according to a distribution that is pseudorandom for LTFs. This simple proof was suggested by an anonymous reviewer, and follows an approach of Ajtai and Wigderson [AW85]. Our original proof of this fact was more complicated, but uses techniques that may be of independent interest; we therefore include the original proof in Appendix 3.5.6.1.

3.5.2.2 Proof overview for Theorem 3.5.2

Towards presenting the proof of our second main result, we say that a circuit family $\{C_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ of constant depth d is extremely sparse if for any sufficiently large $n \in \mathbb{N}$, the number of wires in C_n is at most $n^{1+c^{-d}}$, for some constant $c > 1$.

We prove Theorem 3.5.2 by showing an efficient reduction of *standard* derandomization of *all* of $\mathcal{TC}^0$ to *quantified* derandomization of *extremely sparse* $\mathcal{TC}^0$. Specifically, we construct an algorithm that is given a $\mathcal{TC}^0$ circuit C of depth d_0 over m input bits, and outputs an extremely sparse $\mathcal{TC}^0$ circuit C' of depth $d > d_0$ over $n = \mathrm{poly}(m)$ input bits such that if C accepts (resp., rejects) at least $2/3$ of its inputs then C' accepts (resp., rejects) all but $B(n) = 2^{n^{1-\exp(-d)}}$ of its inputs. The circuit C' will use its input in order to sample inputs for C by a seeded extractor (equivalently, by an averaging sampler), and output the majority of the evaluations of C on these inputs.

The main technical challenge that underlies this approach is constructing an extractor $\mathtt{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ such that the mapping of input $x \in \{0,1\}^n$ to the 2^t m-bit outputs of the extractor on all seeds can be computed by *uniform extremely sparse $\mathcal{TC}^0$ circuits*. This is indeed the technical result that underlies Theorem 3.5.2, and it improves a previous construction from [**tell18**], in which the circuit had $n^{1+O(1/d)}$ wires. As mentioned in the introduction, our circuit has $2^t \cdot m = n^{1+\exp(-d)}$ output bits but it uses only $n^{1+\exp(-d)}$ wires, and so to construct it we need to perform an efficient "batch computation" of the extractor on all seeds.

Theorem 3.5.4 (an extractor in extremely sparse $\mathcal{TC}^0$)**.** *There exists a polynomial-time algorithm that gets as input 1^n and a constant $d \geq 7$, and outputs a $\mathcal{TC}^0$ circuit $C : \{0,1\}^n \to (\{0,1\}^m)^s$ of depth d and size $n^{1+\exp(-d)}$ that satisfies the following: The function $\mathtt{Ext}(x,i) = C(x)_i$ is a (k,ϵ)-extractor for min-entropy $k = n^{1-\exp(-d)}$, with output length $m = n^{\exp(-d)}$, seed length $t = \log(s) = (1+\exp(-d)) \cdot \log(n)$ and error $\epsilon = 1/m$.*

algorithm.

Recall that Theorem 3.5.2 asserts that standard derandomization reduces to quantified derandomization of circuits with $n^{1+c^{-d}}$ wires, for any $c < \phi = \frac{1+\sqrt{5}}{2}$. By carefully accounting for the parameters in our construction, the number of wires in the circuit from Theorem 3.5.4 is indeed $n^{1+c^{-d}}$, for any $c < \phi$ (see Proposition 3.5.32). However, in the current high-level overview we will not care about the specific constant c.

As a first step, let us describe a *non-uniform* construction of $\mathcal{TC}^0$ circuits as in Theorem 3.5.4; that is, we will first show that such circuits *exist*, and later show that they can be constructed uniformly. The underlying extractor will be Trevisan's extractor [Tre01]. In order to compute the mapping $x \mapsto \{\mathtt{Ext}(x,z)\}_{z\in\{0,1\}^t}$ when $\mathtt{Ext}$ is Trevisan's extractor, we first need to encode x to a codeword $\bar{x}$ of a *balanced code* (i.e., every codeword has relative Hamming weight close to $1/2$), and then determine the $s = 2^t$ outputs of the extractor as projections of the bits of $\bar{x}$, according to appropriate *combinatorial designs* (in the sense of Nisan and Wigderson [NW94]). Note that, crucially, to compute all the outputs of the extractor on a given input $x \in \{0,1\}^n$, we only need to encode $x \mapsto \bar{x}$ *once*, and all the outputs are projections of bits of $\bar{x}$.

Our non-uniform circuit first encodes $x \in \{0,1\}^n$ to a codeword $\hat{x} \in \{0,1\}^{O(n)}$ of a code with constant rate and *constant relative distance*. Gál *et al.* [GHK+13] showed that there exist depth-two circuits with *only parity gates* that can compute such a code using only $n \cdot \operatorname{poly}\log(n)$ wires. To convert such a "parity-gates" circuit into a $\mathcal{TC}^0$ circuit, we replace each parity gate g with fan-in n_g in by a $\mathcal{TC}^0$ circuit computing parity with depth d and $n_g^{1+\exp(-d)}$ wires, using the constructions of [BBL92; PS94]. The resulting $\mathcal{TC}^0$ circuit has depth $2d$, and its number of wires is at most $\sum_{g \text{ gate}} \left(n_g^{1+\exp(-d)}\right) < \left(\sum_g n_g\right)^{1+\exp(-d)} \leq (n \cdot \operatorname{poly}\log(n))^{1+\exp(-d)} = n^{1+\exp(-d)}$.

We now amplify the distance of the code from $\Omega(1)$ to approximately $1/2$, using the strategy of Naor and Naor [NN93]. Specifically, we map $\hat{x}$ to a new codeword $\bar{x}$ such that every bit of $\bar{x}$ corresponds to a walk of an appropriate length ℓ on an expander on $[|\hat{x}|] = [O(n)]$, and the bit in $\bar{x}$ is the parity of the bits of $\hat{x}$ encountered during this walk. Trevisan's extractor requires the distance to be $1/2 - \delta$, where $\delta \approx 1/m^2 = 1/n^{\exp(-d)}$; to get such a distance, it suffices for the walk to be of length $O(\log(1/\delta))$, and thus the length of $\bar{x}$ is $O(n \cdot \operatorname{poly}(1/\delta)) = n^{1+\exp(-d)}$ (assuming that $n = \operatorname{poly}(m)$ is sufficiently large). Since each bit in $\bar{x}$ is the parity of $O(\log(n))$ bits in $\hat{x}$, we can compute each bit in $\bar{x}$ by a depth-two circuit with $O(\log^2(n))$ wires, and the resulting circuit will be of depth $2d+2$ and $n^{1+\exp(-d)}$ wires.

Finally, the circuit outputs projections of $\bar{x}$ according to predetermined combinatorial designs. We will use *weak designs*, introduced by Raz, Reingold, and Vadhan [RRV02], which suffice for Trevisan's extractor. Using a specific construction of weak designs suitable for our parameter setting, in which the required min-entropy is $n^{1-\exp(-d)}$, there exist weak designs in a universe of size $t = (1+\exp(-d)) \cdot \log(n)$ (see Lemma 3.5.30). Thus we have that $s = 2^t = n^{1+\exp(-d)}$ as we wanted.

The remaining challenge is to turn the construction above into a *uniform* construction. Most parts of the construction are already uniform: The circuits for the distance

amplification step, the $\mathcal{TC}^0$ circuits for computing parity, and the weak designs, can all be constructed in polynomial time. Thus, the only "non-uniform" part is the sparse "parity-gates" circuit that computes a code with constant relative distance. Indeed, Gál *et al.* [GHK+13] posed the construction of a uniform "parity gate" circuit with parameters matching the ones of their non-uniform construction as an important open problem, noting that this problem is closely related to the long-standing open problems of explicitly constructing superconductors and unbalanced lossless expanders.

Fortunately, for our purposes we can afford a mild degradation in the parameters of the code construction. Specifically, our code does not need to have constant rate, since rate $1/n^{o(1)}$ also suffices for our purposes (because we map $\hat{x}$ to $\bar{x}$ of length $|\hat{x}|^{1+\exp(-d)}$ in the distance amplification step anyway). Also, the "parity gates" circuit that we start from does not need to have $n \cdot \text{poly}\log(n)$ wires, and a circuit with $n^{1+o(1)}$ wires also suffices for our purposes (because we convert each gate g into a $\mathcal{TC}^0$ circuit with $n_g^{1+\exp(-d)}$ wires anyway). Indeed, we construct a uniform "parity gates" circuit of depth two with $n \cdot \exp(\text{poly}\log\log(n))$ wires that computes a code with rate $1/n^{\exp(\text{poly}\log\log(n))}$ and constant relative distance (see Proposition 3.5.20).

The construction of this code (and circuit) appears in Section 3.5.4.1. Let us now describe this construction, at a high level. Since we will only use parity gates, our code will be linear; hence, to get constant relative distance, we only need to ensure that on any non-zero input $x \in \{0,1\}^n$ a constant fraction of the output gates will be set to one. The basic building-blocks in our construction are *range detectors*, as defined in [GHK+13]. Intuitively, these are functions that map any n-bit input with Hamming weight between $w/2$ and w (for some $w \in [n]$) to an m-bit output with Hamming weight between $.01m$ and $.99m$, using only *parity* gates. More formally:

Definition 3.5.5 (range detectors)**.** *An* (n,m,w,ℓ,h)-range detector *is a function* $D : \{0,1\}^n \to \{0,1\}^m$ *such that every output bit of* D *is a parity of input bits, and for any* $x \in \{0,1\}^n$ *with Hamming weight in* $[w/2,w]$ *it holds that the Hamming weight of* $D(x)$ *is in* $[\ell,h]$.

We are interested in constructing, for every $w < n$, an $(n,m,w,.01m,.99m)$-range detector, for some m, that only uses *few* wires (i.e., it only uses $n^{1+o(1)}$ wires).[33] In [GHK+13] it is shown that for every $w < n$, there *exist* $(n,m,w,.01m,.99m)$-range detectors with $m = O(w \cdot \log(n))$ that can be computed by a layer of parity gates with only $O(n \cdot \log(n))$ wires (see [GHK+13, Sec. 1.2 and Cor. 19]). For our uniform circuit, we need to *explicitly* construct range detectors; we do this relying on the explicit construction of *unbalanced lossless expanders* by Capalbo *et al.* [CRV+02]. Specifically, in Section 3.5.4.1.1 we rely on the latter expanders to explicitly construct $(n,m,w,.01m,.99m)$-range detectors with $m = O(w \cdot \exp(\text{poly}\log\log(n)))$ that can be computed by a layer of parity gates with $n \cdot \exp(\text{poly}\log\log(n))$ wires.

In the circuit that encodes our code, the layer above the input gates contains $\log(n)$ range detectors: For each $i = 1,...,\log(n)$ and $w_i = 2^i$, the layer contains an

[33] Indeed, for $w = \Omega(n)$ this is easy (e.g., for $.02n \leq w \leq .99n$, the identity mapping with $n = m$ yields a suitable range detector), and the main challenge is to handle smaller values of w.

$(n, m_i, w_i, .01m_i, .99m_i)$-range detector with $n \cdot \exp(\text{poly}\log\log(n))$ wires and $m_i = w_i \cdot \exp(\text{poly}\log\log(n))$ outputs. Thus, this layer contributes overall $n \cdot \exp(\text{poly}\log\log(n))$ wires to the circuit. We add to this construction a top layer of $n_0 = \max_{i \in [\log(n)]}\{m_i\} = \exp(\text{poly}\log\log(n))$ gates with the following property: If a constant fraction of the outputs of at least one range detector are set to one, then a constant fraction of the top gates *touch* a gate in the middle layer that is set to one.[34] To do so, for each range detector, we split the n_0 top gates into m_i blocks of n_0/m_i gates, and connect each output of the range detector to all top gates in the corresponding block.

Thus, for any non-zero $x \in \{0,1\}^n$, a constant fraction of the top gates touch a gate in the middle layer that is set to one. Now, note that each of the top n_0 gates has degree exactly $\log(n)$. Using another idea from [GHK+13], we replace each such gate g with $O(\log(n))$ parity gates that compute a good error-correcting code on the $\log(n)$ gates that fed to g. Hence, if g touched a middle gate that is set to one, then a constant fraction of the $O(\log(n))$ gates that replaced g are set to one. Thus, on any non-zero input $x \in \{0,1\}^n$, a constant fraction of the top gates will be set to one, and it follows that our linear code has constant relative distance. The number of gates in the top layer is $\hat{n} = n_0 \cdot O(\log(n)) = n \cdot \exp(\text{poly}\log\log(n))$, and each of them has degree $\log(n)$, yielding overall $n \cdot \exp(\text{poly}\log\log(n))$ wires between the middle layer and the top layer. Finally, replacing parity gates by $\mathcal{TC}^0$ circuits, we get the following:

Proposition 3.5.6 (uniform extremely sparse $\mathcal{TC}^0$ circuit for encoding an "almost–good" code; see Proposition 3.5.24)**.** *For every $d \geq 4$ there exists a uniform family of $\mathcal{TC}^0$ circuits of depth d that, for every $n \in \mathbb{N}$, encode a linear code $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ with constant relative distance, where $\hat{n} = n \cdot \exp(\text{poly}\log\log(n))$, using at most $n^{1+\exp(-d)}$ wires.*

Note that the use of $n^{1+\exp(-d)}$ wires in the circuit in Theorem 3.5.6 is unavoidable if we want the code to be linear (since computing even a single parity of $\Omega(n)$ bits in $\mathcal{TC}^0$ requires $n^{1+\exp(-d)}$ wires).

3.5.3 Proof of Theorem 3.5.1

Let us now state a more general version of Theorem 3.5.1 and prove it.

Theorem 3.5.7 (Theorem 3.5.1, restated)**.** *Let $d \geq 1$, let $\epsilon > 0$, and let $\delta = d \cdot 30^{d-1} \cdot \epsilon$. Then, there exists a deterministic algorithm that for every $n \in \mathbb{N}$, when given as input a circuit $C \in \mathcal{C}_{n,d,n^{1+\epsilon}}$, runs in time $n^{O(\log\log(n))^2}$, and for the parameter $B(n) = \frac{1}{10} \cdot 2^{n^{1-\delta}}$ satisfies the following:*

1. *If C accepts all but at most $B(n)$ of its inputs, then the algorithm accepts C.*
2. *If C rejects all but at most $B(n)$ of its inputs, then the algorithm rejects C.*

[34] This does not yet suffice for the full construction, since a top gate might touch an *even* number of gates in the middle layer that are set to one.

To obtain the specific parameters of Theorem 3.5.1 from Theorem 3.5.7, for any $d \geq 1$ let $\epsilon = \epsilon_d = 2^{-10d}$. Then, the algorithm from Theorem 3.5.7 works when the number of exceptional inputs of C is at most $B(n) = \frac{1}{10} \cdot 2^{n^{1-\delta}} > 2^{n^{1-1/5d}}$.

The proof of Theorem 3.5.7 is based on the existence of the following *pseudorandom restriction algorithm*. We will first prove Theorem 3.5.7 relying on the existence of the latter algorithm, and then construct the pseudorandom restriction algorithm itself.

Proposition 3.5.8 (pseudorandom restriction algorithm)**.** *Let $d \geq 1$, let $\epsilon > 0$ be a sufficiently small constant, and let $\delta = d \cdot 30^{d-1} \cdot \epsilon$. Then, there exists a polynomial-time algorithm that for every $n \in \mathbb{N}$, when given as input a circuit $C \in \mathcal{C}_{n,d,n^{1+\epsilon}}$ and a random seed of length $O(\log(n) \cdot (\log\log(n))^2)$, with probability at least $1 - n^{-\epsilon/2}$ satisfies the following:*

1. *The algorithm outputs a restriction $\rho \in \{-1, 1, \star\}^n$ that keeps at least $n^{1-\delta}$ variables alive.*
2. *The algorithm outputs an LTF $\Phi : \{-1,1\}^{\rho^{-1}(\star)} \to \{-1,1\}$ such that Φ is $1/10$-close to $C\restriction_\rho$ (i.e., $\Pr_{x \in \{-1,1\}^{\rho^{-1}(\star)}}[C(x) = \Phi(x)] \geq 9/10$).*

Let us now prove the main result (i.e., Theorem 3.5.7) relying on Proposition 3.5.8.

Proof of Theorem 3.5.7. We iterate over all seeds for the algorithm from Proposition 3.5.8. For each seed that yields both a restriction ρ that keeps at least $n^{1-\delta}$ variables alive and an LTF Φ over $\{-1,1\}^{\rho^{-1}(\star)}$, we estimate the acceptance probability of Φ up to an error of $\frac{1}{5}$; this is done by iterating over the seeds of the pseudorandom generator from Theorem 2.4.20 (instantiated with error parameter $1/5$). If for most of the seeds, our estimate of the acceptance probability of Φ is at least $\frac{3}{5}$, then we accept C; and otherwise we reject C. The running time of the algorithm is $2^{O(\log(n)\cdot(\log\log(n))^2)} = n^{O(\log\log(n))^2}$.

Recall that all but $O(n^{-\epsilon})$ of the seeds yield ρ and Φ such that ρ keep at least $n^{1-\delta} > \log(10 \cdot B(n))$ variables alive and such that Φ is $1/10$-close to $C\restriction_\rho$; we call such seeds *good* seeds. Now, if C accepts all but at most $B(n)$ inputs, then for every good seed, the acceptance probability of $C\restriction_\rho$ is at least $9/10$, and thus the acceptance probability of Φ is at least $\frac{4}{5}$, which implies that our estimate of the latter will be at least $3/5$. Thus, the algorithm will accept C. On the other hand, if C rejects all but at most $B(n)$ inputs, then by a similar argument for all good seeds it holds that the estimate of the acceptance probability of Φ will be at most $2/5$, and thus the algorithm will reject C. ■

We now prove Proposition 3.5.8 in three steps. The first step, in Section 3.5.3.1, will be to prove that a suitably-chosen pseudorandom restriction turns any single LTF to be very biased, with high probability. The second step, in Section 3.5.3.2, will leverage the first step to an algorithm that gets as input a linear threshold circuit, and applies pseudorandom restrictions to reduce the depth of the circuit by one layer. And the final step, in Section 3.5.3.3, will be to iterate the construction of the second step in order to prove Proposition 3.5.8.

3.5.3.1 Pseudorandom restrictions and a single LTF

As mentioned in the introduction, an illustrative example for the effects of restrictions on LTFs is the majority function $\Phi(x) = \text{sgn}(\sum_{i\in[n]} x_i)$. For $p \in (0,1)$, denote by $\boldsymbol{\mathcal{R}}_p$ the distribution of restrictions on n variables such that for every $i \in [n]$ independently it holds that the i^{th} variable remains alive with probability p, and is otherwise assigned a uniform random bit. Then, we have the following:

Fact 3.5.9 (a random restriction and the majority function)**.** *Let $\Phi(x) = \text{sgn}(\sum_{i\in[n]} x_i)$, and let $p = n^{-\Omega(1)}$. Then, for every $t \geq 1$, with probability at least $1 - O(t \cdot \sqrt{p})$ over $\rho \sim \boldsymbol{\mathcal{R}}_p$ it holds that $\Phi\restriction_\rho$ is t-imbalanced*

Proof. Let $I \subseteq [n]$ be the set of variables that ρ keeps alive. With probability $1 - \exp(-n^{\Omega(1)})$ it holds that $\|w_I\|_2 \in \sqrt{pn} \pm \sqrt{pn}/2$. Conditioned on $\|w_I\|_2 \leq 2 \cdot \sqrt{pn}$, it also holds that $\left\|w_{[n]\setminus I}\right\|_2 \geq \sqrt{n}/2$, which implies that for every $i \in ([n] \setminus I)$ it holds that $|w_i| = 1 \leq (2/\sqrt{n}) \cdot \left\|w_{[n]\setminus I}\right\|_2$. In this case, by the Berry-Esséen theorem (i.e., by Theorem 2.1.5), for any $t \geq 1$, the probability that $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I}\right\rangle$ falls in the interval $\pm 4t \cdot \sqrt{p} \cdot \left\|w_{[n]\setminus I}\right\|_2$ (which contains the interval $\pm t \cdot \|w_I\|_2$) is at most $O(t \cdot \sqrt{p} + \frac{2}{\sqrt{n}}) = O(t \cdot \sqrt{p})$. ■

Our goal in this section is to prove a statement that is similar to Fact 3.5.9, but that holds for an *arbitrary LTF* Φ, and holds also when the restriction ρ is sampled pseudorandomly, rather than uniformly. For simplicity, we only state the proposition informally at the moment (for a formal statement see Proposition 3.5.14):

Proposition 3.5.10 (pseudorandom restriction lemma for a single LTF; informal)**.** *Let $n \in \mathbb{N}$, let $p = n^{-\Omega(1)}$, and let $t = p^{-\Omega(1)}$. Let $\mathbf{y}$ be a distribution over $\{-1,1\}^{\log(1/p)\cdot n}$ that is p-almost $O(\log(1/p))$-wise independent, and let $\mathbf{z}$ be a distribution over $\{-1,1\}^n$ that is $p^{\Omega(1)}$-pseudorandomly concentrated. Then, for any LTF Φ over n input bits, the probability over choice of restriction $\rho \sim (\mathbf{y}, \mathbf{z})$ that $\Phi\restriction_\rho$ is t-balanced is at most $p^{\Omega(1)}$.*

A high-level description of the proof. Let $\Phi = (w, \theta)$ be an LTF over n input bits, and without loss of generality assume that $|w_1| \geq |w_2| \geq ... \geq |w_n|$. Denote by $I \subseteq [n]$ the set of variables that ρ keeps alive, and by $z_{[n]\setminus I} \in \{-1,1\}^{[n]\setminus I}$ the values that ρ assigns to the fixed variables. Then, the restricted function is of the form $\Phi\restriction_\rho = \left(w_I, \theta - \left\langle w_{[n]\setminus I}, z_{[n]\setminus I}\right\rangle\right)$, and the restricted function is t-balanced if and only if the sum $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I}\right\rangle$ falls in the interval $\theta \pm 2t \cdot \|w_I\|_2$. Our goal will be to show that this event is unlikely.

The proof is based on a modification of the case analysis that appears in [CSS16, Lem. 34, Sec. 4.2, Apdx. C.]. Specifically, for the parameter values $\mu = \Omega(1/t)$ and $k = \tilde{O}(t^2)$, we will consider two separate cases.

Case 1: The μ-critical index of Φ is at most k. Let $h \leq k$ be the μ-critical index of Φ, and denote $T = [n] \setminus [h]$. We first claim that with probability $1 - p^{\Omega(1)}$ over choice of $y \sim \mathbf{y}$ it holds that $\|w_I\|_2 \leq p^{\Omega(1)} \cdot \|w_T\|_2$. This is the case since with probability at least $1 - h \cdot p = 1 - p^{\Omega(1)}$, all the first h variables are fixed by ρ, and since the expected value of $\|w_{I \cap T}\|_2$ is $\sqrt{p} \cdot \|w_T\|_2$.

Condition on any fixed choice of $y \sim \mathbf{y}$ such that $\|w_I\|_2 \leq p^{\Omega(1)} \cdot \|w_T\|_2$. We will prove that with probability $1 - p^{\Omega(1)}$ over a *uniform choice of* $z \in \{-1,1\}^n$ it holds that $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I} \right\rangle$ does not fall in the interval $\theta \pm t \cdot p^{\Omega(1)} \cdot \|w_T\|_2$ (which contains the interval $\theta \pm t \cdot \|w_I\|_2$, due to our fixed choice of y). Since $\mathbf{z}$ is $p^{\Omega(1)}$-pseudorandomly concentrated, it will follow that this event also holds with probability $1 - p^{\Omega(1)}$ under a choice of $z \sim \mathbf{z}$.

To prove the claim about a uniform choice of $z \in \{-1,1\}^n$, condition *any arbitrary fixed values* $z_{[h]} \in \{-1,1\}^h$ for the first h variables. Then, the probability that $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I} \right\rangle$ falls in the interval $\theta \pm t \cdot p^{\Omega(1)} \cdot \|w_T\|_2$ (which is what we want to bound) equals the probability that $\langle w_{T\setminus I}, z_{T\setminus I} \rangle_2$ falls in the interval $\theta' \pm t \cdot p^{\Omega(1)} \cdot \|w_T\|_2$, where $\theta' = \theta - \left\langle w_{[h]}, z_{[h]} \right\rangle$. Since h is the μ-critical index of w we have that w_T is μ-regular; also, since $\|w_I\|_2 \leq p^{\Omega(1)} \cdot \|w_T\|_2$ (due to our choice of y), it follows that $w_{T\setminus I}$ is also (2μ)-regular and that $\|w_T\|_2 \approx \left\|w_{T\setminus I}\right\|_2$. By the Berry-Esséen theorem, the probability that $\langle w_{T\setminus I}, z_{T\setminus I} \rangle$ falls in an interval of length $t \cdot p^{\Omega(1)} \cdot \left\|w_{T\setminus I}\right\|_2$ is at most $O(t \cdot p^{\Omega(1)} + \mu) = p^{\Omega(1)}$ (see Lemma 3.5.11).

Case 2: The μ-critical index of Φ is larger than k. Similarly to the previous case, with probability at least $1 - p^{\Omega(1)}$ it holds that all the first k variables are fixed by ρ. Condition on any fixed $y \sim \mathbf{y}$ that fixes all the first k variables. What we will show is that with high probability over $z \sim \mathbf{z}$, the sum $\left\langle w_{[n]\setminus I}, \mathbf{z}_{[n]\setminus I} \right\rangle$ falls outside the interval $\theta \pm (1/4\mu) \|w_{>k}\|_2$, which contains the interval $\theta \pm t \cdot \|w_I\|_2$ (since $I \subseteq ([n] \setminus [k])$ and $\mu = \Omega(1/t)$).

As before, we first analyze the case in which z is chosen uniformly in $\{-1,1\}^n$. To do so we rely on a lemma of Servedio [Ser07], which asserts that the weights in w decrease exponentially up to the critical index. Intuitively, since the critical index is large (i.e., more than k), the exponential decay of the weights implies that $\|w_{>k}\|_2$ is small. Thus, when uniformly choosing $z \in \{-1,1\}^n$, the sum $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I} \right\rangle$ is unlikely to fall in the small interval $\theta \pm (1/4\mu) \cdot \|w_{>k}\|_2$; specifically, this happens with probability at most $\mu = p^{\Omega(1)}$ (see Claim 3.5.13.1 for a precise statement).

Since the event $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I} \right\rangle \in \theta \pm (1/4\mu) \cdot \|w_{>k}\|_2$ happens with probability $p^{\Omega(1)}$ when $z \in \{-1,1\}^n$ is chosen uniformly, and the distribution $\mathbf{z}$ is $p^{\Omega(1)}$-pseudorandomly concentrated, the event also happens with probability at most $p^{\Omega(1)}$ over a choice of $z \sim \mathbf{z}$.

The full proof. We will first prove an auxiliary lemma, which analyzes the effect of uniformly-chosen restrictions on regular LTFs (see Lemma 3.5.11). Then, we will prove a version of Proposition 3.5.10 that only holds for LTFs with *bounded critical index* (see Lemma 3.5.12), and a version of Proposition 3.5.10 that only holds for LTFs with *large critical index* (see Lemma 3.5.13). Finally, we will formally state a more general version of Proposition 3.5.10 and prove it (see Proposition 3.5.14).

The following auxiliary lemma considers a regular vector $w \in \mathbb{R}^m$, a fixed set of variables $I \subseteq [m]$ that will be kept alive, and a uniformly-chosen assignment $z \in \{-1,1\}^m$ for the fixed variables. The lemma will be used in the proof of Lemma 3.5.12.

Lemma 3.5.11 (pseudorandom restriction lemma for regular LTFs)**.** *Let $m \in \mathbb{N}$, let $\mu \in (0,1)$, and let $\lambda \leq 3/4$. Let $w' \in \mathbb{R}^m$ be a μ-regular vector, and let $I \subseteq [m]$ such that $\|w'_I\|_2 < \lambda \cdot \|w'\|_2$. Then, for any $\theta' \in \mathbb{R}$ and $t > 0$, the probability over uniform choice of $z \in \{-1,1\}^m$ that $\left\langle w'_{[m]\setminus I}, z_{[m]\setminus I} \right\rangle \in \theta' \pm t \cdot \lambda \cdot \|w'\|_2$ is at most $O(t \cdot \lambda + \mu)$.*

Proof. Note that $\left\|w'_{[m]\setminus I}\right\|_2^2 > \|w'\|_2^2/4$; this is the case because $\|w'_I\|_2^2 < \lambda \cdot \|w'\|_2^2 \leq \frac{3}{4} \cdot \|w'\|_2^2$. It follows that $w'_{[m]\setminus I}$ is 2μ-regular, since for every $i \in [m]$ we have that $\left|w'_i\right| \leq \mu \cdot \|w'\|_2 \leq 2\mu \cdot \left\|w'_{[m]\setminus I}\right\|_2$. It also follows that the interval $\theta \pm t \cdot \lambda \cdot \|w'\|_2$ is contained in the interval $\theta \pm 2t \cdot \lambda \cdot \left\|w'_{[m]\setminus I}\right\|_2$. By the Berry-Esséen theorem (i.e., by Theorem 2.1.5), the probability over a uniform choice of $z \in \{-1,1\}^m$ that the sum $\left\langle w_{[m]\setminus I}, z_{[m]\setminus I} \right\rangle$ falls in a fixed interval of length $2t \cdot \lambda \cdot \left\|w_{[m]\setminus I}\right\|$ is at most $O(t \cdot \lambda + \mu)$. ■

The following lemma asserts that a suitably-chosen pseudorandom restriction turns every LTF with *bounded critical index* to be very biased, with high probability. The specific parameters that are chosen for the lemma will be useful for us when proving the general case (i.e., Proposition 3.5.14, which holds for arbitrary LTFs).

Lemma 3.5.12 (pseudorandom restriction lemma for LTFs with small critical index)**.** *Let $n \in \mathbb{N}$, let $p \in [0,1]$ be a power of two, let $c \in \mathbb{N}$ be a constant, and let $t \leq p^{-1/(3c-2)}$ and $\mu = 1/4t^c$. Let $\mathbf{y}$ be a distribution over $\{-1,1\}^{\log(1/p)\cdot n}$ that is p-almost $O(\log(1/p))$-wise independent, and let $\mathbf{z}$ be a distribution over $\{-1,1\}^n$ that is μ-pseudorandomly concentrated. Then, for any LTF Φ over n input bits with μ-critical index at most $k = 10^3 \cdot \mu^{-2} \cdot \log^2(1/\mu)$, the probability over choice of $\rho \sim (\mathbf{y}, \mathbf{z})$ that $\Phi\restriction_\rho$ is t-balanced is at most $\tilde{O}(t^{1+c/2}) \cdot \sqrt{p} + O(t^{-c})$.*

Proof. Let $\Phi = (w, \theta)$ be an LTF gate over n input bits with critical index $h \leq k$, and without loss of generality assume that $|w_1| \geq |w_2| \geq \ldots \geq |w_n|$. Let $I \subseteq [n]$ be the random variable that is the set of live variables under $\mathbf{y}$; then, it holds that:

Claim 3.5.12.1. *With probability at least $1 - O(\mu + p \cdot k)$ over $y \sim \mathbf{y}$ it holds that $I \subseteq ([n] \setminus [h])$ and that $\|w_I\|_2 \leq \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$.*

Proof. Since $\mathbf{y}$ is p-almost $O(\log(1/p))$-wise independent, each variable is kept alive with probability at most $2p$. Thus, the probability over $y \sim \mathbf{y}$ that the first h variables are all fixed is at least $1 - 2p \cdot h$. Also, the expected value of $\left\|w_{I\cap([n]\setminus[h])}\right\|_2^2$ is at most $2p \cdot \left\|w_{[n]\setminus[h]}\right\|_2^2$, and hence with probability at least $1 - 2\mu$ it holds that $\left\|w_{I\cap([n]\setminus[h])}\right\|_2 \leq \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$. By a union-bound, with probability at least $1 - O(\mu + p \cdot h) > 1 - O(\mu + p \cdot k)$ it holds that $I \subseteq ([n] \setminus [h])$ and that $\|w_I\|_2 = \left\|w_{I\cap([n]\setminus[h])}\right\|_2 \leq \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$. □

Fix any $y \sim \mathbf{y}$ such that the first h variables are all fixed, and such that $\|w_I\|_2 \leq \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$. Our goal will be to prove that with high probability over $z \sim \mathbf{z}$ it holds that $\left\langle w_{[n]\setminus I}, \mathbf{z}_{[n]\setminus I}\right\rangle \notin \theta \pm t \cdot \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$; this suffices to prove the lemma, since $t \cdot \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\| \geq t \cdot \|w_I\|_2$. To do so, we first analyze the setting in which $z \in \{-1,1\}^n$ is chosen uniformly, rather than from the distribution $\mathbf{z}$:

Claim 3.5.12.2. *The probability over a uniform choice of $z \in \{-1,1\}^n$ that $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I}\right\rangle \in \theta \pm t \cdot \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$ is at most $O(t \cdot \sqrt{p/\mu} + \mu)$.*

Proof. The claim is trivial for $\mu \leq 2p$, so it suffices to prove the claim under the assumption that $\mu > 2p$. Condition on any arbitrary assignment $z_{[h]} \in \{-1,1\}^h$ for the first h variables, and note that the vector $w_{>h} \in \{-1,1\}^{n-h}$ is μ-regular (since h is the μ-critical index of Φ).

Let $T = [n] \setminus [h]$. Observe that when conditioning on $z_{[h]}$, the event $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I}\right\rangle \in \theta \pm t \cdot \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$ happens if and only if the event $\langle w_{T\setminus I}, z_{T\setminus I}\rangle \in \theta' \pm t \cdot \sqrt{p/\mu} \cdot \|w_T\|_2$ happens, where $\theta' = \theta - \left\langle w_{[h]}, z_{[h]}\right\rangle$. Since w_T is μ-regular, we can invoke Lemma 3.5.11 with $w' = w_T$ and with $\lambda = \sqrt{p/\mu} \leq 3/4$ (the inequality is since $\mu > 2p$), and deduce the probability of the event $\langle w_{T\setminus I}, z_{T\setminus I}\rangle \in \theta' \pm t \cdot \sqrt{p/\mu} \cdot \|w_T\|_2$ is at most $O(t \cdot \sqrt{p/\mu} + \mu)$. □

Since $\mathbf{z}$ is μ-pseudorandomly concentrated, it follows from Claim 3.5.12.2 that the probability over $z \sim \mathbf{z}$ that $\left\langle w_{[n]\setminus I}, \mathbf{z}_{[n]\setminus I}\right\rangle \in \theta \pm t \cdot \sqrt{p/\mu} \cdot \left\|w_{[n]\setminus[h]}\right\|_2$ is at most $O(t \cdot \sqrt{p/\mu} + \mu)$. Thus, the probability over choice of $\rho \sim (\mathbf{y}, \mathbf{z})$ that $\Phi\restriction_\rho$ is t-balanced is at most $O(t \cdot \sqrt{p/\mu} + \mu + p \cdot k) = \tilde{O}(t^{1+c/2}) \cdot \sqrt{p} + O(t^{-c})$, where the last equality relied on the hypothesis that $t \leq p^{-1/(3c-2)}$. ■

The following lemma is similar to Lemma 3.5.12, but holds for LTFs with *large critical index*.

Lemma 3.5.13 (pseudorandom restriction lemma for LTFs with large critical index)**.** *Let $n \in \mathbb{N}$, let $p \in [0,1]$ be a power of two, and let $\mu > 0$. Let $\mathbf{y}$ be a distribution over $\{-1,1\}^{\log(1/p)\cdot n}$ that is p-almost $O(\log(1/p))$-wise independent, and let $\mathbf{z}$ be a distribution over $\{-1,1\}^n$ that is μ-pseudorandomly concentrated. Then, for any LTF Φ over n input bits with μ-critical index larger than $k = 10^3 \cdot \mu^{-2} \cdot \log^2(1/\mu)$, the probability over choice of $\rho \sim (\mathbf{y}, \mathbf{z})$ that $\Phi\restriction_\rho$ is $(1/4\mu)$-balanced is $\tilde{O}(\mu^{-2}) \cdot p + O(\mu)$.*

Proof. Let $\Phi = (w, \theta)$ be an LTF gate over n input bits with μ-critical index larger than k, and without loss of generality assume that $|w_1| \geq |w_2| \geq ... \geq |w_n|$. Also, let $I \subseteq [n]$ be the random variable that is the set of live variables under $\mathbf{y}$. Note that the probability over $y \sim \mathbf{y}$ that $I \cap [k] \neq \emptyset$ is at most $2p \cdot k = \tilde{O}(\mu^{-2}) \cdot p$ (since $\mathbf{y}$ keeps each variable alive with probability at most $2p$).

Condition on any arbitrary $y \sim \mathbf{y}$ such that $[k] \cap I = \emptyset$. Our goal now is to show that the probability over $z \sim \mathbf{z}$ that $\Phi\restriction_\rho$ is $(1/4\mu)$-balanced is $O(\mu)$. We will actually prove a stronger claim: We will show that with probability at least $1 - O(\mu)$ it holds that $\left\langle w_{[n]\setminus I}, \mathbf{z}_{[n]\setminus I} \right\rangle \notin \theta \pm (1/4\mu) \cdot \|w_{>k}\|_2$ (this claim is stronger, since $I \subseteq ([n] \setminus [k])$, which implies that $\|w_{>k}\|_2 \geq \|w_I\|_2$). To prove this assertion we will rely on the following claim, which is essentially from [CSS16, Prop. 45] and generalizes [DGJ+10, Lem. 5.8]. (Since the proof is sketched in [CSS16], we include a full proof.)

Claim 3.5.13.1. *Let $\mu > 0$, let $r \in \mathbb{N}$, and let $k_{r,\mu} = \frac{4r \cdot \ln(3/\mu^2)}{\mu^2}$. Let $\Phi = (w, \theta)$ be an LTF over n input bits with μ-critical index larger than $k_{r,\mu}$ such that $|w_1| \geq ... \geq |w_n|$, and let $J \subseteq [n]$ such that $J \supseteq [k_{r,\mu}]$. Then, the probability under uniform choice of $z \in \{0,1\}^n$ that $\langle w_J, z_J \rangle \in \theta \pm (1/4\mu) \cdot \left\|w_{>k_{r,\mu}}\right\|_2$ is at most 2^{-r}.*

Proof. Since the critical index of Φ is larger than $k_{r,\mu}$, a lemma of Servedio [Ser07, Lem. 3] asserts that for any $1 \leq i < j \leq k_{r,\mu}$ it holds that

$$|w_j| \leq \left\|w_{\geq j}\right\|_2 \leq \left(1-\mu^2\right)^{(j-i)/2} \cdot \|w_{\geq i}\|_2 \leq \left(1-\mu^2\right)^{(j-i)/2} \cdot |w_i|/\mu \,. \tag{3.5.1}$$

(For an equivalent statement of the lemma see [DGJ+10, Lem. 5.5].) In particular, fixing $\gamma = \frac{2\ln(3/\mu^2)}{\mu^2}$, for any $i \in \mathbb{N}$ such that $i \cdot \gamma < k_{r,\mu}$ it holds that $|w_{i\cdot\gamma}| < |w_1|/3^i$.

Let $R = 1, \gamma, ..., r \cdot \gamma < k_{r,\mu}$, and consider any arbitrary fixed value of $z_{J\setminus R}$. Then, by a claim of Diakonikolas *et al.* [DGJ+10, p. Clm. 5.7], there exists at most a single value $z_R \in \{-1,1\}^r$ such that $\langle w_R, z_R \rangle \in \left(\theta - \left\langle w_{J\setminus R}, z_{J\setminus R} \right\rangle\right) \pm |w_{r\cdot\gamma}|/4$. Thus, the probability under a uniform choice of $z \in \{0,1\}^n$ that $\langle w_J, z_J \rangle \in \theta \pm |w_{r\cdot\gamma}|/4$ is at most 2^{-r}.

The claim follows since $\left\|w_{>k_{r,\mu}}\right\|_2 \leq \left\|w_{\geq(r+1)\cdot\gamma}\right\|_2 \leq \mu \cdot |w_{r\cdot\gamma}|$, where the first inequality is since $k_{r,\mu} > (r+1)\cdot\gamma$ and the second inequality is due to Eq. (3.5.1). □

We invoke Claim 3.5.13.1 with the value $r = \log(1/\mu)$ and with the set $J = [n] \setminus I$, while noting that the critical index of Φ is indeed larger than $k \geq k_{r,\mu}$. Since the interval $\theta \pm (1/4\mu) \cdot \|w_{>k}\|_2$ is contained in the interval $\theta \pm (1/4\mu) \cdot \left\|w_{>k_{r,\mu}}\right\|_2$ (because $k \geq k_{r,\mu}$),

we deduce that the event $\left\langle w_{[n]\setminus I}, z_{[n]\setminus I} \right\rangle \in \theta \pm (1/4\mu) \cdot \|w_{>k}\|_2$ happens with probability at most μ under a uniform choice of $z \in \{0,1\}^n$. Since $\mathbf{z}$ is μ-pseudorandomly concentrated, this event happens with probability at most $O(\mu)$ also under a choice of $z \sim \mathbf{z}$. ■

Finally, we are ready to state a more general version of Proposition 3.5.10 and to prove it. The proof will rely on Lemmas 3.5.12 and 3.5.13.

Proposition 3.5.14 (pseudorandom restriction lemma for an arbitrary LTF)**.** *Let $n \in \mathbb{N}$, let $p \in [0,1]$ be a power of two, let $c \in \mathbb{N}$ be a constant, and let $t \leq p^{-1/(3c-2)}$. Let $\mathbf{y}$ be a distribution over $\{-1,1\}^{\log(1/p)\cdot n}$ that is p-almost $O(\log(1/p))$-wise independent, and let $\mathbf{z}$ be a distribution over $\{-1,1\}^n$ that is $(1/4t^c)$-pseudorandomly concentrated. Then, for any LTF Φ over n input bits, the probability over choice of $\rho \sim (\mathbf{y}, \mathbf{z})$ that $\Phi\restriction_\rho$ is t-balanced is at most $\tilde{O}(t^{1+c/2}) \cdot \sqrt{p} + O(t^{-c})$.*

To obtain the parameters that were stated in Section 3.5.2.1, invoke Proposition 3.5.14 with $c = 2$. (When $c = 2$, the hypothesis that $t \leq p^{-1/(3c-2)} = p^{-1/4}$ is not required, since for $t > p^{-1/4}$ the probability bound in the lemma's statement is trivial.)

Proof of Proposition 3.5.14. Let $\Phi = (w, \theta)$ be an LTF gate over n input bits, let $\mu = 1/4t^c$, and let $k = 10^3 \cdot \mu^{-2} \cdot \log^2(1/\mu)$. If the μ-critical index of Φ is at most k, the asserted probability bound follows immediately from Lemma 3.5.12. On the other hand, if the μ-critical index of Φ is larger than k, we can rely on Lemma 3.5.13. The lemma asserts that the probability that $\Phi\restriction_\rho$ is $(1/4\mu)$-balanced is at most $\tilde{O}(\mu^{-2}) \cdot p + O(\mu) < \tilde{O}(t^{1+c/2}) \cdot \sqrt{p} + O(t^{-c})$, where the inequality relies on the hypothesis that $t \leq p^{-1/(3c-2)}$. Since $(1/4\mu) \geq t$, whenever $\Phi\restriction_\rho$ is $(1/4\mu)$-imbalanced it is also t-imbalanced. ■

3.5.3.2 Pseudorandom restriction algorithm for a "layer" of LTFs

The next step is to construct a pseudorandom restriction algorithm that transforms a depth-d linear threshold circuit into a depth-$(d-1)$ linear threshold circuit. The key part in this step is an application of Proposition 3.5.14.

Proposition 3.5.15 (pseudorandom restriction algorithm for a "layer" of LTFs)**.** *For every three constants $d \geq 2$ and $\epsilon > 0$ and $c > 0$, there exists a polynomial-time algorithm that gets as input a circuit $C \in \mathcal{C}_{n,d,n^{1+\epsilon}}$ and a random seed of length $O(\log(n) \cdot (\log\log(n))^2)$, and with probability at least $1 - n^{-\epsilon}$ outputs the following:*

1. *A restriction $\rho \in \{-1,1,\star\}^n$ that keeps at least $n' = \Omega(n^{1-24\cdot\epsilon})$ variables alive.*
2. *A circuit $\widetilde{C} \in \mathcal{C}_{n',d-1,(n')^{1+30\epsilon}}$ that agrees with C on at least $1 - n^{-c}$ of the inputs in the subcube that corresponds to ρ (i.e., $\Pr_{x \in \{-1,1\}^{|\rho^{-1}(\star)|}}[C\restriction_\rho(x) = \widetilde{C}(x)] > 1 - n^{-c}$).*

High-level overview of the proof. The key step of the algorithm is to apply Proposition 3.5.14 with parameters $p = n^{-\beta}$ and $c = 1$ and $t = p^{-1/5}$, where $\beta = O(\epsilon)$. The lemma asserts that, in expectation, all but approximately $n^{-\beta/5}$ of the gates will become t-imbalanced (for simplicity, ignore polylogarithmic factors for now). Such imbalanced gates are extremely close to a constant function, so we can replace the gates by the corresponding constants and get a circuit that agrees with the original circuit on almost all inputs.

As for the other $n^{-\beta/5}$-fraction of the gates, we expect that the number of wires feeding into them will decrease by a factor of p after the restriction. Specifically, assume that indeed the fan-in of each gate decreased by a factor of at least p; then, the expected number of wires feeding into the balanced gates after the restriction is at most

$$\sum_{\Phi \text{ gate}} \Pr[\Phi \text{ balanced}] \cdot p \cdot (\#\text{ wires incoming to } \Phi) \leq n^{-\beta/5} \cdot p \cdot n^{1+\epsilon} . \tag{3.5.2}$$

Thus, with probability at least $1 - n^{-\beta/10}$, the number of wires feeding into balanced gates is at most $(n^{\epsilon-\beta/10}) \cdot p \cdot n$, which is much smaller than the expected number of living variables (i.e., than $p \cdot n$) if $\beta > 10\epsilon$. When this happens, we can afford to simply fix all the variables that feed into balanced gates, making those gates constant too.

The argument above relied on the assumption that the fan-in of each gate Φ decreased by a factor of at least p. We can argue that this indeed holds with high probability for all gates with fan-in at least n^{α}, where $\alpha > \beta$, but we will need to separately handle gates with fan-in at most n^{α}. This will be done in two steps: The first is an initial preprocessing step (before applying Proposition 3.5.14), in which we fix every variable with fan-out more than $2 \cdot n^{\epsilon}$; since there are at most $n^{1+\epsilon}$ wires, this step fixes at most $n/2$ variables. Then, after applying Proposition 3.5.14 and fixing the variables that feed into balanced gates with fan-in at least n^{α}, we show that there exists a set I of variables of size approximately $n^{-(\alpha+\epsilon)} \cdot (p \cdot n)$ such that after fixing all variables outside I, each gate with fan-in at most n^{α} has fan-in at most one (see Claim 3.5.16.1). Thus, we can fix the variables outside I, and then replace each gate with fan-in at most n^{α} with the corresponding variable (or with its negation). At this point all the gates in the bottom layer have been replaced by constants or by variables.

Proof of Proposition 3.5.15. Let $G = \{\Phi_1, ..., \Phi_r\}$ be the set of gates in the bottom layer of C. For $\alpha = 12\epsilon$, let $S \subseteq G$ be the set of gates with fan-in at most n^{α}, and let $L = G \setminus S$ be the set of gates with fan-in more than n^{α}.

The restriction ρ will be composed of four restrictions $\rho_1, ..., \rho_4$. When describing the construction of each restriction, we will always assume that all previous restrictions were successful (we will describe exactly what "successful" means for each restriction). Also, after each restriction, we fix additional variables if necessary, in order to obtain an exact number of living variables in the end of the step.

Let $\mathbf{z}$ be a distribution over $\{-1, 1\}^n$ that is $(1/q(n))$-pseudorandom for LTFs, where q is a sufficiently large polynomial. We mention in advance that for each $i \in [4]$,

the values for variables that are fixed by ρ_i will always be decided by sampling from $\mathbf{z}$.

The first restriction ρ_1: Reduce the fan-out of input gates. We sample $z \sim \mathbf{z}$, and fix all variables with fan-out more than $2 \cdot n^\epsilon$ to values according to z. Since the number of wires between the bottom-layer gates and the input variables is at most $n^{1+\epsilon}$, and each fixing of a variable eliminates $2 \cdot n^\epsilon$ wires, we will fix no more than $n/2$ variables in this step. Let $n_1 = n/2$ be the number of living variables after the first step.

The second restriction ρ_2: Applying Proposition 3.5.14. We use Proposition 3.5.14 with the values $p = n^{-\beta}$, where $\beta = 11\epsilon$, and $c = 1$, and $t = p^{-1/5}$. [35] The distributions that we use are a $(1/\text{poly}(n))$-almost $O(\log(1/p))$-wise independent distribution $\mathbf{y}$ over $\{-1,1\}^{\log(1/p)\cdot n}$ and the aforementioned distribution $\mathbf{z}$ over $\{-1,1\}^n$.

Let $\mathcal{E}$ be the event in which ρ_2 keeps at least $(p \cdot n_1)/2$ variables alive, and for every gate $\Phi \in L$ it holds that $\text{fan-in}(\Phi\restriction_{\rho_2}) \leq 2p \cdot \text{fan-in}(\Phi)$. We claim that $\mathcal{E}$ happens with probability at least $1 - 1/\text{poly}(n)$. To see that this is the case, note that the expected number of living variables is $p \cdot n_1 = n^{\Omega(1)}$, and that for each gate $\Phi \in G$, the expected fan-in of $\Phi\restriction_{\rho_2}$ is $n^{\alpha-\beta} = n^{\Omega(1)}$. Since the choice of variables to keep alive is $\frac{1}{\text{poly}(n)}$-almost $O(1)$-independent, we can use Fact 2.1.1 to deduce that $\Pr[\mathcal{E}] \geq 1 - \frac{1}{\text{poly}(n)}$.

Now, assume without loss of generality that $L = \{\Phi_1, ..., \Phi_{r'}\}$, for some $r' \leq r$. For any $i \in [r']$, denote by $\mathcal{B}_i$ the event that Φ_i is t-balanced. Note that when conditioning on $\mathcal{E}$, the probability of each $\mathcal{B}_i$ is at most $\tilde{O}(n^{-\beta/5})$. Therefore, conditioned on $\mathcal{E}$, the expected number of wires feeding into t-balanced gates in L after the restriction is

$$\begin{aligned}
\mathbb{E}\left[\sum_{i\in[r']} \mathbf{1}_{\mathcal{B}_i} \cdot \text{fan-in}(\Phi_i\restriction_{\rho_2}) \middle| \mathcal{E}\right] &= \sum_{i\in[r']} \Pr[\mathcal{B}_i|\mathcal{E}] \cdot \mathbb{E}[\text{fan-in}(\Phi_i\restriction_{\rho_2})|\mathcal{E}, \mathcal{B}_i] \\
&\leq \sum_{i\in[r']} \tilde{O}(n^{-\beta/5}) \cdot (2p \cdot \text{fan-in}(\Phi_i)) \\
&= \tilde{O}(n^{-\beta/5}) \cdot p \cdot n^{1+\epsilon} \,.
\end{aligned}$$

Hence, conditioned on $\mathcal{E}$, the probability that the number of wires feeding into t-balanced gates in L after the restriction is more than $\tilde{O}(n^{-\beta/10}) \cdot p \cdot n^{1+\epsilon} = \tilde{O}(n^{\epsilon-\beta/10}) \cdot n^{1-\beta}$ is at most $O(n^{-\beta/10})$. We consider the restriction ρ_2 successful if $\mathcal{E}$ happens and if the number of wires between t-balanced gates in L and input gates is at most $\tilde{O}(n^{\epsilon-\beta/10}) \cdot n^{1-\beta}$. In this case, the number of currently-living variables is $n_2 = p \cdot n_1/2 = \frac{1}{4} \cdot n^{1-\beta}$.

After applying ρ_2, we replace any t-imbalanced gate $\Phi_i \in L$ with its most probable value $\sigma_i \in \{-1,1\}$. Note that by Theorem 2.1.4, each t-imbalanced gate Φ_i is $(\exp(-n^{\Omega(1)}))$-close to σ_i in the subcube that corresponds to the currently-living variables.

[35] For simplicity, we assume that $p = n^{-11\epsilon}$ is a power of two. Otherwise, we can choose β to be a value very close to 11ϵ such that p will be a power of two, with no meaningful change to the rest of the proof (the proof only relies on the fact that $10\epsilon < \beta < \alpha$).

The third restriction ρ_3: Eliminate L-gates that remained unbiased. In this step we sample $z \sim \mathbf{z}$ again, and fix all the variables that feed into t-balanced gates according to z. Assuming that ρ_2 was successful, the number of such variables is at most $\tilde{O}(n^{\epsilon-\beta/10}) \cdot n^{1-\beta} = o(n_2)$, where we used the fact that $\beta > 10\epsilon$. Denote the restriction applied in this step by ρ_3, and note that the number of living variables after applying ρ_3 is $n_3 = \Omega(n_2) = \Omega(n^{1-11\epsilon})$.

Our goal now is to claim that for each gate Φ_i that was replaced by a constant $\sigma \in \{-1,1\}$ prior to applying ρ_3, it still holds that Φ_i is close to σ in the subcube $\{-1,1\}^{\rho_3^{-1}(\star)}$. To do so we will rely on the following lemma:

Lemma 3.5.16 (bias preservation lemma)**.** *Let $n \in \mathbb{N}$, let $I \subseteq [n]$, and let $\delta > 0$. Let $\Phi = (w,\theta)$ be an LTF over n input bits that is δ-close to a constant function $\sigma \in \{-1,1\}$, and let $\mathbf{z}$ be a distribution over $\{-1,1\}^{[n]\setminus I}$ that is δ-pseudorandom for LTFs. Then, with probability at least $1-\sqrt{2\delta}$ over choice of $z \sim \mathbf{z}$ it holds that $\Phi\restriction_{(I,z)}$ is $\sqrt{2\delta}$-close to σ.*

Proof. Let $k = |I|$, and for simplicity of notation assume that $I = \{1,2,...,k\} \subseteq [n]$ and that $\sigma = 1$. By our hypothesis it holds that

$$\mathbb{E}_{x\in\{0,1\}^k}\left[\Pr_{z\in\{0,1\}^{n-k}}[\Phi(x\circ z)=1]\right] \geq 1-\delta\,.$$

Now, for every fixed $x \in \{0,1\}^k$, let $\Phi_x : \{0,1\}^{n-k} \to \{0,1\}$ be defined by $\Phi_x(z) = \Phi(x\circ z) = \operatorname{sgn}\left(\langle w_{>k}, z\rangle - (\theta - \langle x, w_{\leq k}\rangle)\right)$, and note that Φ_x is a linear threshold function of its input z. Since $\mathbf{z}$ is δ-pseudorandom for LTFs, we have that

$$\mathbb{E}_{x\in\{0,1\}^k}\left[\Pr_{z\sim\mathbf{z}}[\Phi(x\circ z)=1]\right] \geq \mathbb{E}_{x\in\{0,1\}^k}\left[\Pr_{z\in\{0,1\}^{n-k}}[\Phi(x\circ z)=1]-\delta\right]\,,$$

which implies that

$$\mathbb{E}_{z\sim\mathbf{z}}\left[\Pr_{x\in\{0,1\}^k}[\Phi(x\circ z)=1]\right] \geq 1-2\delta\,.$$

Finally, by Markov's inequality, the probability over $z \sim \mathbf{z}$ that $\Pr_{x\in\{0,1\}^k}[\Phi(x\circ z) = 1] < 1-\sqrt{2\delta}$ is less than $\sqrt{2\delta}$. □

We invoke Lemma 3.5.16 with I being the set of variables that are kept alive by ρ_3, and with $\delta = 1/\mathrm{poly}(n)$ for a sufficiently large polynomial (recall that each gate Φ_i that was replaced by a constant was in fact $\exp(-n^{\Omega(1)})$-close to the constant). After union-bounding over at most $r \leq n^{1+\epsilon}$ gates that were replaced by constants, with probability $1-1/\mathrm{poly}(n)$ it holds that all these gates are $\sqrt{2\delta}$-close to constants in the subcube $\{-1,1\}^{\rho_3^{-1}(\star)}$.

The fourth restriction ρ_4: Eliminate gates with small fan-in. We will rely on the following claim, which is an algorithmic version of [CSS16, Prop. 36]:

Claim 3.5.16.1. *For $k' = 2 \cdot n^{\alpha+\epsilon}$, we can deterministically find in $\mathrm{poly}(n)$ time a set I of at least n_3/k' living variables such that when fixing all variables not in I to any arbitrary values, the fan-in of each gate in S is at most one.*

Proof. Consider the graph in which the vertices are the input gates $x_1, ..., x_{n_3}$, and two vertices x_i and x_j are connected (in the graph) if and only if there exists a gate $\Phi_i \in S$ that is connected (in the circuit) to both x_i and x_j. Note that this graph has degree at most k', since every living variable has fan-out at most $2 \cdot n^{\epsilon}$, and every gate in S has fan-in at most n^{α}. Therefore, we can greedily construct an independent set I in the graph of size at least n_3/k', which is indeed the set of variables that we wanted. □

The algorithm finds a set I using Claim 3.5.16.1, samples $z \sim \mathbf{z}$, and fixes all the variables outside I according to z. This yields a restriction that reduces the fan-in of each gate in S to one. Thus, each gate $\Phi \in S$ now simply takes the value of an input gate (or its negation), which implies that the gates that are connected to Φ (in the layer above it) can be connected immediately to the corresponding input gate, and we can remove Φ from the circuit. The number of living variables is $n_4 = n_3/k' = \Omega(n^{1-24\epsilon})$.

To conclude, we claim that the gates that were previously replaced by constants are still close to constants in the new subcube. This is done by invoking Lemma 3.5.16 with I being the aforementioned set of size n_4, and with the parameter value $\sqrt{2\delta}$. After union-bounding over the gates that were replaced by constants, with probability at least $1 - n^{-2\epsilon}$ it holds that all these gates are δ'-close to constants in the final subcube, where $\delta' = n^{-(c+1+\epsilon)} \geq \sqrt{2\sqrt{2\delta}}$. It follows that the original circuit is δ''-close to the new circuit in the final subcube, where $\delta'' \leq \delta' \cdot n^{1+\epsilon} \leq n^{-c}$.

Accounting for the parameters. We obtained a circuit in $\tilde{C} \in \mathcal{C}_{n_4,d-1,n^{1+\epsilon}}$. Since $n^{1+\epsilon} = O(n_4^{\frac{1+\epsilon}{1-24\epsilon}}) < n_4^{(1+\epsilon)(1+25\epsilon)} \leq n_4^{1+30\epsilon}$, we have that $\tilde{C} \in \mathcal{C}_{n_4,d-1,n_4^{1+30\epsilon}}$. To sample the restriction $\rho = \rho_4 \circ ... \circ \rho_1$, we sampled from the distribution $\mathbf{z}$ four times, and from the distribution $\mathbf{y}$ a single time. A sample from $\mathbf{y}$ can obtained with seed length $O(\log(n))$, and relying on Theorem 2.4.20, each sample from $\mathbf{z}$ can be obtained with seed length $O(\log(n) \cdot (\log\log(n))^2)$.

Finally, let us account for the error probability. The first step is deterministic and always succeeds. In the second step, the algorithm is unable to simplify the circuit if the event $\mathcal{E}$ does not happen, or if the number of wires between t-balanced gates in L and input gates is too large. Denoting the latter event by $\mathcal{E}'$, the probability of error is at most $\Pr[\neg\mathcal{E}] + \Pr[\mathcal{E}'|\mathcal{E}] \leq O(n^{-\beta/10})$. The last type of error to account for is the probability that $\tilde{C}$ is not n^{-c}-close to C in $\{-1,1\}^{\rho^{-1}(\star)}$; as detailed above, this happens with probability at most $n^{-2\epsilon}$. The overall error is thus $O(n^{-\beta/10} + n^{-2\epsilon}) < n^{-\epsilon}$. ■

3.5.3.3 Pseudorandom restriction algorithm for linear threshold circuits

We are now ready to construct the pseudorandom restriction algorithm that simplifies any linear threshold circuit to a single LTF gate (i.e., Proposition 3.5.8). The proof will consist of $d-1$ applications of Proposition 3.5.15. In each application, we will use Lemma 3.5.16 to claim that all the approximations in previous applications of Proposition 3.5.15 still hold.

Proposition 3.5.17 (Proposition 3.5.8, restated)**.** *Let $d \geq 1$, let $\epsilon > 0$ be a sufficiently small constant, and let $\delta = d \cdot 30^{d-1} \cdot \epsilon$. Then, there exists a polynomial-time algorithm that for every $n \in \mathbb{N}$, when given as input a circuit $C \in \mathcal{C}_{n,d,n^{1+\epsilon}}$ and a random seed of length $O(\log(n) \cdot (\log\log(n))^2)$, with probability at least $1 - n^{-\epsilon/2}$ satisfies the following:*

1. *The algorithm outputs a restriction $\rho \in \{-1,1,\star\}^n$ that keeps at least $n^{1-\delta}$ variables alive.*
2. *The algorithm outputs an LTF $\Phi : \{-1,1\}^{\rho^{-1}(\star)} \to \{-1,1\}$ such that Φ is $1/10$-close to $C\restriction_\rho$ (i.e., $\Pr_{x \in \{-1,1\}^{\rho^{-1}(\star)}}[C(x) = \Phi(x)] \geq 9/10$).*

Proof. We repeatedly invoke Proposition 3.5.15, for $d-1$ times. For $i \in [d-1]$, let $\rho^{(i)}$ be the restriction that is obtained in the i^{th} invocation of Proposition 3.5.15, and let $\rho = \rho^{(d-1)} \circ ... \circ \rho^{(1)}$ be the final restriction. Let $C_0 = C$, and for $i \in [d-1]$, let C_i be the circuit that is obtained after the i^{th} invocation of Proposition 3.5.15. Also let $\epsilon_0 = \epsilon$ and $\epsilon_i = 30 \cdot \epsilon_{i-1} = 30^i \cdot \epsilon$, and let $n_0 = n$ and $n_i = \Omega\left((n_{i-1})^{1-24\epsilon_{i-1}}\right)$.

We say that an invocation of Proposition 3.5.15 is *successful* if the two items in the proposition's statement are satisfied (i.e., the algorithm outputs a restriction that keeps sufficiently many live variables, and a circuit of smaller depth that agrees with the original circuit on almost all inputs). Assuming all invocations of Proposition 3.5.15 are successful, for each $i \in [d-1]$ it holds that $C_i \in \mathcal{C}_{n_i,d-i,n_i^{1+\epsilon_i}}$, and in particular C_{d-1} is a single LTF Φ. Also, in this case, the number of living variables after all invocations is

$$n_{d-1} = n^{\Pi_{i=0}^{d-2}(1-24\epsilon_i)} > n^{1-24\cdot\sum_{i=0}^{d-2}\epsilon_i} > n^{1-24\cdot d\cdot\epsilon_{d-2}} > n^{1-\delta} . \tag{3.5.3}$$

The required seed length for the $d-1$ invocations of Proposition 3.5.15 is $\tilde{O}(\log(n))$. To bound the probability of error, for each $i \in [d-1]$, assume that all previous $i-1$ invocations were successful, and note that the probability that the i^{th} invocation of Proposition 3.5.15 fails is at most $n_{i-1}^{-\epsilon_{i-1}} < (n^{1-\delta})^{-\epsilon}$ (the inequality is since we assumed that the previous invocations of Proposition 3.5.15 were successful, which implies that $n_{i-1} \geq n^{1-\delta}$, by a calculation similar to Eq. (3.5.3)). Thus, the accumulated probability of error is at most $d \cdot (n^{1-\delta})^{-\epsilon} < n^{-\epsilon/2}$, where the inequality relied on the fact that ϵ is sufficiently small.

Condition on all the $d-1$ invocations of Proposition 3.5.15 being successful. Recall that in this case, for every $i \in [d-1]$ it holds that C_i is n^{-c}-close to $C_{i-1}\restriction_{\rho^{(i)}}$; we now claim that, with high probability, this approximation continues to hold even in the subcube that corresponds to the final restriction ρ.

Claim 3.5.17.1. *For every $i \in [d-1]$, with probability $1 - 1/\mathrm{poly}(n)$ it holds that $(C_{i-1})\restriction_\rho$ is $1/10d$-close to $(C_i)\restriction_\rho$.*

Proof. For each $j \in \{i, ..., d-1\}$, recall that $\rho^{(j)}$ is the composition of four restrictions, denoted by $\rho_1^{(j)}, ..., \rho_4^{(j)}$. Fix $i \in [d-1]$, condition on any fixed choice for $\rho_1^{(i)}$ and $\rho_2^{(i)}$, and let $C' = (C_{i-1})\restriction_{\rho_1^{(i)}, \rho_2^{(i)}}$. Recall that immediately after applying $\rho_2^{(i)}$, the algorithm from Proposition 3.5.15 replaces a set of $m \leq n^{1+\epsilon_{d-(i-1)}}$ LTF gates, denoted $\Phi_1, ..., \Phi_m$, with a corresponding set of constants $\sigma_1, ..., \sigma_m \in \{-1, 1\}$. Let $\widetilde{C}'$ be the circuit that is obtained from C' by the aforementioned replacement. Finally, note that for every choice of final restriction ρ it holds that $(C_{i-1})\restriction_\rho = C'\restriction_\rho$ and $(C_i)\restriction_\rho = \widetilde{C}'\restriction_\rho$.

Our goal now will be to show that for every fixed $k \in [m]$, with probability $1 - 1/\mathrm{poly}(n)$ over choice of ρ it holds that $(\Phi_k)\restriction_\rho$ is $1/(10dm)$-close to σ_k. This suffices to conclude the proof, since it follows (by a union-bound over the m gates) that with probability $1 - 1/\mathrm{poly}(n)$, for every $k \in [m]$ it holds that $(\Phi_k)\restriction_\rho$ is $1/(10dm)$-close to σ_k; and whenever the latter event happens we have that $C'\restriction_\rho$ is $1/(10d)$-close to $\widetilde{C}'\restriction_\rho$.

Towards the aforementioned goal, fix $k \in [m]$, and recall that Φ_k is δ_0-close to some constant function $\sigma_k \in \{-1, 1\}$, where $\delta_0 = \exp\left(n_{i-1}^{-\Omega(1)}\right) = \exp\left(n^{-\Omega(1)}\right)$, where the inequality is since $n_{i-1} = n^{\Omega(1)}$ (recall that we conditioned on all invocations of Proposition 3.5.15 being successful). Observe that the final restriction ρ is composed of $t \stackrel{\text{def}}{=\!=} 4 \cdot (d - i - 1) + 2$ additional restrictions on the domain of Φ_k: Two additional restrictions $\rho_3^{(i)}$ and $\rho_4^{(i)}$ in the i^{th} invocation of Proposition 3.5.15, and for each $j \in \{i+1, ..., d-1\}$, four restrictions $\rho_1^{(j)}, ..., \rho_4^{(j)}$ in the j^{th} invocation of Proposition 3.5.15. Recall that each of the t restrictions is chosen by first choosing (deterministically or pseudorandomly) a set of variables to keep alive, and then *independently* choosing values for the fixed variables. Therefore, we will now repeatedly use Lemma 3.5.16, to claim that each restriction preserves the closeness of Φ_k to σ_k.

For convenience, rename the t restrictions $\rho_3^{(i)}, \rho_4^{(i)}, \rho_1^{(i+1)}, ..., \rho_4^{(i+1)}, ..., \rho_1^{(d-1)}, ..., \rho_4^{(d-1)}$, and denote them by $\rho'^{(1)}, ..., \rho'^{(t)}$. Note that $\delta_0 < 2^{-t} \cdot n^{-2^t \cdot c}$, and for every $r \in [t]$ let $\delta_r = 2(\delta_{r-1})^{1/2} \geq \sqrt{2\delta_{r-1}}$. Repeatedly invoking Lemma 3.5.16, with probability at least $1 - O(\sqrt{\delta_{t-1}})$ it holds that $(\Phi_k)\restriction_{\rho'^{(1)} \circ ... \circ \rho'^{(r)}}$ is δ_r-close to σ_k. [36] Hence, with probability at least $1 - O(n^{-c})$ it holds that $(\Phi_k)\restriction_\rho$ is δ_t-close to σ_k, where $\delta_t = n^{-c} < 1/(10dm)$. □

Thus, with probability $1 - 1/\mathrm{poly}(n)$, for every $i \in [d-1]$ it holds that $(C_{i-1})\restriction_\rho$ is $1/10d$-close to $(C_i)\restriction_\rho$. Whenever this holds, by a union-bound it follows that $C\restriction_\rho = (C_0)\restriction_\rho$ is $1/10$-close to $(C_{d-1})\restriction_\rho = C_{d-1} = \Phi$. ■

[36] Formally, we prove by induction on $r \in [t]$ that with probability at least $1 - O(n^{-c})$ it holds that $(\Phi_k)\restriction_{\rho'^{(1)} \circ ... \circ \rho'^{(r)}}$ is δ_r-close to σ_k. For the base case $r = 1$ we rely on the hypothesis that Φ_k is δ_0-close to σ_k, and use Lemma 3.5.16 with the parameter value $\delta = 2^{-r} \cdot n^{-2^r \cdot c}$; and for the induction step $r > 1$, we condition on $(\Phi_k)\restriction_{\rho'^{(1)} \circ ... \circ \rho'^{(r-1)}}$ being δ_{r-1}-close to σ_k, and use Lemma 3.5.16 with the parameter value $\delta = \delta_{r-1}$.

3.5.4 Proof of Theorem 3.5.2

In this section we prove Theorem 3.5.2, and the corresponding result for derandomization of $\mathcal{ACC}^0$ (which was mentioned in the end of Section 3.5.1.2). In Section 3.5.4.1 we construct uniform sparse $\mathcal{CC}^0[2]$ circuits and $\mathcal{TC}^0$ circuits that encode balanced codes. In Section 3.5.4.2 we construct uniform sparse $\mathcal{CC}^0[2]$ circuits and $\mathcal{TC}^0$ circuits that compute averaging samplers. Finally, in we use the averaging samplers to prove Theorem 3.5.2 (in Section 3.5.4.3) and the bootstrapping result for derandomization of $\mathcal{ACC}^0$ (in Section 3.5.4.4).

3.5.4.1 A balanced code in uniform sparse $\mathcal{CC}^0[2]$ and $\mathcal{TC}^0$

In Section 3.5.4.1.1 we construct uniform sparse $\mathcal{CC}^0[2]$ circuits that compute *range detectors*, which are basic building blocks that will be useful in constructing the encoding circuit. Then, in Section 3.5.4.1.2 we use the range detectors to construct uniform sparse $\mathcal{CC}^0[2]$ circuits that encode codes with constant relative distance. In Section 3.5.4.1.3 we use the foregoing construction to construct uniform sparse $\mathcal{CC}^0[2]$ circuits that encode balanced codes. Finally, in Section 3.5.4.1.4 we show how to convert the latter circuits into uniform extremely sparse $\mathcal{TC}^0$ circuits that encode balanced codes.

3.5.4.1.1 Range detectors in uniform sparse $\mathcal{CC}^0[2]$ from lossless expanders As pointed out in [GHK+13], the task of constructing range detectors reduces to the task of constructing unbalanced *lossless expanders*: The latter are bipartite graphs with n "input" vertices and m "output" vertices such that each input vertex has degree d, and each set S of input vertices of size at most w has at least $(1-\epsilon)\cdot d\cdot|S|$ distinct neighbors among the output vertices. In particular, for each set S of size at most w, at least $(1-2\epsilon)\cdot d\cdot|S|$ of its neighbors are connected to a *single* vertex in S. [37]

The reason that lossless expanders are useful for constructing range detectors is the following. Assume that we construct a depth-one circuit $C:\{0,1\}^n\to\{0,1\}^m$ of parity gates by wiring according to a lossless expander. Then, for every input $x\in\{0,1\}^n$ with Hamming weight between $w/2$ and w, we have that $S_x=\{i\in[n]:x_i=1\}$ satisfies $|S_x|\leq w$. Since the graph is an expander, at least $(1-2\epsilon)\cdot d\cdot|S_x|$ output gates are connected to a *single* coordinate $i\in[n]$ such that $x_i=1$; hence, the Hamming weight of the output $C(x)$ is at least $(1-2\epsilon)\cdot d\cdot|S_x|=\Omega(d\cdot w)$. Also, since $|S_x|\leq w$, the Hamming weight of $C(x)$ is at most $d\cdot w$. Thus, if we use a lossless expander with $m=O(d\cdot w)$, we obtain an $(n,m,w,\Omega(m),m/2)$ range detector.

For our purposes we need a lossless expander with *small* input-degree d (to get a sparse circuit) and $m=O(d\cdot w)$. We will use the explicit construction of lossless expanders by Capalbo *et al.* [CRV+02]. To be consistent with their notation, we use uppercase letters instead of lowercase ones (i.e., the number of inputs is N, the number

[37] This is because $(1-\epsilon)\cdot d\cdot|S|$ edges are needed to obtain $(1-\epsilon)\cdot d\cdot|S|$ distinct neighbors of S, which leaves only $\epsilon\cdot d\cdot|S|$ edges that can "donate" a second edge to a neighbor of S.

of outputs is M, and the weight is W). The input-degree that is obtained using their construction is $D = \exp(\text{poly}\log\log(N))$, which is sufficiently small for our purposes.

Proposition 3.5.18 (constructing sparse range detectors using [CRV+02])**.** *There exist two constants $\eta, \mu > 0$ such that the following holds. There exists a deterministic polynomial-time algorithm that gets as input 1^N, where N is a power of two, and $W \leq \eta \cdot N$, and outputs the following depth-one circuit that consists of parity gates. The circuit has N input bits and $M = O\left(W \cdot \exp(\text{poly}\log\log(N))\right)$ output gates, and $N \cdot \exp(\text{poly}\log\log(N))$ wires, and the circuit computes an $(N, M, W, \mu \cdot M, M/2)$-range detector.*

Proof. We use the lossless expander construction of [CRV+02, Thm 7.3], instantiated with the following parameters: $n = \log(N)$, and $\epsilon = 0.01$, and $t \in \mathbb{N}$ such that $t + \alpha \cdot \log^3(t/\epsilon) = n - \log(W) - \beta$, where $\alpha, \beta \geq 1$ are positive constants that will be specified later.[38] Their construction yields a lossless expander with the following parameters:

1. The number of input vertices is N.
2. The number of output vertices is $M = 2^{n-t} = 2^{\beta} \cdot W \cdot 2^{\alpha \cdot \log^3(t/\epsilon)}$.
3. The degree of input vertices is $D = 2^d = 2^{\gamma \cdot \log^3(t/\epsilon)} < \exp(\text{poly}\log\log(N))$ (where $\gamma > 1$ is a universal constant from [CRV+02]).
4. All sets $S \subseteq [N]$ of size at most $K = 2^{k_{max}}$ have at least $0.99 \cdot D \cdot |S|$ neighbors, where $K = 2^{n-t-d-\log(1/\epsilon)-\delta} \geq W$ (in the calculation, $\delta > 1$ is a universal constant from [CRV+02], and we choose $\alpha = \gamma$ and $\beta \geq \log(1/\epsilon) + \delta$).

The neighbor function $E : [N] \times [D] \to [M]$ is computable in time $\text{poly}\log(N)$, which means that the circuit (with parity gates) corresponding to the expander can be constructed in time $\text{poly}(N)$. By the preceding discussion, to prove that the circuit is an $(N, M, W, \Omega(M), M/2)$-range detector it remains to verify that $M \geq 2 \cdot D \cdot W$ and $M = O(D \cdot W)$. The first inequality holds because $\beta \geq 1$, and the second inequality holds because β is bounded by a universal constant. ∎

Proposition 3.5.18 gives range detectors for all weights w up to $\Omega(n)$. Constructing range detectors for weight $w = \Omega(n)$ is much simpler: The identity mapping preserves the constant relative weight, and so we just need to handle the edge case of $w > n/2$ (recall that we want the output weight to be at most $(1 - \Omega(1)) \cdot m$).[39]

[38] Specifically, in the proof α will be a specific universal constant, and β will be lower-bounded by another universal constant. We will thus increase β (by a constant) such that $n - \log(W) - \beta$ will touch the set $\{t + \alpha \cdot \log^3(t/\epsilon)\}_{t \in \mathbb{N}}$. The fact $n - \log(W) - \beta \geq 1 + \alpha \cdot \log^3(1/\epsilon)$ follows from the hypothesis that $W \leq \eta \cdot N$ (where η is sufficiently small and depends on α and on the lower bound for β).

[39] We will not use the fact that the output weight is at most $(1 - \Omega(1)) \cdot m$, but we nevertheless present this construction for completeness.

Proposition 3.5.19 (constructing a layer of range detectors)**.** *For some universal constant $\rho > 0$, there exists a deterministic polynomial-time algorithm that gets as input 1^n, where n is a power of two, and outputs the following $\log(n)$ depth-one circuits that consist of parity gates. For $i = 1, ..., \log(n)$, the i^{th} circuit computes an $(n, m, w, \rho \cdot m, (1-\rho) \cdot m)$-range detector, where $w = 2^i$ and $m = O(w \cdot \exp(\text{poly} \log \log(n))$ and the circuit has at most $n \cdot \exp(\text{poly} \log \log(n))$ wires.*

Proof. Let $\eta, \mu > 0$ be the two constants from Proposition 3.5.18. For every $i \leq \log(n) - \log(1/\eta)$, we use Proposition 3.5.18 to get an $(n, m, w, \eta \cdot m, m/2)$-range detector with $m = O(n \cdot \exp(\text{poly} \log \log(n))$ and $n \cdot \exp(\text{poly} \log \log(n))$ wires. For $\log(n) - \log(1/\eta) < i < \log(n)$, we just use the identity mapping to get an $(n, m = n, w, \eta \cdot m, m/2)$-range detector with n wires.

For $i = \log(n)$, we map n input bits to $m = (3/2) \cdot n$ output bits as follows: The first n output bits are the identity mapping of the input bits, and each of the last $n/2$ output bits is connected to a consecutive pair of input bits (i.e., for $i \in [n]$, the i^{th} input bit is connected to the i^{th} output bit; and for $i \in [n/2]$, the $(n+i)^{th}$ output bit is connected to input bits $2i-1$ and $2i$). Note that for any input $x \in \{0,1\}^n$ of Hamming weight at least $n/2$, the Hamming weight of the output is at least $n/2 = m/3$. Now, if the Hamming weight of x is at most $(3/4) \cdot n$, then the Hamming weight of the output is at most $n = (5/6) \cdot m$; and on the other hand, if the Hamming weight of x is more than $(3/4) \cdot n$, then $\Pr_{i \in [n/2]}[x_{2i-1} \oplus x_{2i} = 0] \geq \Pr_{i \in [n/2]}[x_{2i-1} = x_{2i} = 1] \geq 1/2$, which implies that the Hamming weight of the output is at most $m - n/4 = (5/6) \cdot m$. Thus, this yields an $(n, m, w = n, m/2, (5/6) \cdot m)$-range detector with $2n$ wires. The proposition follows by taking $\rho = \min\{\mu, \eta, 1/6\}$. ■

3.5.4.1.2 A code with constant relative distance in uniform sparse $\mathcal{CC}^0[2]$

Our goal now is to use the range detectors to construct an encoding circuit. To do so, we will put the range detectors as a layer of gates above the inputs, and use an additional top layer to combine the range detectors into a code. The construction of the top layer follows the construction in [GHK+13, p. Clm. 34], with different parameters.

Proposition 3.5.20 (uniform extremely sparse $\mathcal{CC}^0[2]$ circuits for encoding an "almost–good" code)**.** *For some universal constant $\rho > 0$, there exists a deterministic polynomial-time algorithm that gets as input 1^n, where n is a power of two, and outputs a depth-two circuit of that consists of parity gates, and satisfies the following:*

1. *The circuit computes the encoding function of a linear code $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ with constant relative distance $\rho > 0$, where $\hat{n} = n \cdot \exp(\text{poly} \log \log(n))$.*

2. *The circuit has $n \cdot \exp(\text{poly} \log \log(n))$ gates and $n \cdot \exp(\text{poly} \log \log(n))$ wires.*

Proof. We first use Proposition 3.5.19 to obtain $\log(n)$ range detectors with parameters as in the proposition; the middle layer of the circuit consists of these range detectors. Note that for every non-zero $x \in \{0,1\}^n$ it holds that for some $i \in [\log(n)]$, the i^{th}

range detector maps x to $m_i = O(2^i \cdot \exp(\text{poly}\log\log(n)))$ outputs such that between $\rho \cdot m_i$ and $(1-\rho) \cdot m_i$ of the outputs are set to one.

Towards constructing the top layer, we first construct a top layer of n_0 gates, where $n_0 = n \cdot \exp(\text{poly}\log\log(n)) \geq \max_{i \in [\log(n)]}\{m_i\}$, that satisfies the following property: For every non-zero $x \in \{0,1\}^n$, a constant fraction of the n_0 top gates are connected to a gate in the middle layer that is set to one. This property does not suffice for the complete construction, since a top gate might touch an *even* number of gates in the middle layer that are set to one (i.e., their parity will be zero); later on we will replace the top n_0 gates by $\hat{n}$ parity gates in a manner that will solve this problem.

For now, imagine a top layer of n_0 gates. For $i \in [\log(n)]$, we connect the i^{th} range detector to these top gates, while ensuring that if a constant fraction of the outputs of the range detector are set to one, then a constant fraction of the top gates touch an output gate of the range detector that is set to one. Specifically, we connect the layers such that each top gate is connected to exactly one output of the range detector, and the degree of the outputs of the range detector is at least $\lfloor n_0/m_i \rfloor$. [40] Let S be the set of output gates of the range detector that are set to one. If $|S| \geq \rho \cdot m_i$, then the number of wires outgoing from S is at least $\rho \cdot m_i \cdot \lfloor n_0/m_i \rfloor$, which implies that the number of top gates connected to S is at least $\rho \cdot m_i \cdot \lfloor n_0/m_i \rfloor = \Omega(\rho \cdot n_0)$.

After connecting all range detectors to the top layer of n_0 gates in this manner, the degree of each top gate is exactly $\log(n)$. To obtain the actual top layer (with $\hat{n}$ parity gates instead of n_0 gates), we replace each of the n_0 gates, denoted g, with $O(\log(n))$ parity gates that compute a good code on the middle gates that feed into g. (This can be any code with constant rate and constant relative distance, since we do not require that these sub-circuits over $\log(n)$ bits will be sparse.) Thus, if a constant fraction of the previous n_0 top gates were connected to a gate in the middle layer that is set to one, then a constant fraction of the actual $\hat{n}$ top parity gates are set to one.

The number of wires between the top layer and the middle layer is at most $\hat{n} \cdot \log(n) = n_0 \cdot O(\log(n)) \cdot \log(n) = n \cdot \exp(\text{poly}\log\log(n))$, and the number of wires between the middle layer and the inputs layer is also bounded by $n \cdot \exp(\text{poly}\log\log(n))$. Also, the top layer has more gates than the middle layer, and thus the circuit has $O(n \cdot \exp(\text{poly}\log\log(n)))$ gates. ■

3.5.4.1.3 A balanced code in uniform sparse $\mathcal{CC}^0[2]$

In this section we use an additional sparse circuit to amplify the distance of the code in Proposition 3.5.20 from $\Omega(1)$ to $1/2 - \epsilon$. This is done relying on the well-known strategy of Naor and Naor [NN93], which uses random walks on an expander on the vertex-set $[\hat{n}]$. Specifically, we use the following construction:

Proposition 3.5.21 (amplifying the distance of a code by a sparse circuit)**.** *For some universal constant $r_0 > 1$, there exists a polynomial-time algorithm that is given as input $1^{\hat{n}}$, a constant $\rho > 0$, and $\epsilon = \epsilon(\hat{n}) > 0$, and outputs a circuit C such that:*

[40] That is, we split the n_0 top gates into m_i blocks of size either $\lceil n_0/m_i \rceil$ or $\lfloor n_0/m_i \rfloor$, and connect each of the m_i gates in the middle to all top gates in a corresponding block.

1. *The circuit C maps $\hat{n}$ input bits to $\bar{n} = \hat{n} \cdot (1/\epsilon)^{r_0/\rho}$ output bits.*

2. *For every $\hat{x} \in \{0,1\}^{\hat{n}}$ with relative Hamming weight at least ρ, the relative Hamming weight of $\bar{x} = C(\hat{x})$ is between $1/2 - \epsilon$ and $1/2$.*

3. *The circuit C has depth one, and each output bit of C is a linear function of $O(\log(1/\epsilon)/\rho)$ input bits.*

Proof. The algorithm first constructs an expander graph G on $\hat{n}$ vertices; that is, a d_G-regular graph over the vertex-set $[\hat{n}]$ with a constant spectral gap.[41] Consider a random walk that starts from a uniform $i \in [\hat{n}]$ and walks $\ell - 1$ steps, where $\ell = \frac{c_G}{\rho} \cdot \log(1/\epsilon)$ and $c_G > 1$ is a constant that depends only on the spectral gap of G. By the hitting property of expander random walks (see, e.g., [Gol08, Thm 8.28]), with probability at least $1 - \epsilon$ such a walk hits $i \in [\hat{n}]$ such that $x_i \neq 0$ (this is because the set $\{i \in [\hat{n}] : x_i \neq 0\}$ has density at least ρ). Thus, if we first take such a random walk, and then output a random parity of the values of $\hat{x}$ at the coordinates corresponding to the vertices in the walk, the output will equal one with probability at least $1/2 - \epsilon$ and at most $1/2$.

The mapping of $\hat{x}$ to $\bar{x} = C(\hat{x})$ is obtained by considering all the possible outcomes of the random process above. Specifically, for every random walk $W = \left(i_1^{(W)}, ..., i_\ell^{(W)}\right)$ of length $\ell - 1$ on G, and every subset $S \subseteq [\ell]$, we have a corresponding coordinate (W, S) in $C(\hat{x})$. The value of $C(\hat{x})$ at coordinate (W, S) is the parity of the bits of $\hat{x}$ in the locations corresponding to S in walk W; that is, $C(\hat{x})_{(W,S)} = \oplus_{j \in S} \hat{x}_{i_j^{(W)}}$. Note that the length of $C(\hat{x})$ is $\hat{n} \cdot (d_G)^{\ell-1} \cdot 2^\ell = \hat{n} \cdot (1/\epsilon)^{c'_G/\rho}$, where c'_G is a large constant that only depends on the degree and the spectral gap of the expander G. Also, the mapping of $\hat{x}$ to $C(\hat{x})$ is linear, and every coordinate of $C(\hat{x})$ is the parity of ℓ coordinates of $\hat{x}$. ∎

Combining Propositions 3.5.20 and 3.5.21, we get the following:

Theorem 3.5.22 (a balanced code in superlinear $\mathcal{CC}^0$)**.** *For some universal constant $r_1 > 1$ there exists a polynomial-time algorithm that is given as input 1^n and $\epsilon = \epsilon(n)$, and outputs a $\mathcal{CC}^0[2]$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n} = n \cdot \exp(\text{poly}\log\log(n)) \cdot (1/\epsilon)^{r_1}$ such that every codeword has relative Hamming weight $1/2 \pm \epsilon$.*

2. *The circuit C has depth two and at most $n \cdot \exp(\text{poly}\log\log(n)) \cdot (1/\epsilon)^{r_1} \cdot \log(1/\epsilon)$ wires.*

[41] For a suitable construction see, e.g., [Gol08, Thm E.10]. This specific construction requires $\hat{n}$ to be a square, so we might need to pad the input $x \in \{0,1\}^{\hat{n}}$ with zeroes such that it will be of length $4^k = (2^k)^2$, for $k \in \mathbb{N}$. Since such a padding will not affect the rest of the argument, we ignore this issue.

Proof. Let $\rho > 0$ be any sufficiently small constant. We first use the algorithm from Proposition 3.5.20 to construct a depth-two $\mathcal{CC}^0[2]$ circuit $C_0 : \{0,1\}^n \to \{0,1\}^{\hat{n}}$, where $\hat{n} = n \cdot \exp(\text{poly}\log\log(n))$, that computes a linear code with relative distance ρ using $n \cdot \exp(\text{poly}\log\log(n))$ wires. Now, we use the algorithm from Proposition 3.5.21 to construct a depth-one $\mathcal{CC}^0[2]$ circuit $C_1 : \{0,1\}^{\hat{n}} \to \{0,1\}^{\bar{n}}$, where $\bar{n} = \hat{n} \cdot (1/\epsilon)^{r_0/\rho}$, that maps any $\hat{x} \in \{0,1\}^{\hat{n}}$ of relative weight at least ρ to $\bar{x} \in \{0,1\}^{\bar{n}}$ of relative weight at least $1/2 - \epsilon$, using $O(\bar{n} \cdot \log(1/\epsilon))$ wires.

We now combine C_0 and C_1. Note that the naive combination has depth three, but that the top layer has fan-in $O(\log(1/\epsilon))$ and that the middle layer has fan-in $O(\log(n))$. Thus, we can collapse the two layers, and obtain a depth-two circuit in which the top layer has fan-in $O(\log(n) \cdot \log(1/\epsilon))$. Overall, we obtain a circuit with at most $n \cdot \exp(\text{poly}\log\log(n)) \cdot (1/\epsilon)^{r_0/\rho} \cdot \log(1/\epsilon)$ wires. ■

Relying on the Johnson bound (see, e.g., [AB09, Thm. 19.23]), Theorem 3.5.22 also yields a list-decodable code:

Corollary 3.5.23 (a list-decodable code in superlinear $\mathcal{CC}^0$)**.** *For some universal constant $r_2 > 1$, there exists a polynomial-time algorithm that is given as input 1^n and $\delta = \delta(n)$, and outputs a $\mathcal{CC}^0[2]$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n} = n \cdot \exp(\text{poly}\log\log(n)) \cdot (1/\delta)^{r_2}$ such that in any Hamming ball of radius $1/2 - \delta$ in $\{0,1\}^{\bar{n}}$ there exist at most $O(1/\delta^2)$ codewords.*
2. *The circuit C has depth two and at most $n \cdot \exp(\text{poly}\log\log(n)) \cdot (1/\delta)^{r_2} \cdot \log(1/\delta)$ wires.*

3.5.4.1.4 A balanced code in extremely sparse $\mathcal{TC}^0$ In this section we convert our construction of uniform $\mathcal{CC}^0[2]$ circuits into uniform extremely sparse $\mathcal{TC}^0$ circuits. We do this by replacing each parity gate by a $\mathcal{TC}^0$ circuit, using the constructions of uniform circuits by [BBL92; PS94], in each of the constructions in the previous sections. We start with the construction of encoding circuits for a code with constant relative distnce:

Proposition 3.5.24 (uniform extremely sparse $\mathcal{TC}^0$ circuits for encoding an "almost–good" code)**.** *For some universal constants $\rho > 0$ and $c_0 > 1$, there exists a deterministic polynomial-time algorithm that gets as input 1^n, where n is a sufficiently large power of two, and a constant $d \geq 2$, and outputs a $\mathcal{TC}^0$ circuit that satisfies the following:*

1. *The circuit computes the encoding function of a linear code $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ with constant relative distance $\rho > 0$, where $\hat{n} = n \cdot \exp(\text{poly}\log\log(n))$.*
2. *The circuit has depth $d + 2$ and at most $n^{1+c_0 \cdot \phi^{-d} + o(1)}$ wires, where $\phi = \frac{1+\sqrt{5}}{2}$.*

Proof. Observe that in the construction in the proof of of Proposition 3.5.20, each of the gates in the middle layer (i.e., the gates that compute the range detectors) may

compute the parity of an arbitrary number of input bits, but each gate in the top layer only computes the parity of $\log(n)$ gates in the middle layer.

We first replace each parity gate in the middle layer by a depth-d $\mathcal{TC}^0$ circuit for parity, using the construction in [PS94, Thm. 1]. For every gate g, denote by n_g the fan-in of g; then, the $\mathcal{TC}^0$ circuit that computes g has $n_g^{1+c_0 \cdot \phi^{-d}}$ wires, for some $c_0 > 1$. Thus, the overall number of wires that are contributed by the $\mathcal{TC}^0$ circuits that compute the middle layer is at most

$$\sum_g n_g^{1+c_0 \cdot \phi^{-d}} \leq \left(\sum_g n_g \right)^{1+c_0 \cdot \phi^{-d}} \leq \left(n \cdot \exp(\operatorname{poly} \log \log(n)) \right)^{1+c_0 \cdot \phi^{-d}} ,$$

which is $n^{1+c_0 \cdot \phi^{-d} + o(1)}$.

Now, we replace each gate in the top layer, which computes the parity of $\log(n)$ bits, by a $\mathcal{TC}^0$ circuit of depth two with $O(\log^2(n))$ wires (using the standard construction with a quadratic number of wires; see, e.g., [BBL92, Sec. 1]). The number of $\mathcal{TC}^0$ circuits that we use (i.e., the number of top gates) is $n \cdot \exp(\operatorname{poly} \log \log(n))$, and each of them contributes $O(\log^2(n))$ wires, so the overall contribution of this layer to the number of wires is $n^{1+o(1)}$. ∎

In Proposition 3.5.21, we amplify the distance of the code from $\Omega(1)$ to $1/2 - \epsilon$ by a single layer of parity gates, each of which has fan-in at most $O(\log(1/\epsilon))$. By converting each such parity gate to a depth-two $\mathcal{TC}^0$ circuit, we obtain the following:

Proposition 3.5.25 (amplifying the distance of a code by a sparse $\mathcal{TC}^0$ circuit)**.** *For some universal constant $r_0 > 1$, there exists a polynomial-time algorithm that is given as input $1^{\hat{n}}$, a constant $\rho > 0$, and $\epsilon = \epsilon(\hat{n}) > 0$, and outputs a $\mathcal{TC}^0$ circuit C such that:*

1. *The circuit C maps $\hat{n}$ input bits to $\bar{n} = \hat{n} \cdot (1/\epsilon)^{r_0/\rho}$ output bits.*
2. *For every $\hat{x} \in \{0,1\}^{\hat{n}}$ with relative Hamming weight at least ρ, the relative Hamming weight of $\bar{x} = C(\hat{x})$ is between $1/2 - \epsilon$ and $1/2$.*
3. *Each output bit of C is a linear function of $O(\log(1/\epsilon)/\rho)$ input bits.*
4. *The circuit C has depth two and $O\left(\hat{n} \cdot (1/\epsilon)^{r_0/\rho} \cdot \log^2(1/\epsilon)\right)$ wires.*

Now, by combining Propositions 3.5.24 and 3.5.25, we get the following:

Theorem 3.5.26 (a balanced code in extremely sparse $\mathcal{TC}^0$)**.** *For some universal constants $c_0 > 1$ and $r_1 > 1$, there exists a polynomial-time algorithm that is given as input 1^n and $\epsilon = \epsilon(n)$ and a constant $d \geq 2$, and outputs a $\mathcal{TC}^0$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n} = n \cdot \exp(\operatorname{poly} \log \log(n)) \cdot (1/\epsilon)^{r_1}$ such that every codeword has relative Hamming weight $1/2 \pm \epsilon$.*

2. *The circuit C has depth $d+4$ and at most $n^{1+c_0\cdot\phi^{-d}+o(1)}+n\cdot\exp(\text{poly}\log\log(n))\cdot(1/\epsilon)^{r_1}\cdot\log^2(1/\epsilon)$ wires, where $\phi=\frac{1+\sqrt{5}}{2}$.*

Corollary 3.5.27 (a list-decodable code in sparse $\mathcal{TC}^0$)**.** *For some universal constants $c_0>1$ and $r_2>1$, there exists a polynomial-time algorithm that is given as input 1^n and $\delta=\delta(n)$ and a constant $d\geq 2$, and outputs a $\mathcal{TC}^0$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n}=n\cdot\exp(\text{poly}\log\log(n))\cdot(1/\delta)^{r_2}$ such that in any Hamming ball of radius $1/2-\delta$ in $\{0,1\}^{\bar{n}}$ there exist at most $O(1/\delta^2)$ codewords.*

2. *The circuit C has depth $d+4$ and at most $n^{1+c_0\cdot\phi^{-d}+o(1)}+n\cdot\exp(\text{poly}\log\log(n))\cdot(1/\delta)^{r_2}\cdot\log^2(1/\delta)$ wires, where $\phi=\frac{1+\sqrt{5}}{2}$.*

3.5.4.2 An averaging sampler in uniform sparse $\mathcal{CC}^0[2]$ and $\mathcal{TC}^0$

Let us recall the notion of weak combinatorial designs, which was introduced by Raz, Reingold, and Vadhan [RRV02], and state their result that Trevisan's extractor [Tre01] can be instantiated with weak designs instead of standard combinatorial designs.

Definition 3.5.28 (weak designs)**.** *For positive integers $m,\ell,t\in\mathbb{N}$ and an integer $\rho>1$, an (m,ℓ,t,ρ) weak design is a collection of sets $S_1,\ldots,S_m\subseteq[t]$ such that for every $i\in[m]$ it holds that $|S_i|=\ell$ and $\sum_{j<i}2^{|S_i\cap S_j|}<(m-1)\cdot\rho$.*

Theorem 3.5.29 (extractors from weak designs [RRV02, Prop. 10])**.** *Let $m<k<n$ be three integers, and let $\epsilon>0$. Let $\mathtt{ECC}:\{0,1\}^n\to\{0,1\}^{\bar{n}}$ be a code such that in every Hamming ball of radius $1/2-\delta$ in $\{0,1\}^{\bar{n}}$ there exist at most $1/\delta^2$ codewords, where $\delta=\epsilon/4m$. Let $S_1,\ldots,S_m\subseteq[t]$ be an (m,ℓ,t,ρ) weak design with $\ell=\log(\bar{n})$ and $\rho=\frac{k-3\cdot\log(m/\epsilon)-t-3}{m}$. Then, the function $\mathtt{Ext}:\{0,1\}^n\times\{0,1\}^t\to\{0,1\}^m$ that is defined by $\mathtt{Ext}(x,z)=(\mathtt{ECC}(x)_{z_{S_1}},\ldots,\mathtt{ECC}(x)_{z_{S_m}})$ is a (k,ϵ)-extractor.*

We will need a specific construction of weak designs, in which the intersection parameter $\log(\rho)$ is large, but the universe size t is small (i.e., for sets of size $|S_i|=\ell$ we will require that $\log(\rho)\approx.99\cdot\ell$ and $t\approx1.01\cdot\ell$).

Lemma 3.5.30 (constructing weak designs)**.** *There exists an algorithm that gets as input $m\in\mathbb{N}$ and $\ell\in\mathbb{N}$ and $\rho\in\mathbb{N}$ such that $\log(\rho)=(1-\alpha)\cdot\ell$, where $\alpha\in(0,1/4)$, and satisfies the following. The algorithm runs in time $\text{poly}(m,2^\ell)$ and outputs an (m,ℓ,t,ρ) weak design, where $t=\lceil(1+4\alpha)\cdot\ell\rceil$.*

Proof. Let $t=\lceil(1+4\alpha)\cdot\ell\rceil$. The algorithm constructs the sets $S_1,\ldots,S_m\subseteq[t]$ in iterations. In each iteration $i\in[m]$ the algorithm finds S_i such that $\sum_{j<i}2^{|S_i\cap S_j|}\leq(i-1)\cdot\rho$. To do so, the algorithm initially fixes a partition of $[t]$ into ℓ blocks. The first $t-\ell$ blocks, denoted $B_1,\ldots,B_{t-\ell}$, are each comprised of two elements (i.e., for $j\in[t-\ell]$ it holds that $B_j=\{2j-1,2j\}$). The remaining $2\ell-t$ blocks, denoted $B_{t-\ell+1},\ldots,B_\ell$, each consist of a single element (i.e., for $j\in\{t-\ell+1,\ldots,\ell\}$ it holds that $B_j=\{t-\ell+j\}$).

For $i \in [m]$, let us describe the i^{th} iteration, after $S_1, ..., S_{i-1}$ were already chosen in previous iterations. Consider a set S_i that is chosen by independently choosing one random element from each of the ℓ blocks to include in S_i. [42] For $j \in [i-1]$ and $k \in [\ell]$, let $Y_{j,k}$ be the indicator variable of whether the element from the k^{th} block that is included in S_j is also included in S_i (i.e., $Y_{j,k} = 1$ iff $B_k \cap S_j \cap S_i \neq \emptyset$). Note that for $k \neq k' \in [m]$ it holds that $Y_{j,k}$ and $Y_{j,k'}$ are independent. Thus, the expected value of $\sum_{j<i} 2^{|S_i \cap S_j|}$ is

$$\begin{aligned}
\mathbb{E}\left[\sum_{j<i} 2^{|S_i \cap S_j|}\right] &= \sum_{j<i} \mathbb{E}\left[2^{\sum_{k\in[\ell]} Y_{j,k}}\right] \\
&= \sum_{j<i} \mathbb{E}\left[\prod_{k\in[\ell]} 2^{Y_{j,k}}\right] \\
&= \sum_{j<i} \prod_{k\in[\ell]} \mathbb{E}\left[2^{Y_{j,k}}\right] \\
&= (i-1)\cdot(3/2)^{t-\ell}\cdot 2^{2\ell-t}\,, \qquad (3.5.4)
\end{aligned}$$

where the last equality is because for every $k \in [t-\ell]$ it holds that $\Pr[Y_{j,k} = 1] = 1/2$ (since $|B_k| = 2$), and for every $k \in \{t-\ell+1, ..., \ell\}$ it holds that $Y_{j,k} \equiv 1$ (since B_k is a singleton). Now, plugging-in $t = \lceil (1-4\alpha)\cdot\ell \rceil$ and $\ell = \frac{\log(\rho)}{1-\alpha}$ into Eq. (3.5.4), we can upper-bound the expression by $(i-1)\cdot\rho$. [43] Hence, the algorithm can find a set S_i such that $\sum_{j<i} 2^{|S_i \cap S_j|} \leq (i-1)\cdot\rho$ by trying out all $2^{t-\ell} < 2^{\ell}$ possibilities. ■

We now instantiate Theorem 3.5.29 with the code from Corollary 3.5.23 and the weak designs from Lemma 3.5.30 in order to construct uniform sparse $\mathcal{CC}^0[2]$ circuits that computes the following averaging sampler: The sampler gets an input of length n, and two parameters $0 < \gamma \ll \beta < 1$, and constructs a sampler that outputs $m = n^\gamma$ bits and has accuracy $1/m$ and error $2^{n^\beta - n}$.

Theorem 3.5.31 (an averaging sampler in superlinear $\mathcal{CC}^0$)**.** *For any sufficiently large constant $r > 1$, there exists a polynomial-time algorithm that gets as input 1^n and two parameters $\beta = \beta(n) > 3/4$ and $\gamma = \gamma(n) < \frac{\beta - 3/4}{r}$, and outputs a $\mathcal{CC}^0[2]$ circuit C that satisfies the following:*

1. *The circuit C gets input $x \in \{0,1\}^n$ and outputs $2^t < n^{5-4(\beta - r\gamma)} \cdot \exp(\text{poly}\log\log(n))$ strings of length $m = n^\gamma$.*
2. *The function $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ such that $\mathsf{Samp}(x,i) = C(x)_i$ is an averaging sampler with accuracy $\epsilon = 1/m$ and error $2^{n^\beta - n}$.*

[42] That is, for each $k \in [\ell]$ let X_k be a random element from the block B_k, such that for $k \neq k' \in [\ell]$ it holds that X_k and $X_{k'}$ are independent. Then, $S_i = \cup_{k\in[\ell]} X_k$.

[43] Denoting $c = \log(e)/2$ and $t = (1+4\beta)\cdot\ell$, where $\beta \geq \alpha$, we have that $2^{2\ell-t}\cdot(3/2)^{t-\ell} < 2^{2\ell-t}\cdot e^{(t-\ell)/2} = 2^{2\ell - t + c\cdot(t-\ell)} \leq 2^{\frac{1-4(1-c)\cdot\beta}{1-\alpha}\cdot\log(\rho)} < \rho$.

3. *The depth of C is three and its number of wires is $n^{5-4(\beta-r\gamma)+\gamma} \cdot \exp(\text{poly}\log\log(n))$.*

In particular, if $\beta \geq 1 - r\gamma$, then the number of outputs of C is $2^t \leq n^{1+8r\cdot\gamma} \cdot \exp(\text{poly}\log\log(n))$, and its number of wires is at most $n^{1+(8r+1)\cdot\gamma} \cdot \exp(\text{poly}\log\log(n))$.

Proof. Let $r_2 > 1$ be the constant from Corollary 3.5.23, let $r_3 = 3 \cdot r_2$, and let $r \geq r_3 + 2$. We first use Corollary 3.5.23 with the parameter value $\delta = \epsilon/4m$ to construct a depth-two circuit C_0 that encodes its input $x \in \{0,1\}^n$ to a codeword $\bar{x}$ of length $\bar{n} = n \cdot \exp(\text{poly}\log\log(n)) \cdot (1/\delta)^{r_2}$. Then, we use Lemma 3.5.30 to construct an (m, ℓ, t, ρ) weak design $S_1, ..., S_m \subseteq [t]$ with the following parameters: For $\alpha = 1 - \beta + r \cdot \gamma < 1/4$ (the inequality is since $\beta > 3/4$ and $\gamma < (\beta - 3/4)/r$), we construct a design with $\ell = \log(\bar{n})$ and $\rho = 2^{(1-\alpha)\cdot\ell}$ and $t = \lceil (1 + 4\alpha) \cdot \ell \rceil$. Now, define a function $\texttt{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ as in Theorem 3.5.29; that is, for $x \in \{0,1\}^n$ and $z \in \{0,1\}^t$, the m-bit string $\texttt{Ext}(x,z)$ is the projection of $\bar{x}$ to the coordinates $z_{S_1}, ..., z_{S_m}$. The circuit C outputs the 2^t strings corresponding to $\{\texttt{Ext}(x,z)\}_{z\in\{0,1\}^t}$, where each output string is a projections of m bits of $\bar{x}$.

Let $k = n^\beta$. By our choice of α, we have that $\rho = 2^{(1-\alpha)\cdot\ell} < k/2m < \frac{k-3\cdot\log(m/\epsilon)-t-3}{m}$.[44] Thus, relying on Theorem 3.5.29, the function $\texttt{Ext}$ is an $(n^\beta, \epsilon = 1/m)$-extractor, and also (by Proposition 2.5.6) a sampler with accuracy $\epsilon = 1/m$ and error $2^{n^\beta - n}$. The number of wires in C_0 is at most $n \cdot \exp(\text{poly}\log\log(n)) \cdot m^{r_3}$, and the number of wires between $\bar{x}$ and the outputs is $2^t \cdot m = 2^{\lceil(1+4\alpha)\cdot\log(\bar{n})\rceil} \cdot n^\gamma \leq n^{5-4\beta+(4r+1)\gamma} \cdot \exp(\text{poly}\log\log(n))$. Hence, the total number of wires in the sampler is at most $n^{5-4\beta+(4r+1)\gamma} \cdot \exp(\text{poly}\log\log(n))$. ■

We now construct averaging samplers in uniform extremely sparse $\mathcal{TC}^0$. Similarly to Theorem 3.5.31, we use Theorem 3.5.29 and Lemma 3.5.30, but this time with the code construction in $\mathcal{TC}^0$ from Corollary 3.5.27. The sampler will get an input of length n, and for two constants $0 < \gamma \ll \beta < 1$, the sampler will output $m = n^\gamma$ bits and have accuracy $1/m$ and error $2^{n^\beta - n}$.

Theorem 3.5.32 (an averaging sampler in extremely sparse $\mathcal{TC}^0$)**.** *For a universal constant $c_0 > 1$ and any sufficiently large constant $r > 1$, there exists a polynomial-time algorithm that gets as input 1^n and three constants $d \geq 2$ and $\beta > 3/4$ and $\gamma < \frac{\beta-3/4}{r}$, and outputs a $\mathcal{TC}^0$ circuit C that satisfies the following:*

1. *The circuit C gets input $x \in \{0,1\}^n$ and outputs $2^t < n^{(1+4r\cdot\gamma)\cdot(5-4\beta)}$ strings of length $m = n^\gamma$.*
2. *The function $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ such that $\mathsf{Samp}(x,i) = C(x)_i$ (i.e., $\mathsf{Samp}(x,i) \in \{0,1\}^m$ is the i^{th} output string of $C(x)$) is an averaging sampler with accuracy $\epsilon = 1/m$ and error $2^{n^\beta - n}$.*

[44] To see that $(1-\alpha)\cdot\ell < \log(k/2m)$, note the following. Since $\ell = \log(\bar{n}) < (1 + o(1)) \cdot \log(n) + r_3 \cdot \log(m)$, we have that $(1-\alpha)\cdot\ell \leq (\beta - r\cdot\gamma) \cdot ((1+o(1)) \cdot \log(n) + r_3 \cdot \log(m)) < (\beta - r\cdot\gamma + o(1)) \cdot \log(n) + r_3 \cdot \log(m)$. On the other hand, we have that $\log(k/2m) = \beta \cdot \log(n) - \log(2m)$. Thus, it suffices to prove that $(r\cdot\gamma - o(1)) \cdot \log(n) - r_3 \cdot \log(m) \geq \log(2m)$, which holds since $n = m^{1/\gamma}$ and $r - r_3 > 1$.

3. *The depth of C is $d+5$ and its number of wires is $n^{1+c_0\cdot\phi^{-d}+o(1)}+O\left(n^{(1+r\cdot\gamma)\cdot(5-4\beta)+8r\cdot\gamma}\right)$, where $\phi=\frac{1+\sqrt{5}}{2}$.*

In particular, if $\gamma\leq 1/(24r\cdot\phi^d)$ and $\beta\geq 1-\phi^{-d}/2$, then both the number of outputs of C (i.e., 2^t) and the number of wires in C are bounded by $n^{1+c_0\cdot\phi^{-d}+o(1)}$.

Proof. The parametrization of the sampler in this proof is similar to that in the proof of Theorem 3.5.31. Specifically, let $r_2>1$ and $c_0>1$ be the constants from Corollary 3.5.27, let $r_3=3\cdot r_2$, and let $r\geq r_3+2$. We use Corollary 3.5.27 with $\delta=\epsilon/4m$ to construct a circuit C_0 of depth $d+4$ that encodes its input $x\in\{0,1\}^n$ to a codeword $\bar{x}$ of length $\bar{n}=n\cdot\exp(\operatorname{poly}\log\log(n))\cdot(1/\delta)^{r_2}<n^{1+o(1)}\cdot m^{r_3}$. Then, for $\alpha=1-\beta+r\cdot\gamma<1/4$, we use Lemma 3.5.30 to construct an (m,ℓ,t,ρ) weak design $S_1,\ldots,S_m\subseteq[t]$ with $\ell=\log(\bar{n})$ and $\rho=2^{(1-\alpha)\cdot\ell}$ and $t=\lceil(1+4\alpha)\cdot\ell\rceil$. Let $\mathtt{Ext}:\{0,1\}^n\times\{0,1\}^t\to\{0,1\}^m$ such that for $x\in\{0,1\}^n$ and $z\in\{0,1\}^t$, the m-bit string $\mathtt{Ext}(x,z)$ is the projection of $\bar{x}$ to the coordinates $z_{S_1},\ldots,z_{S_m}$. The circuit C outputs the 2^t strings $\{\mathtt{Ext}(x,z)\}_{z\in\{0,1\}^t}$.

Let $k=n^\beta$, and note that $\rho=2^{(1-\alpha)\cdot\ell}<k/2m<\frac{k-3\cdot\log(m/\epsilon)-t-3}{m}$. Relying on Theorem 3.5.29 and Proposition 2.5.6, the function $\mathtt{Ext}$ is a sampler with accuracy $\epsilon=1/m$ and error $2^{n^\beta-n}$. The depth of C is $d+5$ (since the depth of C_0 is $d+4$, and the 2^t outputs are projections of $\bar{x}$). Finally, the number of wires in C_0 is at most $n^{1+c_0\cdot\phi^{-d}+o(1)}+n^{1+o(1)}\cdot m^{r_3}<n^{1+c_0\cdot\phi^{-d}+o(1)}+n^{1+\gamma\cdot r}$, and the number of wires between $\bar{x}$ and the outputs is $2^t\cdot m=2^{\lceil(1+4\alpha)\cdot\log(\bar{n})\rceil}\cdot n^\gamma<n^{(1+r\cdot\gamma)\cdot(5-4\beta+4r\cdot\gamma)}<n^{(1+r\cdot\gamma)\cdot(5-4\beta)+8r\cdot\gamma}$. Hence, the total number of wires in the sampler is $n^{1+c_0\cdot\phi^{-d}+o(1)}+O\left(n^{(1+r\cdot\gamma)\cdot(5-4\beta)+8r\cdot\gamma}\right)$. ■

3.5.4.3 Using the averaging sampler to prove Theorem 3.5.2

Let us now formally state Theorem 3.5.2 and prove it using the averaging sampler from Theorem 3.5.32.

Theorem 3.5.33 (Theorem 3.5.2, restated)**.** *For $\phi=\frac{1+\sqrt{5}}{2}$, there exists a universal constant $c_0>1$ such that for any sufficiently large constant $r>1$ the following holds.*

Let $d\in\mathbb{N}$. Assume that for some $d'\geq d+7$ there exists an algorithm that gets as input a $\mathcal{TC}^0$ circuit $C':\{0,1\}^n\to\{0,1\}$ with depth d' and $n^{1+c_1\cdot\phi^{-d'}}$ wires, where $c_1=c_0\cdot\phi^{d+5}+1/r$, runs in time $T(n)$, and for $\beta=1-\phi^{-d'}$ satisfies the following: If C' rejects all but at most 2^{n^β} of its inputs, then the algorithm rejects C', and if C' accepts all but at most 2^{n^β} of its inputs, then the algorithm accepts C'.

Then, there exists an algorithm that for every $k\in\mathbb{N}$, when given as input a $\mathcal{TC}^0$ circuit $C:\{0,1\}^m\to\{0,1\}$ with depth d and m^k wires, runs in time $T(m^{24r\cdot k\cdot\phi^{d'}})$, and satisfies the following: If C accepts at least $2/3$ of its inputs then the algorithm accepts C, and if C rejects at least $2/3$ of its inputs then the algorithm rejects C.

Recall that in Theorem 3.5.2 the hypothesis is that for $c<\phi$ (e.g., $c=1.61$) and every d', the quantified derandomization algorithm will be able to handle circuits

with depth d' and $n^{1+c^{-d'}}$ wires. This hypothesis is stronger than the hypothesis in Theorem 3.5.33, since for any fixed $d \in \mathbb{N}$ and sufficiently large d' it holds that $c^{-d'} > c_1 \cdot \phi^{-d'}$.

Proof of Theorem 3.5.33. Let $c_0 > 1$ be the universal constant from Theorem 3.5.32, and assume that $r > 1$ is sufficiently large to satisfy the hypothesis of Theorem 3.5.32. We construct an algorithm that gets as input a $\mathcal{TC}^0$ circuit $C : \{0,1\}^m \to \{0,1\}$ with depth d and m^k wires, and acts as follows. For $\gamma = 1/(24r \cdot k \cdot \phi^{d'})$ and $n = m^{1/\gamma}$, the algorithm constructs a circuit $C' : \{0,1\}^n \to \{0,1\}$ of depth d' with $n^{1+c_1 \cdot \phi^{-d'}}$ wires such that the following holds: If C rejects at least a $2/3$ fraction of its inputs, then C' rejects all but at most 2^{n^β} inputs; and if C accepts at least a $2/3$ fraction of its inputs, then C' accepts all but 2^{n^β} of its inputs. Then, the algorithm invokes the quantified derandomization algorithm for C', which runs in time $T(n) = T\left(m^{24r \cdot k \cdot \phi^{d'}}\right)$, to decide whether the acceptance probability of C is at least $2/3$ or at most $1/3$.

To construct C', let $d_{\mathsf{Samp}} = d' - d - 5 \geq 2$; we first use Theorem 3.5.32 to construct a $\mathcal{TC}^0$ circuit $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ that is an averaging sampler with the following properties: The input length is n, the output length is $m = n^\gamma$, the accuracy is $\epsilon = n^{\Omega(1)} < 1/100$, and the error is $\delta = 2^{n^\beta - n}$; the depth of Samp is $d_{\mathsf{Samp}} + 5$, and by the "in particular" part of Theorem 3.5.32 (relying on the facts that $\beta = 1 - \phi^{-d'} > 1 - \phi^{-d_{\mathsf{Samp}}}/2$ and that $\gamma = 1/(24r \cdot k \cdot \phi^{d'}) < 1/(24r \cdot \phi^{d_{\mathsf{Samp}}})$, the number of wires in Samp is bounded by $n^{1+c_0 \cdot \phi^{-d_{\mathsf{Samp}}} + o(1)}$. The circuit C' first computes the sampler Samp, then evaluates C in parallel on each of the $2^t < n^{1+c_0 \cdot \phi^{-d_{\mathsf{Samp}}} + o(1)}$ outputs of the sampler, and finally computes the majority of the 2^t evaluations of C. That is, $C'(x) = MAJ_{z \in \{0,1\}^t}[C(\mathsf{Samp}(x,z))]$. The circuit C' is of depth $(d_{\mathsf{Samp}} + 5) + d + 1$, but the gates in one of its layers (i.e., the output layer of the sampler) are just projections of the gates in the layer beneath it; therefore, we can collapse one layer, and obtain an equivalent circuit of depth $d' = d_{\mathsf{Samp}} + d + 5$. The number of wires in C' is at most

$$\begin{aligned} n^{1+c_0 \cdot \phi^{-d_{\mathsf{Samp}}} + o(1)} + 2^t \cdot m^k + 2^t &< n^{1+c_0 \cdot \phi^{-d_{\mathsf{Samp}}} + o(1)} + n^{1+c_0 \cdot \phi^{-d_{\mathsf{Samp}}} + 1/(24r \cdot \phi^{d'}) + o(1)} \\ &< n^{1+\left(c_0 \cdot \phi^{d+5} + 1/r\right) \cdot \phi^{-d'}}, \end{aligned}$$

where we relied on the facts that $m^k = n^{1/(24r \cdot \phi^{d'})}$ and that $\phi^{-d_{\mathsf{Samp}}} = \phi^{d+5} \cdot \phi^{-d'}$.

Finally, note that for any $x \in \{0,1\}^n$ such that $\Pr_{z \in \{0,1\}^t}[C(\mathsf{Samp}(x,z)) = 1] \in \Pr[C(\mathbf{u}_n) = 1] \pm \epsilon$, we have that $C'(x)$ outputs the most frequent value of C. Since the error of the sampler is $\delta = 2^{n^\beta - n}$, the number of inputs $x \in \{0,1\}^n$ such that $\Pr_{z \in \{0,1\}^t}[C(\mathsf{Samp}(x,z)) = 1] \notin \Pr[C(\mathbf{u}_n) = 1] \pm \epsilon$ is at most 2^{n^β}. Thus, the circuit C' outputs the most frequent value of C on all but at most 2^{n^β} inputs $x \in \{0,1\}^n$. ∎

Relying on known relaxations of Williams' "algorithmic method" (see [Wil13; SW13; BSV14; FS16; MW18]), we obtain the following corollary of Theorem 3.5.33:

Corollary 3.5.34 (quantified derandomization of sparse $\mathcal{TC}^0$ implies lower bounds for $\mathcal{TC}^0$)**.** *There exists a constant $\epsilon > 0$ such that the following holds. Let $c > 1$ be any fixed constant smaller than $\frac{1+\sqrt{5}}{2}$. Assume that* if $\mathcal{NEXP} \subseteq \mathcal{TC}^0$, *then for every $d \in \mathbb{N}$ there exists a* non-deterministic *machine that gets as input a $\mathcal{TC}^0$ circuit $C : \{0,1\}^n \to \{0,1\}$ with depth d and $n^{1+c^{-d}}$ wires, and also n^ϵ* bits of non-uniform advice, *runs in time $2^{n^{o(1)}}$, and solves the following problem: If C accepts all of its inputs, then there exist non-deterministic choices that cause the machine to accept C; and if C rejects all but $B(n) = 2^{n^{1-c^{-d}}}$ of its inputs, then the machine rejects C regardless of the non-determinism. Then, $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$.*

3.5.4.4 On quantified derandomization of $\mathcal{ACC}^0$

In this section we prove that a algorithm for quantified derandomization of $\mathcal{ACC}^0$ circuits with $n^{1+\Omega(1)}$ wires and with a subexponential $B(n)$ that runs in time $2^{n^{o(1)}}$ would yield a corresponding algorithm for standard derandomization of $\mathcal{ACC}^0$ with "one-sided error" that runs in time $2^{n^{o(1)}}$.

The proof strategy is similar to that of the proof of Theorem 3.5.2. Specifically, we construct an algorithm that gets as input an $\mathcal{ACC}^0$ circuit $C : \{0,1\}^m \to \{0,1\}$, and outputs an $\mathcal{ACC}^0$ circuit $C' : \{0,1\}^n \to \{0,1\}$ such that if C has acceptance probability one then C' has acceptance probability one, and if C has acceptance probability at most half then C' rejects all but $B(n)$ of its inputs. To do so, the algorithm first constructs a uniform sparse $\mathcal{CC}^0[\oplus]$ circuit that computes an averaging sampler; this is done relying on the construction that was presented in Section 3.5.4.2. Then, for $n = \text{poly}(m)$, the algorithm constructs $C' : \{0,1\}^n \to \{0,1\}$ that first uses its input to sample inputs for C using the foregoing averaging sampler, then evaluates C on the inputs in the sample, and finally outputs the conjunction of the latter evaluations of C.[45]

Note that our transformation of C to C' only involves adding $\oplus$ gates and a single AND gate (at the top). Thus, our construction actually works not only for $\mathcal{ACC}^0$ circuits, but for essentially any circuit class whose gates can compute the AND and $\oplus$ functions (e.g., it also works for $\mathcal{AC}^0[\oplus]$). Specifically:

Theorem 3.5.35 (a bootstrapping result for quantified derandomization of $\mathcal{ACC}^0$)**.** *Let $\mathcal{C}$ be any typical circuit class whose gates can compute the AND and $\oplus$ functions. Let $d \in \mathbb{N}$, and let $\gamma_d < 1/4r$, where $r > 1$ is a universal constant. Assume that there exists an algorithm that gets as input a $\mathcal{C}$-circuit $C' : \{0,1\}^n \to \{0,1\}$ with depth $d' = d + 3$ and $O\left(n^{1+(8r+2)\cdot\gamma_d}\right) \cdot \exp(\text{poly}\log\log(n))$ wires, runs in time $T(n)$, and for $\beta = 1 - r \cdot \gamma_d$ satisfies the following: If C' rejects all but at most 2^{n^β} of its inputs, then the algorithm rejects C', and if C' accepts all of its inputs, then the algorithm accepts C'. Then, there exists an algorithm that for every $k \in \mathbb{N}$, when given as input a $\mathcal{C}$-circuit $C : \{0,1\}^m \to \{0,1\}$ with depth d and m^k wires, runs in time $T(m^{k/\gamma_d})$, and satisfies the following: If C accepts all of*

[45] The reason that we use a top AND gate, instead of a sub-circuit for approximate majority, is that the known constructions for the latter circuit are of polynomial size (i.e., are not super-linear). Indeed, this is the reason that our result is limited to derandomization with "one-sided error".

its inputs then the algorithm accepts C, and if C rejects at least half of its inputs then the algorithm rejects C.

We comment that Theorem 3.5.35 holds also when γ_d is a *sub-constant* function of n (rather than a constant that depends only on d), albeit with a more complicated expression for the running time of the algorithm for standard derandomization in the conclusion of the theorem. We defer the discussion of this point until after the proof.

Proof of Theorem 3.5.35. Assume that $r > 1$ is sufficiently large to satisfy the hypothesis of Theorem 3.5.31. Our algorithm gets as input a $\mathcal{C}$-circuit $C : \{0,1\}^m \to \{0,1\}$ with depth d and m^k wires, and constructs a corresponding $\mathcal{C}$-circuit C', as follows.

Let n be the minimal integer such that $n^{\gamma_d(n)/k} \geq m$. The algorithm first uses Theorem 3.5.31 to construct a depth-three $\mathcal{CC}^0[2]$ circuit $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ that is an averaging sampler with the following properties: For $\gamma' = \gamma_d(n)$ and $\beta' = 1 - r \cdot \gamma'$, the input length is $n = m^{k/\gamma'}$, the output length is m, the accuracy is $\epsilon = n^{\Omega(1)} < 1/2$, and the error is $\delta = 2^{n^{\beta'}-n}$; by the "in particular" part of Theorem 3.5.31, the number of wires in Samp is bounded by $n^{1+(8r+1)\cdot\gamma'}$. Now, the circuit C' first computes the sampler Samp, then evaluates C in parallel on each of the $2^t < n^{1+8r\cdot\gamma'}$ outputs of the sampler, and finally computes the conjunction of the 2^t evaluations of C. That is, $C'(x) = \mathsf{AND}_{z\in\{0,1\}^t}[C(\mathsf{Samp}(x,z))]$.

The circuit C' is of depth $d+4$, but the gates in one of its layers (i.e., the output layer of Samp) just compute projections of the layer beneath it, so the algorithm can collapse this layer to obtain an equivalent circuit of depth $d' = d+3$. The number of wires in C' is $O\left(n^{1+(8r+2)\cdot\gamma'}\right)$. Also, we only added gates that compute AND and $\oplus$ functions, and therefore $C' \in \mathcal{C}$. Lastly, if C accepts all of its inputs then C' also accepts all of its inputs, and if C rejects at least half of its inputs, then C' accepts all but at most $2^{n^{\beta'}} > 2^{n^{\beta}}$ of its inputs (the inequality is since $2^{n^{\beta'}} = 2^{n^{1-r\cdot\gamma'}} > 2^{n^{1-r\cdot\gamma_d(n)}}$, relying again on the fact that $\gamma_d(n) > \gamma'$). The latter statement is since if C rejects at least half of its inputs, then for all but $\delta \cdot 2^n = 2^{n^{\beta'}}$ of the inputs $x \in \{0,1\}^n$ to Samp it holds that $\Pr_{z\in\{0,1\}^t}[C(\mathsf{Samp}(x,z)) = 1] \geq \Pr[C(\mathbf{u}_n) = 1] - \epsilon > 0$; and for every $x \in \{0,1\}^n$ such that the latter holds we have that $C'(x) = 1$.

Therefore, our algorithm can now invoke the quantified derandomization algorithm for C', which runs in time $T(n)$, to decide whether the acceptance probability of C is 1 or at most $1/2$. ■

As mentioned after the statement of Theorem 3.5.35, the theorem holds also when γ_d is a sub-constant function of n, where the only change is that the running time of the algorithm for standard derandomization (in the conclusion of the theorem) will be larger. Specifically, given a function $\gamma_d(n)$ in the hypothesis, in the proof we set n to be the minimal integer such that $n^{\gamma_d(n)/k} \geq m$; then, the running time of the algorithm for standard derandomization will be $T(n)$ (rather than $T(m^{k/\gamma_d})$ as in Theorem 3.5.35). Note that when $\gamma_d(n) = o(1)$, the algorithm in the hypothesis of the theorem only

needs to handle circuits with $n^{1+o(1)}$ wires; in particular, when $\gamma_d(n) = \frac{\text{poly}\log\log(n)}{\log(n)}$, the algorithm only needs to handle circuits of size $n \cdot \exp(\text{poly}\log\log(n))$.

3.5.5 Depth-2 linear threshold circuits

In this section we construct a quantified derandomization algorithm for depth-2 linear threshold circuits with $n^{3/2-\Omega(1)}$ wires. In fact, we construct a *pseudorandom generator* for the class of depth-2 linear threshold circuits with $n^{3/2-\Omega(1)}$ wires that either accept all but $B(n) = 2^{n^{\Omega(1)}}$ of their inputs or reject all but $B(n)$ of their inputs. That is, we construct an algorithm G that gets as input a seed s of length $\tilde{O}(\log(n))$, and outputs an n-bit string such that for every $C \in \mathcal{C}_{n,2,n^{3/2-\Omega(1)}}$ the following holds: If C accepts all but $B(n) = 2^{n^{\Omega(1)}}$ of its inputs, then the probability that $C(G(s)) = 1$ is very high, and if C rejects all but $B(n)$ of its inputs, then the probability that $C(G(s)) = 0$ is very low.

The pseudorandom generator that we construct in this appendix is incomparable to the pseudorandom generator of Servedio and Tan [ST17b]. On the one hand, their generator is $\frac{1}{\text{poly}(n)}$-pseudorandom for *every* sparse depth-two linear threshold circuit, whereas our generator only "fools" sparse depth-two circuits with acceptance probability that is either very high or very low. Moreover, their generator can handle circuits with $n^{2-\Omega(1)}$ wires, whereas our generator can only handle circuits with $n^{3/2-\Omega(1)}$ wires. But on the other hand, their generator requires a seed of length $n^{1-\Omega(1)}$, whereas our generator only requires a seed of length $\tilde{O}(\log(n))$.

Recall that our main quantified derandomization algorithm (from Theorem 3.5.1) leverages the techniques underlying the average-case lower bounds of Chen, Santhanam, and Srinivasan [CSS16] for depth-d linear threshold circuits. The generator in this section leverages the techniques underlying the average-case lower bounds of Kane and Williams [KW16] for depth-2 linear threshold circuits.

Specifically, our first step is to prove a derandomized version of the restriction lemma of Kane and Williams [KW16]. We actually state a slightly generalized version, which is implicit in the original argument. We say that a distribution $\mathbf{y}$ over $\{0,1\}^n$ is p-bounded in pairs if for every $i \neq j \in [n]$ it holds that $\Pr[\mathbf{y}_i = 1] \leq p$ and $\Pr[\mathbf{y}_i = 1 \wedge \mathbf{y}_j = 1] \leq p^2$. One example for a distribution that is p-bounded in pairs is the distribution $\mathbf{y}$ in which each coordinate is independently set to 1 with probability p. Another example, which is used in [KW16], is the following: Consider a equipartition of $[n]$ to $p \cdot n$ disjoint sets $S_1, ..., S_{p \cdot n}$; then, sampling $y \sim \mathbf{y}$ is equivalent to uniformly choosing a single coordinate in each set S_i in the partition, fixing y in the chosen coordinates to one, and fixing y in all other coordinates to zero (so that the Hamming weight of $y \sim \mathbf{y}$ is always $p \cdot n$).

Proposition 3.5.36 (derandomized version of [KW16, Lem 3.1])**.** *Let $\Phi = (w, \theta)$ be an LTF on m input bits. For $p > 0$, let $\mathbf{y}$ be a distribution over $\{0,1\}^n$ that is p-bounded in pairs, and let $\mathbf{z}$ be a distribution over $\{-1,1\}^n$ that is $\frac{1}{\text{poly}(m)}$-pseudorandomly concentrated. Let $\boldsymbol{\rho}$ be the distribution over restrictions obtained by sampling $y \sim \mathbf{y}$ in order to determine which variables are kept alive (the i^{th} variable is kept alive if and only if $\mathbf{y}_i = 1$), and independently*

sampling $z \sim \mathbf{z}$ to determine values for the fixed variables. Then,

$$\Pr_{\rho\sim\boldsymbol{\rho}}[\Phi\restriction_\rho \textit{ depends on more than one input bit}] = O(m \cdot p^{3/2}) \,.$$

Proof. For every choice of $y \sim \mathbf{y}$, let $I = I_y \subseteq [n]$ be the set of live variables (i.e., $I = \{i \in [n] : y_i = 1\}$). Then, the probability that $\Phi\restriction_\rho$ depends on more than one input bit is at most

$$\begin{aligned}
&\Pr_{\rho\sim\boldsymbol{\rho}}\left[|I| \geq 2 \wedge \Phi\restriction_\rho \text{ is not constant}\right] \\
&= \mathbb{E}_{y\sim\mathbf{y}}\left[\Pr_{z\sim\mathbf{z}}\left[|I| \geq 2 \wedge \Phi\restriction_\rho \text{ is not constant}\right]\right] \\
&= \mathbb{E}_{y\sim\mathbf{y}}\left[\mathbf{1}_{|I|\geq 2} \cdot \Pr_{z\sim\mathbf{z}}\left[\Phi\restriction_\rho \text{ is not constant}\right]\right] \,,
\end{aligned} \tag{3.5.5}$$

where the first equality relied on the fact that y and z are sampled independently, and the second equality is since the random variable I only depends on y (and not on z).

Fix an arbitrary choice of y, and let us upper-bound the probability over $z \sim \mathbf{z}$ that $\Phi\restriction_\rho$ is not constant. Note that $\Phi\restriction_\rho$ is a constant function if and only if

$$\left|\theta - \left\langle w_{[m]\setminus I}, z_{[m]\setminus I}\right\rangle\right| > \|w_I\|_1 \iff \left\langle w_{[m]\setminus I}, z_{[m]\setminus I}\right\rangle \notin \theta \pm \|w_I\|_1 \,. \tag{3.5.6}$$

For each $i \in [m]$, let k_i be the index of the i^{th} variable when the variables are sorted according to the magnitudes $|w_i|$ in ascending order (breaking ties arbitrarily). In [KW16, Proof of Lemma 1.1] it is shown that the probability over a *uniform choice of* z that Eq. (3.5.6) holds is at most $\sum_{i\in I} \frac{O(1)}{\sqrt{k_i}}$. Since $\mathbf{z}$ is $(1/\mathrm{poly}(m))$-pseudorandomly concentrated, the probability under $z \sim \mathbf{z}$ that Eq. (3.5.6) holds is at most $\sum_{i\in I} \frac{O(1)}{\sqrt{k_i}} + \frac{1}{\mathrm{poly}(m)}$. Therefore, the expression in Eq. (3.5.5) is upper-bounded by

$$\begin{aligned}
&\mathbb{E}_{y\sim\mathbf{y}}\left[\mathbf{1}_{|I|\geq 2} \cdot \sum_{i\in I} \frac{O(1)}{\sqrt{k_i}}\right] + \frac{1}{\mathrm{poly}(m)} \\
&= \mathbb{E}_{y\sim\mathbf{y}}\left[\sum_{i\in[m]} \frac{O(1)}{\sqrt{k_i}} \cdot \mathbf{1}_{i\in I \wedge |I|\geq 2}\right] + \frac{1}{\mathrm{poly}(m)} \\
&= \sum_{i\in[m]} \frac{O(1)}{\sqrt{k_i}} \cdot \Pr_{y\sim\mathbf{y}}\left[i \in I \wedge |I| \geq 2\right] + \frac{1}{\mathrm{poly}(m)} \,.
\end{aligned} \tag{3.5.7}$$

For any fixed $i \in [m]$, we upper-bound the probability of the event $i \in I \wedge |I| \geq 2$ in two ways: The first upper-bound is $\Pr[i \in I] \leq p$, and the second upper-bound is $\Pr[\exists j \in [m] \setminus \{i\}, j \in I \wedge i \in I] < m \cdot p^2$ (since $\mathbf{y}$ is p-bounded in pairs). Hence,

$$\Pr_{y\sim\mathbf{y}}\left[i \in I \wedge |I| \geq 2\right] \leq \min\left\{p, m \cdot p^2\right\} \leq \sqrt{m \cdot p^3} \,,$$

which implies that the expression in Eq. (3.5.7) is upper-bounded by

$$\sqrt{m \cdot p^3} \cdot \sum_{i \in [m]} \frac{O(1)}{\sqrt{k_i}} + \frac{1}{\mathrm{poly}(m)} = O\left(\sqrt{m} \cdot p^{3/2} \cdot \sum_{i \in [m]} \frac{1}{\sqrt{i}}\right) = O\left(m \cdot p^{3/2}\right) . \qquad \blacksquare$$

Our pseudorandom generator, which is contructed next, is based on an application of Proposition 3.5.36 as well as on the pseudorandom generator of Gopalan, Kane, and Meka (i.e., Theorem 2.4.20).

Theorem 3.5.37 (quantified derandomization of depth-2 linear threshold circuits with $n^{3/2-\Omega(1)}$ wires)**.** *There exists a polynomial-time algorithm G that is given as input a random seed s of length $\tilde{O}(\log(n))$ and a constant $\epsilon > 0$, and outputs a string $G(s, \epsilon) \in \{0,1\}^n$ such that for every $C \in \mathcal{C}_{n,2,n^{3/2-\epsilon}}$ the following holds:*

1. *If C accepts all but at most $B(n) = 2^{n^{\epsilon/2}}$ inputs, then $\Pr_s[C(G(s,\epsilon)) = 1] = 1 - o(1)$.*
2. *If C rejects all but at most $B(n)$ inputs, then $\Pr_s[C(G(s,\epsilon)) = 1] = o(1)$.*

Proof. Let $\delta \in (\epsilon/2, 2\epsilon/3)$ such that $p = n^{-(1-\delta)}$ is a power of two. The algorithm first samples a restriction that meets the requirements of Proposition 3.5.36, as follows: The distribution $\mathbf{y}$ over $\{0,1\}^n$ is obtained by sampling a string y' from a distribution over $\{0,1\}^{\log(1/p)\cdot n}$ that is $\frac{1}{\mathrm{poly}(n)}$-almost $O(\log(n))$-wise independent, and setting $\mathbf{y}_i = 1$ if and only if the i^{th} block in y' is all zeroes; and the distribution $\mathbf{z}$ is $\frac{1}{\mathrm{poly}(n)}$-pseudorandomly concentrated. The required seed length to sample such a restriction is dominated by the seed length required to sample $z \sim \mathbf{z}$, which (using Theorem 2.4.20) is $O(\log(n) \cdot (\log\log(n))^2)$.

We say that a restriction ρ is successful if the circuit $C\restriction_\rho$ can be computed by a single LTF, and if at least $\frac{1}{2} \cdot (p \cdot n) = \frac{1}{2} \cdot n^\delta$ variables remain alive under ρ. We first claim that the probability that ρ is successful is $1 - o(1)$. According to Fact 2.1.1, with probability $1 - 1/\mathrm{poly}(n)$ at least $\frac{1}{2} \cdot n^\delta$ variables remain alive under ρ. To see that with high probability $C\restriction_\rho$ can be computed by a single LTF, let $\mathcal{G}$ be the set of gates in the bottom layer of C. We say that a gate Φ is non-trivial if Φ depends on more than a single input bit; note that any trivial gate can be replaced by a constant or by an input bit (or its negation). Then, the expected number of non-trivial gates in the bottom layer of $C\restriction_\rho$ is

$$\begin{aligned}
\mathbb{E}_\rho\left[\sum_{\Phi \in \mathcal{G}} \mathbf{1}_{\Phi\restriction_\rho \text{ is non-trivial}}\right] &= \sum_{\Phi \in \mathcal{G}} \Pr_\rho[\Phi\restriction_\rho \text{ is non-trivial}] \\
&= O\left(\sum_{\Phi \in \mathcal{G}} \text{fan-in}(\Phi) \cdot p^{3/2}\right) \\
&= O\left(n^{3/2-\epsilon} \cdot n^{3\delta/2-3/2}\right) ,
\end{aligned}$$

which is $o(1)$, since $\delta < 2\epsilon/3$. Therefore, the probability that there are no non-trivial gates in the bottom layer of $C\restriction_\rho$ is $1 - o(1)$.

After sampling the restriction ρ, the algorithm samples a string $x \in \{0,1\}^{|\rho^{-1}(\star)|}$ using the pseudorandom generator G' for LTFs from Theorem 2.4.20, instantiated with error parameter $1/\mathrm{poly}(n)$, and outputs the n-bit string that is obtained by completing x to an n-bit string according to ρ.

To see that the algorithm is correct, assume that C accepts all but $2^{n^{\epsilon/2}}$ of its inputs. Then, for every successful restriction ρ, the acceptance probability of $C\restriction_\rho$ is $1 - o(1)$ (since ρ keeps at least $\frac{1}{2} \cdot n^\delta = \omega(n^{\epsilon/2})$ variables alive). Thus,

$$\begin{aligned}\Pr_s[C(G(s,\epsilon)) = 0] &\leq \Pr_\rho[\rho \text{ not successful}] + \Pr_s[C(G(s,\epsilon)) = 0 | \rho \text{ successful}] \\ &\leq o(1) + \max_{\rho \text{ successful}} \Pr_{s'}[C\restriction_\rho(G'(s')) = 0] \,,\end{aligned}$$

which is $o(1)$ since G' is $\frac{1}{\mathrm{poly}(n)}$-pseudorandom for LTFs. Similarly, if C rejects all but $2^{n^{\epsilon/2}}$ of its inputs, then $\Pr[C(G(s)) = 1] = o(1)$. ■

3.5.6 Appendices for Section 3.5

3.5.6.1 An alternative proof of Lemma 3.5.16

In this section we provide an alternative proof of Lemma 3.5.16, which asserts that biased LTFs remain biased when variables are fixed according to a distribution that is pseudorandom for LTFs. Loosely speaking, the following (alternative) formal statement of the lemma asserts the following: If an LTF Φ_i is δ-close to a constant function, then with probability $1 - \gamma$ over choice of $z \sim \mathbf{z}$ it holds that $\Phi_i\restriction_\rho$ is δ'-close to the same constant function, as long as $\delta \leq \mathrm{poly}(\delta', \gamma)$ and that $\mathbf{z}$ is $\mathrm{poly}(\gamma)$-pseudorandom for LTFs. More specifically:

Lemma 3.5.38 (Lemma 3.5.16, restated)**.** *Let $n \in \mathbb{N}$, and let $\delta, \delta', \gamma > 0$ such that $\delta \leq (\gamma \cdot \delta')^{10}$. Let $\Phi = (w, \theta)$ be an LTF over n input bits that is δ-close to a constant function $\sigma \in \{-1, 1\}$, let $I \subseteq [n]$, and let $\mathbf{z}$ be a distribution over $\{-1,1\}^{[n]\setminus I}$ that is $(\delta' \cdot \gamma^2)$-pseudorandom for LTFs. Then, with probability $1 - O(\gamma)$ over choice of $z \sim \mathbf{z}$ it holds that $\Phi\restriction_{(I,z)}$ is δ'-close to σ.*

A natural approach to prove Lemma 3.5.38 is the following. For any fixed choice of a set $I \subseteq [n]$ of variables to keep alive, we want to choose the values for the fixed variables from a *distribution that "fools" a test that checks whether or not $\Phi\restriction_\rho$ is close to σ*. That is, consider a test $T : \{-1,1\}^{[n]\setminus I} \to \{-1,1\}$ that gets as input values $z \in \{-1,1\}^{[n]\setminus I}$ for the fixed variables $[n] \setminus I$, and decides whether or not Φ remains close to σ in the subcube corresponding to $\rho = \rho_{I,z}$. When z is chosen uniformly, with high probability $\Phi\restriction_\rho$ remains close to σ, and hence the acceptance probability of T is high; thus, any distribution over $\{-1,1\}^{[n]\setminus I}$ that is pseudorandom for T also yields, with high probability, values $z \in \{-1,1\}^{[n]\setminus I}$ such that $\Phi\restriction_{\rho_{I,z}}$ remains close to σ. The *problem with this approach* is that a test T for such a task above might be very inefficient, since it needs to evaluate Φ on all points in the subcube corresponding to $\rho = \rho_{I,z}$;

thus, we might not be able to construct a pseudorandom generator with short seed to "fool" such a "complicated" test.

To solve this problem we use the *randomized tests* technique, and specifically Corollary 3.2.3. For convenience, let us restate the lemma now, while "translating" the $\{0,1\}$ notation into $\{-1,1\}$ notation, which is more consistent with the current context:

Lemma 3.5.39 (randomized tests; Corollary 3.2.3, restated)**.** *Let $n \in \mathbb{N}$, and let $\epsilon > 0$ be an error parameter.*

- *Let $G \subseteq \{-1,1\}^n$, and let $E \subseteq G$ such that $\Pr_{z \in \{-1,1\}^n}[z \in E] \geq 1 - \epsilon$.*
- *Let $\mathbf{T}$ be a distribution over functions $T : \{-1,1\}^n \to \{-1,1\}$ such that for every $z \in E$ it holds that $\Pr_{T \sim \mathbf{T}}[T(z) = -1] \geq 1 - \epsilon$, and for every $z \notin G$ it holds that $\Pr_{T \sim \mathbf{T}}[T(z) = 1] \geq 1 - \epsilon$.*
- *Let $\mathbf{z}$ be a distribution that is ϵ-pseudorandom for all but an ϵ-fraction of the tests in $\mathbf{T}$; that is, the probability over $T \sim \mathbf{T}$ that $\left|\Pr[T(\mathbf{u}_n) = -1] - \Pr[T(\mathbf{z}) = -1]\right| > \epsilon$ is at most ϵ.*

Then, the probability that $\mathbf{z} \in G$ is at least $1 - 6\epsilon$.

Loosely speaking, Lemma 3.5.39 implies the following: Assume that there exists a *distribution* $\mathbf{T}$ *over tests* $\{-1,1\}^{[n] \setminus I} \to \{-1,1\}$ such that for every *fixed input* z for which $\Phi\restriction_{\rho_{I,z}}$ is n^{-100}-close to σ it holds that $\mathbf{T}(z) = -1$, with high probability, and for every *fixed input* z for which $\Phi\restriction_{\rho_{I,z}}$ is not n^{-10}-close to σ it holds that $\mathbf{T}(z) = 1$, with high probability. That is, the distribution $\mathbf{T}$ constitutes a "randomized test" that distinguishes, with high probability, between "excellent" z's (such that $\Phi\restriction_{\rho_{I,z}}$ is very close to σ) and "bad" z's (such that $\Phi\restriction_{\rho_{I,z}}$ is relatively far from σ). Also assume that almost all tests $T : \{-1,1\}^{[n] \setminus I} \to \{-1,1\}$ in the support of $\mathbf{T}$ are "fooled" by a pseudorandom generator G. Then, with high probability over choice of seed for the pseudorandom generator G, the generator outputs z such that $\Phi\restriction_{\rho_{I,z}}$ is n^{-10}-close to σ. The main point when using the technique above is that the distribution $\mathbf{T}$, which may have very high entropy, is *only part of the analysis*; the actual algorithm that generates $\mathbf{z}$ is simply the pseudorandom generator G.

In our specific setting, the distribution $\mathbf{T}$ that we will use is equivalent to the following random process: Given $z \in \{-1,1\}^{[n] \setminus I}$, uniformly sample $\mathrm{poly}(n)$ points in the subcube corresponding to $\rho_{I,z}$, and accept z if Φ evaluates to the constant σ on all the sample points. We show how to construct such a distribution $\mathbf{T}$ such that *almost all* of the residual deterministic tests $T \in \mathtt{support}(\mathbf{T})$ are *conjunctions of $p(n) = \mathrm{poly}(n)$ LTFs*, and have very high acceptance probability (at least $1 - 1/\mathrm{poly}(p(n))$). Thus, any distribution that is $(1/\mathrm{poly}(n))$-pseudorandom for LTFs is also $(1/\mathrm{poly}(n))$-pseudorandom for almost all tests in the support of $\mathbf{T}$. Let us now turn to a formal proof of Lemma 3.5.38.

Proof of Lemma 3.5.38. Without loss of generality, assume that Φ is δ-close to the constant $\sigma = -1$. For any Boolean function f over a domain $\mathfrak{D}$, let $\mathtt{acc}(f) = \Pr_{x \sim \mathfrak{D}}[f(x) =$

$-1]$. Also, denote $J = [n] \setminus I$ and $n' = |J|$, and for any $z \in \{0,1\}^{n'}$, denote by ρ_z the restriction $\rho_z = (I, z)$ (i.e., we suppress I in the notation ρ_z, since I is fixed).

Let $G = \left\{ z \in \{0,1\}^{n'} : \mathtt{acc}(\Phi|_{\rho_z}) \geq 1 - \delta' \right\}$. Our goal is to show that $\Pr_{z \sim \mathbf{z}}[z \in G] \geq 1 - O(\gamma)$. Let $E = \left\{ z \in \{0,1\}^{n'} : \mathtt{acc}(\Phi|_{\rho_z}) \geq 1 - \sqrt{\delta} \right\}$. Note that when $z \in \{-1,1\}^{n'}$ is chosen uniformly it holds that $\mathbb{E}_{z \in \{-1,1\}^{n'}} \left[\mathtt{acc}(\Phi|_{\rho_z}) \right] = \Pr_{x \in \{-1,1\}^n}[\Phi(x) = -1] \geq 1 - \delta$. Therefore, $\Pr_{z \in \{-1,1\}^{n'}}[z \in E] \geq 1 - \sqrt{\delta}$.

We now construct a distribution $\mathbf{T}$ over tests $\{-1,1\}^{n'} \to \{-1,1\}$ that distinguishes, with high probability, between $z \in E$ and $z \notin G$. For $x \in \{0,1\}^{|I|}$, let T_x be the function that gets as input $z \in \{0,1\}^{n'}$, and outputs the value $\Phi(y)$, where $y_J = z$ and $y_I = x$. Note that for any fixed $z \in \{-1,1\}^{n'}$, when uniformly choosing $x \in \{-1,1\}^{|I|}$ it holds that $\Pr[T_x(z) = -1] = \mathtt{acc}(\Phi|_{\rho_z})$. Also, T_x is an LTF of its input z, because

$$T_x(z) = \mathrm{sgn}\left(\langle y, w \rangle - \theta\right) = \mathrm{sgn}\left(\langle z, w_J \rangle - (\theta - \langle x, w_I \rangle)\right) \ . \tag{3.5.8}$$

For $t = O\left(\frac{\log(1/\gamma)}{\delta'}\right)$ and $\bar{x} = (x^{(1)}, ..., x^{(t)}) \in \{0,1\}^{t \cdot |I|}$, let $T_{\bar{x}} : \{-1,1\}^{n'} \to \{-1,1\}$ be the function such that $T_{\bar{x}}(z) = -1$ if and only if for every $i \in [t]$ it holds that $T_{x^{(i)}}(z) = -1$ (i.e., $T_{\bar{x}}$ is the conjunction $\wedge_{i \in [t]} T_{x^{(i)}}$). Our distribution $\mathbf{T}$ is the uniform distribution over the set $\left\{ T_{\bar{x}} : \bar{x} \in \{0,1\}^{t \cdot |I|} \right\}$. Observe that:

- For any fixed $z \in E$ it holds that $\Pr_{T_{\bar{x}} \sim \mathbf{T}}\left[T_{\bar{x}}(z) = -1\right] \geq 1 - t \cdot \sqrt{\delta}$.
- For any fixed $z \notin G$ it holds that $\Pr_{T_{\bar{x}} \sim \mathbf{T}}\left[T_{\bar{x}}(z) = -1\right] \leq \gamma$.

We want to show that almost all of the tests $\{T_{\bar{x}}\}_{\bar{x} \in \{0,1\}^{t \cdot |I|}}$ in the support of $\mathbf{T}$ accept almost all of their inputs. To see that this is the case, observe that

$$\mathbb{E}_{\bar{x}}\left[\mathtt{acc}(T_{\bar{x}})\right] = \Pr_{\bar{x},z}[T_{\bar{x}}(z) = -1] \geq \Pr_{z}[z \in E] \cdot \min_{z \in E}\left\{ \Pr_{\bar{x}}[T_{\bar{x}}(z) = -1] \right\} \ ,$$

which is lower-bounded by $1 - \xi^2$, where $\xi^2 = (t+1) \cdot \sqrt{\delta}$. Therefore, the fraction of tests $T_{\bar{x}}$ that reject more than ξ of their inputs is at most ξ.

Now, let $T_{\bar{x}}$ be a test such that $\mathtt{acc}(T_{\bar{x}}) \geq 1 - \xi$. Since $T_{\bar{x}}$ is a conjunction of $T_{x^{(1)}}, ..., T_{x^{(t)}}$, for each $i \in [t]$ it holds that $\mathtt{acc}(T_{x^{(i)}}) \geq 1 - \xi$. Also, for each $i \in [t]$ it holds that $\mathbf{z}$ is η-pseudorandom for $T_{x^{(i)}}$, where $\eta \leq (\gamma^2 \cdot \delta')$, and therefore $\Pr_{z \sim \mathbf{z}}[T_{x^{(i)}}(z) = -1] \geq 1 - \xi - \eta$. It follows that $\Pr_{z \sim \mathbf{z}}[T_{\bar{x}}(z) = -1] \geq 1 - t \cdot (\xi + \eta)$.

We invoke Lemma 3.5.39 with the parameters $\epsilon_1 = \sqrt{\delta}$, $\epsilon_2 = t \cdot \sqrt{\delta}$, $\epsilon_3 = \gamma$, $\epsilon_4 = \xi$, and $\epsilon_5 = t \cdot (\xi + \eta)$, and deduce that

$$\begin{aligned}
\Pr_{z \sim \mathbf{z}}[z \notin G] &\leq (t+1) \cdot \sqrt{\delta} + \gamma + 2 \cdot \sqrt{t+1} \cdot \delta^{1/4} + t \cdot (\sqrt{t+1} \cdot \delta^{1/4} + \eta) \\
&= O\left(\gamma + t^{3/2} \cdot \delta^{1/4} + t \cdot \eta\right) \\
&= O\left(\gamma + (\gamma \cdot \delta')^{-3/2} \cdot \delta^{1/4} + \eta / (\gamma \cdot \delta')\right) \ ,
\end{aligned}$$

which is $O(\gamma)$ since $\eta \leq (\gamma^2 \cdot \delta')$ and by our hypotheses regarding γ, δ, and δ'. ∎

3.6 Polynomials that vanish rarely

3.6.1 Introduction and the main results

Let $\mathcal{P}_{n,q,d}$ denote the set of all polynomials $\mathbb{F}^n \to \mathbb{F}$ of total degree d over the field of size $q = |\mathbb{F}|$. We think of n as sufficiently large, and of the degree d and the field size q as functions of n. For simplicity, throughout the section we assume that $d < n$.[46]

A fundamental problem in complexity theory is that of constructing *hitting-set generators for low-degree polynomials*. Recall that a Hitting-Set Generator (HSG) for $\mathcal{P}_{n,q,d}$ is a function $H\colon \{0,1\}^\ell \to \mathbb{F}^n$ such that for every non-zero polynomial $p \in \mathcal{P}_{n,q,d}$ there exists $s \in \{0,1\}^\ell$ satisfying $p(H(s)) \neq 0$ (see Definition 2.4.9); in other words, every non-zero polynomial $p \in \mathcal{P}_{n,q,d}$ does not vanish on at least one element in the *hitting-set* $S = \{H(s)\colon s \in \{0,1\}^\ell\}$. The two main measures of efficiency for HSGs are the seed length ℓ (equivalently, the size of the hitting-set S as a multiset) and the computational complexity of H as a function (i.e., the computational complexity of generating an element of the hitting-set S given its index s).

A standard linear-algebraic argument yields a lower bound of $\Omega\left(d \cdot \log\left(n/d\right)\right)$ on the seed length of any HSG for $\mathcal{P}_{n,q,d}$, and a standard probabilistic argument shows that there *exists* a HSG for $\mathcal{P}_{n,q,d}$ with matching seed length $O\left(d \cdot \log(n/d) + \log\log(q)\right)$ (see Facts 2.4.12 and 2.4.13). Naturally, the probabilistic upper-bound does not guarantee that the function H is *efficiently-computable*. Thus, the main open problem concerning HSGs for $\mathcal{P}_{n,q,d}$ is to construct efficiently-computable HSGs with seed length that matches the known lower bound. This well-known problem (as well as a variant that refers to *pseudorandom generators* as in Definition 2.4.11) has attracted a significant amount of attention over the years; see, e.g., [NN93; LVW93; LV98; KS01; Bog05; BV10; BHS08; Lov09; Vio09b; Lu12; CTS13; ST18], and the related survey by Viola [Vio09a].

Several years ago, Goldreich and Wigderson [GW14, Section 5] considered a *relaxed version* of the foregoing problem. In general terms, what they asked is the following:

> Does the HSG problem become easier if we are guaranteed that the polynomial *vanishes rarely* (i.e., has very few roots)?

Note that, intuitively, we expect that the relaxed problem will indeed be easier: This is both since there are less polynomials that vanish rarely (than arbitrary polynomials), and since for any such polynomial p, almost all inputs will "hit" p.

In their original paper, Goldreich and Wigderson considered a specific instance of this problem, geared for a particular application (see Section 3.6.1.2 for details). Our goal here is to *study the relaxed problem in and of itself, in a systematic and general way*. Our motivation for doing so is three-fold. First, this is a special (and potentially-easy) case of the classical HSG problem, and thus constitutes a potential path to make progress on the classical problem. Secondly, the relaxed question is of independent interest as part of the broad study of *quantified derandomization*, which was initiated in the

[46] Most of our results also carry on to the setting of $d > n$, albeit with less "clean" parametrizations.

original work of Goldreich and Wigderson [GW14]. And thirdly, as polynomial-based constructions are ubiquitous in complexity theory, any progress in our understanding of structured classes of polynomials or in related HSG constructions may be valuable for other explicit constructions.

To be more formal, denote by $\mathcal{P}_{n,q,d,\epsilon}$ the set of polynomials $p \in \mathcal{P}_{n,q,d}$ such that $\Pr_{x \in \mathbb{F}^n}[p(x) = 0] \leq \epsilon$; that is, $\mathcal{P}_{n,q,d,\epsilon}$ is the set of degree-d polynomials that *vanish rarely*, where the notion of "rarely" is parametrized by the parameter ϵ. The two main questions we consider in this context are:

- **The combinatorial question:** What is the minimal size of a hitting-set for $\mathcal{P}_{n,q,d,\epsilon}$? Equivalently, we ask what is the minimal seed length of any HSG for $\mathcal{P}_{n,q,d,\epsilon}$. This question is combinatorial since it refers to the *existence* of a HSG, regardless of its computational complexity.
- **The computational question:** For which values of $\epsilon > 0$ can we construct a HSG for $\mathcal{P}_{n,q,d,\epsilon}$ with small seed length that will be *efficiently-computable*? In other words, can we simultaneously optimize not only the seed length but also the *computational complexity* of HSGs for $\mathcal{P}_{n,q,d,\epsilon}$?

3.6.1.1 Context and previous work

Let us first delineate some trivial values for ϵ. To do so, first recall that we expect a random polynomial to vanish on q^{-1} of its inputs. Now, by the Schwartz-Zippel lemma, any non-zero $p \in \mathcal{P}_{n,q,d}$ has at most an $\epsilon = d/q$ fraction of roots; this bound is quite good when q is large compared to d, and in general, for abitrary d and q, any non-zero polynomial vanishes on at most $1 - \delta$ of its inputs, where $\delta \geq q^{-d/(q-1)}$ denotes the relative distance of the Reed-Muller code of degree d over $\mathbb{F}_q$. Therefore, the value $\epsilon = 1 - \delta$ represents the general case (i.e., the case of hitting *any* non-zero polynomial). Remarkably, we also have a minimal non-zero value that ϵ can have: By a theorem of Warning [War35], every polynomial in $\mathbb{F}_q^n \to \mathbb{F}_q$ of degree d that vanishes *somewhere* vanishes on at least a q^{-d} fraction of its inputs. Therefore, hitting polynomials that vanish on $\epsilon < q^{-d}$ fraction of their inputs is trivial, since such polynomials have no zeroes. It will be useful to denote $\epsilon = q^{-t}$ from now on.

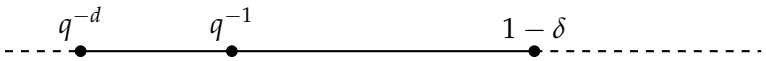

Figure 3.1: The two extremal values of ϵ (i.e., $\epsilon = q^{-d}$ and $\epsilon = 1 - \delta$) and the expected $\epsilon = q^{-1}$ for a random polynomial. (The parameter δ denotes the relative distance of the corresponding q-ary Reed-Muller code $RM(n, d)$.)

Referring to the combinatorial question, the standard probabilistic argument mentioned before shows there exists a HSG for $\mathcal{P}_{n,q,d,\epsilon}$ with seed length $O(\log\log(|\mathcal{P}_{n,q,d,\epsilon}|))$. Thus, the combinatorial question is intimately connected to the long-standing open problem of determining the *weight distribution of the Reed-Muller code*, i.e., *counting* the

number of polynomials in $\mathcal{P}_{n,q,d}$ that vanish on precisely $\epsilon > 0$ of their inputs, for every $\epsilon > 0$. The latter problem has been studied since the late 60's (see, e.g., [BS69; KT70]), but is currently settled only for $d = 2$ (see [SB70; McE69]). Only recently have general results been obtained for $d > 2$, and the bounds in these results are asymptotic (rather than precise bounds) and hold only over $\mathbb{F}_2$ (see [KLP12; ASW15]). More generally, this problem is a special case of the well-known problem of studying weight distributions of (classes of) linear codes, which is typically tackled using weight enumerator polynomials (for relevant background see, e.g., [MS77, Chapter 5]). Note, however, that the weight distribution problem is more general, since it refers to all non-trivial values of $\epsilon > 0$, whereas in our setting we focus only on tiny values of ϵ.

Another related line of works focuses on structural properties of *biased polynomials*. Fixing a polynomial $p : \mathbb{F}^n \to \mathbb{F}$ and looking at the distribution over $\mathbb{F}$ that is obtained by evaluating p at a random point, we can ask whether this distribution is close to uniform, or whether it is far from uniform, in which case we call the polynomial biased. A sequence of works showed that biased polynomials are very "structured", in the sense that they can be determined by a relatively-small number of polynomials of lower degree (see [GT09; KL08; HS10; Bha14; BHT15; BBG16]). Our setting is much more specific than the setting in these works, since their assumption is only that the polynomial is *biased*, whereas our assumption is that the polynomial is biased in a very specific manner (i.e., one output-value has tiny weight $\epsilon > 0$). Thus, the results in these works typically do not seem sufficiently strong to be useful in our more specific setting.[47]

Goldreich and Wigderson [GW14, Section 5], who were motivated by a specific application in circuit complexity (derandomization of $\mathcal{AC}^0[\oplus]$), constructed a polynomial-time computable HSG for the setting of $q = 2$ and $\epsilon = 2^{-(d-O(1))} = O(2^{-d})$ (for details see Section 3.6.1.2). Thus, they gave an upper-bound for the *computational question*, which holds only for $\mathbb{F}_2$ polynomials with extremely few roots. In a subsequent work [Tel19a], two combinatorial lower bounds were proved for the setting of $q = \text{poly}(n)$ and $\epsilon = q^{-O(1)}$ (again, for details see Section 3.6.1.2).[48] Thus, the subsequent work showed lower bounds for the *combinatorial question*, which hold only for polynomials over $\mathbb{F}_{\text{poly}(n)}$ with a relatively-large number of roots (i.e., only mildly less roots than the expected value of $\epsilon = q^{-1}$). In both these works, ad-hoc arguments were used to obtain the corresponding results.

3.6.1.2 Our main results

Our first main result is a general lower bound for the combinatorial problem. For context, in [Tel19a] it was shown that when $q = \text{poly}(n)$, any HSG for $\mathcal{P}_{n,d,q,q^{-O(1)}}$

[47] One exception is the field $\mathbb{F}_2$, in which the notions of bias and of "vanish rarely" converge. Indeed, the proofs of our results for $\mathbb{F}_2$ use insights developed in this sequence of works.

[48] We do not include proofs of these results from [Tel19a] in the current thesis. This is done both for simplicity, and since these results are almost completely superseded by the results included here.

requires a seed of length $\Omega(d^{\Omega(1)} \cdot \log(n/d^{\Omega(1)}))$; and any HSG with constant density[49] for $\mathcal{P}_{n,d,q,q^{-1}}$ requires a seed of length $\Omega(d \cdot \log(n/d))$. Thus, both previous lower bounds referred to the setting of $q = \text{poly}(n)$ and of $\epsilon = q^{-O(1)}$ (i.e., $t = O(1)$).

The following result shows a lower bound that is both significantly stronger, and – more importantly – applies to a far broader parameter setting. In particular, the following result applies to a general $q \leq \text{poly}(n)$ and to values of $\epsilon = q^{-t}$ almost up to the extreme value of $\epsilon = q^{-d}$, and gives a lower bound of $\Omega((d/t) \cdot \log(n))$:

Theorem 3.6.1 (lower bound over general fields)**.** *For every constant $c > 1$ there exists a constant $\gamma > 0$ such that the following holds. For every $n, q, d, t \in \mathbb{N}$ such that $2 \leq q \leq n^c$ is a prime power, $d \leq n^{.49}$, and $t \leq \gamma \cdot d$, any HSG for $\mathcal{P}_{n,q,d,q^{-t}}$ requires a seed of length $\Omega\left((d/t) \cdot \log(n)\right)$.*

Let us parse the meaning of the lower bound in Theorem 3.6.1. For comparison, recall that there exists a HSG for all polynomials of degree $d \leq n^{.49}$ with seed length $O(d \cdot \log(n))$. Theorem 3.6.1 tells us that the relaxation of only requiring to "hit" polynomials that vanish with probability q^{-t} can "buy" a factor of at most $1/t$ in the seed length. In particular, there does not exist a significantly smaller hitting-set for polynomials that vanish with probability $q^{-O(1)}$. Perhaps surprisingly, this is also true for polynomials that vanish with probability $q^{-d^{o(1)}}$ (since the lower bound remains almost linear in $d \cdot \log(n)$). Only for polynomials that vanish with probability $q^{-d^{\Omega(1)}}$ does our lower bound imply that a significantly smaller hitting-set *might* exist; and at an "extreme" value of $q^{-\Omega(d)}$, our lower bound does not rule out a polynomial-sized hitting-set.

For technical statements that include various extensions and improvements of Theorem 3.6.1 (and in particular also hold for polynomials of higher degree $n^{.49} < d \leq \gamma \cdot n$), see the beginning of Section 3.6.4, and specifically Theorems 3.6.14, 3.6.19, and 3.6.20.[50]

Now, still referring to the combinatorial question, we observe that a result of Kaufmann, Lovett, and Porat [KLP12], which upper-bounds the *number* of biased $\mathbb{F}_2$ polynomials (i.e., analyzes the weight distribution of the Reed-Muller code over $\mathbb{F}_2$), yields a corresponding existential upper-bound. Specifically:

Theorem 3.6.2 (upper-bound over $\mathbb{F}_2$, following [KLP12])**.** *Let $n, d, t \in \mathbb{N}$ where $d > t$. Then, there exists a (non-explicit) hitting-set for $\mathcal{P}_{n,2,d,2^{-t}}$ with seed length $O\left((d-t) \cdot \log(\frac{n}{d-t})\right)$.*

Note that while the lower bound in Theorem 3.6.1 holds for any finite field, the upper bound in Theorem 3.6.2 holds only over $\mathbb{F}_2$. Nevertheless, comparing Theorems 3.6.1 and 3.6.2 (for $\mathbb{F} = \mathbb{F}_2$ and $d \leq n^{.49}$) reveals that there is still a *significant gap* between the upper-bound and the lower-bound: The lower bound is of the form $(d/t) \cdot \log(n)$, whereas the existential upper bound is of the form $(d-t) \cdot \log(n)$.

[49] A hitting-set S for a class $\mathcal{P}$ has density $\epsilon > 0$ if for every $p \in \mathcal{P}$ it holds that $\Pr_{s \in S}[p(s) \neq 0] \geq \epsilon$.

[50] In these technical results, the $\log(n)$ term in the lower bound in Theorem 3.6.1 is replaced by a more complicated term that depends on d and on t, for example $\log(n^{.99} \cdot (t/d))$.

For example, the lower bound indicates that there *might* exist a significantly smaller hitting-set for the relaxed problem when $t = d^{\Omega(1)}$, whereas the existential upper bound is significantly better than the one for the original problem only for $t = d - d^{\Omega(1)}$.

Our last main result is computational and shows an *explicit* construction of a HSG. As mentioned above, Goldreich and Wigderson [GW14] constructed a polynomial-time computable HSG with seed length $O(\log(n))$ that "hits" polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ of degree d that vanish on $O(2^{-d})$ of their inputs (for any $d \in \mathbb{N}$). We prove a significantly more general result, by constructing an explicit HSG for $\mathcal{P}_{n,2,d,2^{-t}}$ for any $t < d - O(1)$:

Theorem 3.6.3 (explicit upper-bound over $\mathbb{F}_2$)**.** *Let $n \in \mathbb{N}$ be sufficiently large, and let $d > t + 4$ be integers. Then, there exists a polynomial-time computable HSG for $\mathcal{P}_{n,2,d,2^{-t}}$ with seed length $O\left((d-t) \cdot \left(2^{d-t} + \log(\frac{n}{d-t})\right)\right)$.*

Note that the original result from [GW14] is the special case of Theorem 3.6.3 when $t = d - O(1)$. Also note that the seed length of the explicit HSG from Theorem 3.6.3 depends exponentially on $d - t$, whereas the seed length of the non-explicit HSG from Theorem 3.6.2 depends linearly on $d - t$. We also comment that the result is actually slightly stronger, and asserts that for any $r \in \mathbb{N}$ there exists a polynomial-time computable HSG for $\bigcup_d \mathcal{P}_{n,2,d,q^{d-r}}$ with seed length $O(r \cdot (2^r + \log(n/r)))$; that is, for every r there is a *single* HSG that works for *all degrees* d with $t = d - r$.

Below, in Table 3.2, we present an informal summary of the main results mentioned above, and compare them to previously-known results.

	Seed length		**Field Size**	ϵ
Lower bounds				
[Tel19a]	$\Omega(d^{\Omega(1)} \cdot \log(n/d^{\Omega(1)}))$		$q = \text{poly}(n)$	$q^{-O(1)}$
Thm 3.6.1	$\Omega((d/t) \cdot \log n)$	$(d \leq n^{.49})$	$2 \leq q \leq \text{poly}(n)$	q^{-t}
Thm 3.6.14	$\Omega((d/t) \cdot \log(n^{.99} \cdot t/d))$	$(d/t \lessapprox q \cdot n^{.01})$	$2 \leq q \leq \text{poly}(n)$	q^{-t}
Upper bounds				
[GW14]	$O(\log n)$	(explicit)	$q = 2$	$2^{-d+O(1)}$
Thm 3.6.2	$O((d-t)\log(\frac{n}{d-t})$	(non-explicit)	$q = 2$	2^{-t}
Thm 3.6.3	$O((d-t) \cdot (2^{d-t} + \log(\frac{n}{d-t}))$	(explicit)	$q = 2$	2^{-t}

Table 3.2: An informal summary of our results and comparison to previous results.

3.6.1.3 The connection to small sets with large degree-d closures

In addition to our lower-bounds and upper-bounds for the problem of HSGs for polynomials that vanish rarely, we also tie this problem to the study of a clean and elegant algebraic question; namely, to the study of the degree-d closure of a set $S \subseteq \mathbb{F}^n$, which was recently initiated by Nie and Wang [NW15].

Using terminology from algebraic geometry, the degree-d closure of a set $S \subseteq \mathbb{F}^n$ is a finite-degree analogue of the Zariski closure of S, and is defined as the variety induced by the set of degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$ that vanish on S. In more detail, let us first define the degree-d ideal of S to be $\mathcal{I}^{(d)}(S) = \{p \in \mathcal{P}_d : \forall s \in S, p(s) = 0\}$, where $\mathcal{P}_d$ is the set of degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$.[51] Then, the degree-$d$ closure of S is defined by:

$$\mathtt{Cl}^{(\mathtt{d})}(S) = \{x \in \mathbb{F}^n : \forall p \in \mathcal{I}^{(d)}(S), p(x) = 0\} .$$

As an example, observe that the degree-d closure of any $d+1$ points on a fixed line in $\mathbb{F}^n$ contains the entire line. As another example, recall that the closure of any Kakeya set in $\mathbb{F}_q^n$ with respect to homogeneous degree-$(q-1)$ polynomials is the entire domain $\mathbb{F}_q^n$ (this was proved by Dvir [Dvi09, Section 3] towards showing that any Kakeya set is necessarily of size at least $\binom{q+n-1}{n}$).

Following the latter example, it is natural to ask whether there exists a *very small* set $S \subseteq \mathbb{F}^n$ whose degree-d closure is *very large*. An initial observation towards answering this question is that a set $S \subseteq \mathbb{F}^n$ has *maximal* degree-d closure (i.e., $\mathtt{Cl}^{(\mathtt{d})}(S) = \mathbb{F}^n$) if and only if S is a hitting-set for degree-d polynomials. (This is since in both cases, the only degree-d polynomial that vanishes on S is the zero polynomial.)

Observation 3.6.4 (maximal closure $\iff$ hitting-set)**.** *A set $S \subseteq \mathbb{F}^n$ is a hitting-set for (all) degree-d polynomials* if and only if $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| = q^n$.

Loosely speaking, the main result of Nie and Wang [NW15] extends Observation 3.6.4 by showing that that for any $S \subseteq \mathbb{F}^n$ it holds that $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| \leq \frac{|S|}{\binom{n+d}{d}} \cdot |\mathbb{F}|^n$. The meaning of this result is that, while there exist sets of size $|S| = \binom{n+d}{d}$ whose degree-d closure is $\mathbb{F}^n$, the degree-d closure of smaller sets decreases by a factor of at least $\frac{|S|}{\binom{n+d}{d}}$.[52]

We take another approach to extending Observation 3.6.4, by by establishing a connection between the study of small sets with large closures and the study of HSGs for polynomials that vanish rarely. Specifically, we show two-way implications between the statement that S is a hitting-set generator for polynomials *that vanish rarely*, and the

[51] Note that $\mathcal{I}^{(d)}(S)$ is not an actual ideal in the ring of n-variate polynomials over $\mathbb{F}$, since multiplying $p \in \mathcal{I}^{(d)}(S)$ by another polynomial does not necessarily preserve the degree of p.

[52] Another result along these lines was recently proved by Beelen and Datta [BD18], who showed a tight upper-bound on the size of the variety induced by *any* subspace of degree-d polynomials (rather than only for varieties induced by a subspace of the form $\mathcal{I}^{(d)}(S)$ for some $S \subseteq \mathbb{F}^n$).

statement that S has *large closure*. In more detail, we relate hitting-sets for polynomials that vanish with probability q^{-t} to sets with closure of size q^{n-t}:

Theorem 3.6.5 (small sets with large closures versus hitting-sets for polynomials that vanish rarely)**.** *Let $\mathbb{F}$ be a field of size q, let $n \in \mathbb{N}$ and $t < d < n$, and let $S \subseteq \mathbb{F}^n$. Then,*

1. *If $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| > q^{n-t}$, then S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$.*
2. *If S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$, then $\left|\mathtt{Cl}^{(\mathtt{d}/2(\mathtt{t}+1))}(S)\right| > \frac{1}{2} \cdot q^{n-t}$.*

Notice that Theorem 3.6.5 does not show a complete equivalence between the two notions, since in the second item the closure refers to degree $d/2t$ rather than to degree d. Thus, intuitively, Theorem 3.6.5 asserts that constructing a small set with a large degree-d closure is at least as hard as constructing a hitting-set for polynomials that vanish rarely; and while it also gives a converse reduction (in the second item), it is nevertheless possible that constructing a hitting-set for polynomials that vanish rarely is an easier problem. We also remark that the first item in Theorem 3.6.5 is almost immediate, whereas the second item requires more work (see Section 3.6.5 for details).

Lastly, we comment that one can obtain an upper-bound on the size of $\mathtt{Cl}^{(\mathtt{d})}(S)$ for small sets $S \subseteq \mathbb{F}^n$ by combining the first item in Theorem 3.6.5 with our lower bound from Theorem 3.6.1. (This is since the former asserts that sets with closure of size q^{n-t} are hitting-sets for $\mathcal{P}_{n,q,d,q^{-t}}$, whereas the latter asserts that any such hitting-set must be large.) However, the bounds obtained in this way are not stronger than the known bounds proved in [NW15]. For more details see Section 3.6.5.

3.6.2 Proof overviews

3.6.2.1 Combinatorial lower bounds from low-degree dispersers

The proofs of our lower bounds on HSGs for polynomials that vanish rarely rely on a *complexity-theoretic* approach, rather than on a direct algebraic analysis. Specifically, we reduce the problem of constructing HSGs for *arbitrary* polynomials to the problem of constructing HSGs for polynomials that *vanish rarely*; since we already know lower bounds for the former, we obtain lower bounds for the latter.

Specifically, given an arbitrary non-zero polynomial $p_0 \colon \mathbb{F}^m \to \mathbb{F}$, we will use a form of "error-reduction" for polynomials (akin to error-reduction for probabilistic algorithms; see below) to obtain another polynomial $p \colon \mathbb{F}^n \to \mathbb{F}$ such that:

1. The polynomial p vanishes rarely.
2. Any non-zero input for p can be mapped into a small list of inputs for p_0 that contains a non-zero input for p_0.

To define p, fix a (k,δ)-disperser $\mathsf{Disp} \colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$, for appropriate parameters k and δ that we will determine in a moment.[53] Then, p is the result of the following

[53] A (k,δ)-disperser $\mathsf{Disp} \colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ is a function such that for every $T \subseteq \mathbb{F}^m$ satisfying $|T|/|F|^m \geq \delta$, for all but at most 2^k of the inputs $z \in \mathbb{F}^n$ there exists $i \in \{0,1\}^\ell$ such that $\mathsf{Disp}(z,i) \in T$.

procedure: Given $z \in \mathbb{F}^n$, compute the 2^ℓ inputs $\{\mathsf{Disp}(z,i)\}_{i\in\{0,1\}^\ell}$, evaluate p_0 at each of these inputs, and output the disjunction of these evaluations; that is:

$$p(z) = \bigvee_{i\in\{0,1\}^\ell} p_0\left(\mathsf{Disp}(z,i)\right) \ .$$

The disperser Disp has the property that for every set $T \subseteq \mathbb{F}^m$ of density at least δ it holds that $\Pr_{z\in\mathbb{F}^n}[\forall i\ \mathsf{Disp}(z,i) \notin T] \leq \epsilon = 2^k/q^n$. We take T to be the set of elements in $\mathbb{F}^n$ on which p_0 does not vanish, and take δ to be the density of T (i.e., δ is the distance of the corresponding Reed-Muller code); we also let $k = (n-t)\cdot\log(q)$. Then, the polynomial p vanishes on at most an $\epsilon = 2^k/q^n = q^{-t}$ fraction of its inputs. Also, any non-zero input $z \in \mathbb{F}^n$ for p can be mapped to a list of 2^ℓ inputs $\{x_i = \mathsf{Disp}(z,i)\}_{i\in\{0,1\}^\ell}$ for p_0 such that for some $i \in \{0,1\}^\ell$ it holds that $p_0(x_i) \neq 0$, as we wanted.

The reduction above shows that if there exists a HSG with seed length s for polynomials $\mathbb{F}^n \to \mathbb{F}$ of degree $d = \deg(p)$ that vanish with probability ϵ, then there exists a corresponding HSG with seed length $s+\ell$ for all non-zero polynomials $\mathbb{F}^m \to \mathbb{F}$ of degree $d_0 = \deg(p_0)$. The known lower bound on the latter, which asserts that $s+\ell = \Omega(d_0\cdot\log(m/d_0))$, yields a corresponding lower bound on the former.

While this is indeed our main idea, it unfortunately does not quite work as-is. The main challenge is that the reduction above incurs *significant overheads* that crucially deteriorate the lower bound. Most importantly, the *degree* of the polynomial increases (from $d_0 = \deg(p_0)$ to $d = \deg(p)$), and the number of variables also increases (from m to n); this affects us since we are interested in a lower bound as a function of n and d, whereas our lower bound is a function of m and d_0. Moreover, the lower bound deteriorates by an additive factor of ℓ, since each non-zero input $z \in \mathbb{F}^n$ for p yields 2^ℓ inputs for p_0, one of which is guaranteed to be non-zero. Thus, we want to modify the reduction above, in order to minimize the blowup in the degree and in the number of variables, and also minimize the seed length ℓ of the disperser.

A coding-theoretic perspective. One can view the procedure described above as amplifying the *weight* (i.e., the fraction of non-zero coordinates) of a codeword in the Reed-Muller code. At first glance, this task seems similar to the task of amplifying the *distance* of linear error-correcting codes; in particular, the disperser-based technique described above is technically reminiscent of the well-known distance amplification technique of Alon *et al.* [ABN+92].[54] However, the crucial difference is that we are interested in amplifying the weight to be much larger than $1-1/q$, and indeed our resulting subcode (of polynomials that vanish rarely) is a small and non-linear subcode of the Reed-Muller code. Moreover, as explained above, we will be particularly interested in the degree blow-up, which is a parameter specific to polynomial-based codes.

[54] The main differences are that we will use a specific disperser that is different from theirs, to minimize the degree blow-up; and that we handle alphabet reduction differently (using an OR function instead of code concatenation), since our target weight is much larger than $1-1/q$.

Warm-up: The setting of $d \ll q$. For simplicity, let us assume that $q = \text{poly}(n)$ and that $d \leq n^{.99}$. In this case the fraction δ of non-zeroes of p_0 is very close to one and we only need Disp to be a $(k, .99)$-disperser for $k = (n-t) \cdot \log(q)$.

Note that to compute p at an input $z \in \mathbb{F}^n$, we wish to compute $\text{Disp}_i(z) = \text{Disp}(z, i)$ as a function of z for each *fixed* value i of the seed. Since we want p to have degree as low as possible, we are interested in objects that we call low-degree dispersers: Informally, a disperser $\text{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ has low degree if for any $i \in \{0,1\}^\ell$ and $j \in [m]$, the polynomial $q_{i,j}(z) = \text{Disp}(z, i)_j$ (i.e., $q_{i,j}(z)$ is the j^{th} output element of $\text{Disp}(z, i)$ as a function of z) has low degree (see Definitions 2.5.1 and 2.5.3). Note that in our argument we only need the *existence* of a low-degree disperser (i.e., we do not need the low-degree disperser to be efficiently computable); however, the dispersers that are obtained via naive probabilistic arguments do not have low degree.

Fortunately, in the current "warm-up" setting we can get a good (albeit non-optimal) lower bound even using the "naive disperser" that just performs uniform sampling: That is, the disperser that treats its input $z \in \mathbb{F}^n$ as n/m substrings of length m, and treats its seed as an index $i \in [n/m]$, and outputs the i^{th} substring of length m in z. Note that this disperser is *linear* (i.e., has degree one), since for a fixed seed, each output element is a projection of a corresponding input element.

We do encounter one other problem in implementing our idea in this setting, which is the degree blow-up that comes from the fact that p computes the OR function on the outputs of the disperser (recall that the OR function of 2^ℓ inputs has maximal degree $(q-1) \cdot 2^\ell$). To circumvent this problem, we replace the OR function with a multivalued OR function. Specifically, observe that in the reduction above it suffices that on any non-zero input $y \in \mathbb{F}^{2^\ell}$, the OR function will output *some* non-zero element (rather than map any non-zero y to $1 \in \mathbb{F}$). In contrast to the OR function, there exists a multivalued OR function of 2^ℓ elements with degree roughly 2^ℓ (see Proposition 2.3.8).

Working out the precise parameters, this approach transforms any p_0 of degree d_0 into a corresponding p of degree $d = d_0 \cdot 2^\ell = d_0 \cdot t \cdot \log(q)$, and for every $t \leq d/O(\log(q))$ implies a lower bound of $\Omega(d_0 \cdot \log(m/d_0)) - \ell = \Omega(d/t)$ on the seed length of HSGs for polynomials that vanish with probability q^{-t}. To improve this lower bound to match the bound stated in Theorem 3.6.1, we use a disperser that is better than the naive one, and utilize the techniques that are outlined below (see Section 3.6.4).

The more challenging setting of $d \gg q$. Observe that in the argument above we "paid" for the seed length ℓ of the disperser *twice*: One loss was a blow-up of 2^ℓ in the degree (since the multivalued OR function has degree 2^ℓ), and the other loss was that the lower bound on the seed length of the HSG decayed additively in ℓ (because our reduction maps any non-zero input for p to a list of 2^ℓ inputs for p_0). Also note that the first loss decreases the lower bound itself, whereas the second loss limits the values of t to which the lower bound applies (to ones for which $\ell \ll d_0 \cdot \log(m/d_0)$).

When $d \gg q$ these two losses may deteriorate our lower bound much more severely than in the "warm-up" setting. This is because when q was large we instantiated the

disperser with the parameter $\delta = \Omega(1)$, and hence its seed length was relatively small, whereas in our current setting the value of $\delta = q^{-d_0/(q-1)}$ may be much smaller.[55]

Over prime fields this problem can be overcome by starting not from a lower bound for hitting all degree-d_0 polynomials, but rather from a lower bound for hitting a large subcode of the corresponding Reed-Muller code (i.e., a subcode with dimension linear in $\binom{m+d_0}{d_0}$) that still has distance $\Omega(1)$; see Appendix 3.6.6.2 for an explanation. To overcome the problem also over non-prime fields, we show a general method that, regardless of the disperser, *allows us to "pay" only an $O(t)$ factor in the degree blow-up*, instead of the 2^ℓ factor. This method does not prevent the additive loss of ℓ in the seed length, and we will explain how this additive loss affects us in the end of the current section.

To explain this method, fix a disperser, and recall that our goal is to "hit" the set $G \subseteq \mathbb{F}^n$ of inputs z such that for some $i \in \{0,1\}^\ell$ it holds that $p_0(\mathsf{Disp}(z,i)) \neq 0$ (since any $z \in G$ maps to 2^ℓ inputs, one of which "hits" the original polynomial p_0). We think of the polynomial p above as a test of its input $z \in \mathbb{F}^n$ that distinguishes between G and $\mathbb{F}^n \setminus G$ (i.e., p vanishes precisely on $\mathbb{F}^n \setminus G$). Our initial approach to hit G was to construct a HSG for the test p, which would output some $z \in G$.

The key observation is that constructing a HSG for p is an "overkill". Specifically, to hit G, *we can replace the test p by a distribution $\mathbf{p}$ over tests that distinguishes between G and $\mathbb{F}^n \setminus G$, with high probability*, and still deduce that any HSG for the tests in the support of $\mathbf{p}$ outputs some $z \in G$. That is, we replace the test p for G by a randomized test $\mathbf{p}$ for G such that the polynomials in the support of $\mathbf{p}$ have lower degree than p, and show that "hitting" the polynomials in the support of $\mathbf{p}$ still allows us to "hit" G. Moreover, since $\mathbf{p}$ "tests" a *dense* set G with small error, by an averaging argument almost all of the polynomials in the support of $\mathbf{p}$ *vanish rarely*; thus, it suffices to "hit" only the polynomials in the support of $\mathbf{p}$ that vanish rarely.

More accurately, let us instantiate our disperser with $k = (n-2t)\cdot\log(q)$, instead of $k = (n-t)\cdot\log(q)$, such that the density of G is $1 - q^{-2t}$ (this is to allow for some slackness in the parameters). Then, the following holds:

Lemma 3.6.6 (informal; see Section 3.2)**.** *Assume there exists a distribution $\mathbf{p}$ over polynomials $\mathbb{F}^n \to \mathbb{F}$ such that for every $z \in G$ it holds that $\Pr[\mathbf{p}(z) \neq 0] \geq 1 - q^{-2t}$ and for every $z \notin G$ it holds that $\Pr[\mathbf{p}(z) = 0] = 1$. Further assume that every polynomial in the support of $\mathbf{p}$ has degree $O(d \cdot t)$. Then, any hitting-set for polynomials of degree $O(d \cdot t)$ that vanish on at most $2q^{-t}$ of their inputs contains some $z \in G$.*

Our construction of the specific distribution $\mathbf{p}$ that we use is simple: Starting from the construction of p above, instead of taking an OR of the evaluations of p_0 on the entire output-set of the disperser (i.e., on all seeds), we *sample from the seeds of the disperser*. More accurately, to sample a polynomial $f \sim \mathbf{p}$, we uniformly sample $2t$

[55] To demonstrate the problem, note that over fields of constant size, even a disperser with optimal parameters would yield a quadratic degree blow-up, regardless of t; that is, $d \geq 2^\ell \cdot d_0 \geq 2^{\log(t\cdot\log(q)/\delta)} \cdot d_0 = \Omega_q((d_0)^2 \cdot t)$, compared to the previous blow-up of $d = \Omega_q(d_0 \cdot t)$ when we had $\delta = \Omega(1)$.

vectors $a^{(1)}, ..., a^{(2t)} \in \mathbb{F}^{2^\ell}$, and output the polynomial

$$f(z) = \mathtt{OR}_{j\in[2t]}\left(\sum_{i\in 2^\ell} a_i^{(j)} \cdot p_0(\mathsf{Disp}(z,i))\right) .$$

To see why this distribution works, observe that if $z \in G$ then a random $\mathbb{F}$-linear sum of the elements $\{\mathsf{Disp}(x,i)\}_{i\in\{0,1\}^\ell}$ will be non-zero with probability $1 - 1/q$, whereas if $z \notin G$ then such a sum will be zero, with probability one. Thus, a random polynomial in $\mathbf{p}$ computes the disjunction of $2t$ such random sums, and it is straightforward to see that its "error probability" is q^{-2t} and its degree is $O(d_0 \cdot t)$ (assuming that the disperser is linear). Using Lemma 3.6.6, any HSG for polynomials of degree $O(d_0 \cdot t)$ that vanish on at most q^{-2t} of their inputs outputs some $z \in G$. We therefore reduced the problem of constructing a HSG for p_0 to the problem of constructing a HSG for polynomials of degree $d = O(d_0 \cdot t)$ that vanish on at most q^{-2t} of their inputs.

The last missing piece is that we need a concrete disperser to instantiate the argument with, and the parameters of the disperser will determine the lower bound that we get. Furthermore, recall that we are losing an additive factor of ℓ in the lower bound, and thus any lower bound that we get using this approach applies only to values of t such that $\ell \ll d_0 \cdot \log(m/d_0)$. Specifically, the approach above gives the following lemma (for simplicity, we state it only for linear dispersers):

Lemma 3.6.7 (linear dispersers yield lower bounds on HSGs for polynomials that vanish rarely; informal, see Corollary 3.6.12)**.** *Let $d_0 < m$ be integers, let $\mathbb{F}$ be a field of size q, and let $t \in \mathbb{N}$. Assume that for $k = (n-2t)\cdot\log(q)$ and $\delta = q^{-d_0/(q-1)}$ there exists a linear (k,δ)-disperser $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$. Then, for $d = 4d_0 \cdot t$, if $\ell \le \frac{d}{8t}\cdot\log(mt/d)$, then the seed length for any HSG for $\mathcal{P}_{n,q,d,2q^{-t}}$ is $\Omega\left((d/t)\cdot\log\left(mt/d\right)\right)$.*

Note that to get a good lower bound using Lemma 3.6.7 we want a *linear* disperser $\mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ for large min-entropy $k = (n-2t)\cdot\log(q)$ that has small seed length ℓ and large output length m.[56] In particular, if there exists a *linear* disperser with *optimal* parameters, then a lower bound of $\Omega((d/t)\cdot\log(nt/d))$ would follow for essentially all settings of the parameters (see Corollary 3.6.13).

Our lower bounds (i.e., Theorem 3.6.1 and its extensions) will be proved by instantiating Lemma 3.6.7 with specific useful dispersers. To prove Theorem 3.6.1 and some of its extensions (i.e., Theorems 3.6.14 and 3.6.19), we use a linear disperser that we obtain by modifying the extractor by Shaltiel and Umans [SU05]; the original extractor works over the binary alphabet, and we modify it to a linear disperser over an arbitrary field $\mathbb{F}_q$ (see Section 3.6.4 for details). We prove another lower bound, which applies only to fields of constant size (see Theorem 3.6.20), using a linear disperser that is based on the recent construction of "linear 1-local expanders" by Goldreich [Gol16], following Viola and Wigderson [VW17] (see Section 3.6.4.3). More details are given in Section 3.6.4.

[56] Moreover, since our error $\delta = q^{-d_0/(q-1)}$ might be large, we want good dependency of the parameters ℓ and m on the error δ.

3.6.2.2 Explicit upper bound over $\mathbb{F}_2$

To construct the explicit HSG for polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish rarely in Theorem 3.6.3 we generalize a construction of [GW14]. In high-level, we reduce the problem of constructing a HSG for polynomials that vanish rarely to the problem of constructing a PRG for arbitrary *low-degree polynomials*, and then use the explicit PRG of Viola [Vio09b] for low-degree polynomials.

In more detail, we say that a polynomial $p : \mathbb{F}_2^n \to \mathbb{F}_2$ is approximated by a distribution $\mathbf{h}$ over polynomials $h \colon \mathbb{F}_2^n \to \mathbb{F}_2$ if for every $x \in \mathbb{F}_2^n$ it holds that $\Pr_h[\mathbf{h}(x) = p(x)] \geq .99$. Our first step is to show that any polynomial $p \in \mathcal{P}_{n,2,d,q^{-t}}$ can be approximated by a distribution $\mathbf{h}$ over polynomials of degree $d - t$. To do so, let $\Delta_a(p)$ be the directional derivative of p in direction $a \in \mathbb{F}_2^n$ (i.e., the function $\Delta_a p(x) = p(x + a) + p(x)$). We sample $h \sim \mathbf{h}$ by uniformly sampling $\vec{a} = a^{(1)}, ..., a^{(k)} \in \mathbb{F}_2^n$, where $k = t - O(1)$, and outputting the polynomial $h_{\vec{a}} = \Delta_{a^{(k)}} \Delta_{a^{(k-1)}} ... \Delta_{a^{(1)}}(p) + 1$; that is, we derive p in k random directions, and "negate" the output.

Note that indeed $\deg(h_{\vec{a}}) = d - t + O(1)$. Now, for any fixed $x \in \mathbb{F}_2^n$ and non-empty $S \subseteq [k]$, the probability over $\vec{a}$ that $p\left(x + \sum_{i \in S} a^{(i)}\right) = 1$ is at least $1 - 2^{-t}$ (since p vanishes with probability at most 2^{-t}, and $x + \sum_{i \in S} a^{(i)}$ is uniform in $\mathbb{F}_2^n$). Thus, by a union bound, with probability at least .99 over the choice of $\vec{a}$, for every non-empty $S \subseteq [k]$ it holds that $p\left(x + \sum_{i \in S} a^{(i)}\right) = 1$. In this case, we have that $h_{\vec{a}}(x) = \sum_{S \subseteq [k]} p\left(x + \sum_{i \in S} a^{(i)}\right) + 1 = p(x) + (2^k - 1) + 1 = p(x)$. Hence, the distribution $\mathbf{h}$ also has the property that for every $x \in \mathbb{F}_2^n$ it holds that $\Pr[\mathbf{h}(x) = p(x)] \geq .99$.

Our next observation is similar to the "randomized tests" technique mentioned in Section 3.6.2.1: We show that if a distribution $\mathbf{h}$ over low-degree polynomials approximates p, then a pseudorandom generator for the polynomials in the support of $\mathbf{h}$ (with sufficiently small constant error) also "hits" p (for a proof see Section 3.2). Combining the two claims, we get a reduction from the problem of constructing a HSG for $\mathcal{P}_{n,2,d,q^{-t}}$ to the problem of constructing a PRG (with small constant error) for arbitrary polynomials of degree $d - t + O(1)$. Thus, the PRG of Viola [Vio09b] for such polynomials, which uses a seed of length $O((d - t) \cdot (2^{d-t} + \log(n)))$, is also a HSG for $\mathcal{P}_{n,2,d,2^{-t}}$.

On the tightness of the reduction above. Recall that there is a gap between the seed length of the explicit HSG above and the seed length of the *non-explicit* HSG from Theorem 3.6.2, which is $O\left((d - t) \cdot \log(\frac{n}{d-t})\right)$. We note that to close this gap, one does not need to improve the *reduction* detailed above, but only the *explicit PRG for arbitrary polynomials* (i.e., Viola's construction). Specifically, if there exists an explicit PRG for all polynomials of degree $d' = d - t + O(1)$ with seed length $O(d' \cdot \log(n/d'))$ (matching the non-explicit upper-bound for such PRGs), then the reduction above yields a HSG for $\mathcal{P}_{n,2,d,2^{-t}}$ with seed length $O((d - t) \cdot \log(n/(d - t)))$.

3.6.3 Upper bounds over $\mathbb{F}_2$

In this section we prove Theorems 3.6.2 and 3.6.3; that is, we construct explicit and non-explicit hitting-set generators for polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish rarely.

We define the weight of a polynomial $p\colon \mathbb{F}^n \to \mathbb{F}$ to be $wt(p) = \Pr_{x\in\mathbb{F}^n}[p(x) \neq 0]$. Indeed, we are interested in polynomials with very high weight. Kaufman, Lovett, and Porat [KLP12] proved a near-tight upper-bound on the number of polynomials with very low weight when $\mathbb{F} = \mathbb{F}_2$; as a consequence, we get the following non-explicit hitting-set generator on polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish rarely:

Theorem 3.6.8 (non-explicit HSGs for $\mathbb{F}_2$ polynomials that vanish rarely, following [KLP12])**.** *Let $n, d, t \in \mathbb{N}$ where $t < d \leq n$. Then, the number of polynomials in $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish with probability at most 2^{t-d} is at most $2^{O(d^2\cdot t/(d-t+1)!\cdot n^{d-t+1})}$. In particular, there exists a hitting-set generator for this set of polynomials with seed length $O\left((d-t)\cdot\log\left(\frac{n}{d-t}\right)\right)$.*

Proof. We define an injective mapping $\Phi\colon \{\mathbb{F}_2^n \to \mathbb{F}_2\} \to \{\mathbb{F}_2^n \to \mathbb{F}_2\}$ that maps every degree-d polynomial p that vanishes on at most 2^{-t} of its inputs to a degree-d polynomial $\Phi(p)$ whose weight is at most 2^{-t}. Indeed, the mapping is simply $\Phi(p) = p + 1$ (i.e., for every $x \in \mathbb{F}_2^n$ it holds that $\Phi(p)(x) = p(x) + 1$). By [KLP12, Thm 14] (using the parameter values $k = d - t + 1$ and $\epsilon = 2^{-t}$), the number of polynomials with weight at most 2^{-t} is at most $2^{O(d^2\cdot t/(d-t+1)!\cdot n^{d-t+1})}$. Since Φ is injective, the number of polynomials that vanish on at most 2^{-t} of their inputs is also at most $N = 2^{O(d^2\cdot t/(d-t+1)!\cdot n^{d-t+1})}$.

Thus, a set of $O(\log(N)) = O(d^2 \cdot t/(d-t+1)! \cdot n^{d-t+1})$ uniformly-chosen elements in $\mathbb{F}_2^n$ "hits", with high probability, every polynomial that vanishes on at most 2^{-t} of its inputs. The seed length required to sample from such a set is

$$\begin{aligned} &O\Big((d-t+1)\cdot\log(n) + \log(d\cdot t) - (d-t)\cdot\log(d-t)\Big) \\ &\qquad = O\Big((d-t+1)\cdot\log(n) - (d-t)\cdot\log(d-t)\Big) \qquad (d\cdot t \leq n^2) \\ &\qquad = O\Big((d-t)\cdot\log(n/(d-t))\Big)\,. \end{aligned}$$

■

We mention that Abbe, Shpilka, and Wigderson [ASW15] proved a tighter upper-bound on the number of polynomials with low weight, which replaces the d^2 term in the result in [KLP12, Thm 14] by a smaller term. It is still an open problem to replace this term by some universal constant (such a result would match a lower bound from [KLP12, Lem 15]). However, even a solution to this open problem would not improve the result in Theorem 3.6.8.[57]

To construct an *explicit* (i.e., polynomial-time computable) hitting-set generator for polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish rarely, we generalize a construction from [GW14].

[57] This is because in our application we refer to the seed length, in which case the term d^2 only "contributes" the term $\log(d\cdot t) < 2\cdot\log(n)$, which is dominated by the term $O((d-t)\cdot\log(n))$.

For the construction we will need the pseudorandom generator of Viola [Vio09b] for low-degree polynomials.

Theorem 3.6.9 (Viola's PRG for low-degree polynomials [Vio09b])**.** *For $n, d' \in \mathbb{N}$ and $\epsilon > 0$, there exists a polynomial-time computable pseudorandom generator for polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ of degree d' with seed length $d' \cdot \log(n) + O(d' \cdot 2^{d'} \cdot \log(1/\epsilon))$.*

Theorem 3.6.10 (explicit hitting-set generator for $\mathbb{F}_2$ polynomials that vanish rarely)**.** *For every $n, d, t \in \mathbb{N}$ such that $d > t + 4$ there exists a polynomial-time computable hitting-set generator with seed length $O\left((d-t) \cdot \left(2^{d-t} + \log(n)\right)\right)$ for the set of polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ of degree d that vanish on at most 2^{-t} of their inputs.*

Proof. We show that for every polynomial $p \colon \mathbb{F}_2^n \to \mathbb{F}_2$ of degree d that vanishes on at most 2^{-t} of its inputs there exists a distribution $\mathbf{h}$ over polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ of degree $(d-t)+4$ such that for every $x \in \mathbb{F}_2^n$ it holds that $\Pr[p(x) = \mathbf{h}(x)] \geq 15/16$. Then, we use Lemma 3.2.4 to deduce that any pseudorandom generator with error $1/16$ for polynomials of degree $(d-t)+4$ is also a pseudorandom generator for p with error less than $1/2$ (and is thus a hitting-set generator for p, which vanishes on at most half of its inputs). In particular, we use the pseudorandom generator from Theorem 3.6.9 for polynomials of degree $d - t + 4$, which has seed length $O\left((d-t) \cdot \left(2^{d-t} + \log(n)\right)\right)$.

To define the distribution $\mathbf{h}$, recall that the discrete directional derivative operator on polynomials $p \colon \mathbb{F}_2^n \to \mathbb{F}_2$ for direction $a \in \mathbb{F}_2^n$ is defined by $\Delta_a(p) = p(x+a) + p(x)$. The iterated operator for $\vec{a} = a^{(1)}, \ldots, a^{(k)} \in \mathbb{F}_2^{n \cdot t}$ is defined in the natural way, and $\Delta_{\vec{a}}(p) = \sum_{S \subseteq [k]} p\left(x + \sum_{i \in S} a^{(i)}\right)$. For $k = t - 4$, sampling $h \sim \mathbf{h}$ is done by uniformly and independently choosing $\vec{a} = a^{(1)}, \ldots, a^{(k)} \in \mathbb{F}_2^n$, and outputting the polynomial

$$h = h_{\vec{a}} = \Delta_{\vec{a}}(p) + 1.$$

Note that h is of degree $d - k = (d-t) + 4$, and that for every $x \in \mathbb{F}_2^n$, the probability that $\mathbf{h}(x) = p(x)$ is at least $15/16$. This is the case since for every fixed $x \in \mathbb{F}_2^n$, if for every non-empty $S \subseteq [k]$ it holds that $p(x + \sum_{i \in S} a^{(i)}) = 1$ then $h(x) = p(x) + (2^k - 1) + 1 = p(x)$; and for every non-empty $S \subseteq [k]$, the probability over the choice of h that $p(x + \sum_{i \in S} a^{(i)}) = 1$ is at least $1 - 2^{-t}$. ∎

3.6.4 Lower bounds over general finite fields

In this section we prove our lower bounds on the seed length of HSGs for polynomials that vanish rarely. First, in Section 3.6.4.1 we give the general framework for deriving lower bounds from low-degree dispersers, corresponding to the high-level description in Section 3.6.2.1 (i.e., we prove Lemma 3.6.7). Then, we prove three incomparable lower bounds, by instantiating this framework with specific dispersers that are suitable for the corresponding parameter settings.

Our first and main lower bound, which is presented in Section 3.6.4.2, is a generalization of Theorem 3.6.1. This lower bound is of the form $\Omega((d/t) \cdot \log(n^{1-\Omega(1)} t/d))$,

and holds under complicated conditions on the degree d and on t; in particular, for $d \leq n^{.49}$ as in Theorem 3.6.1, it holds for all values of t up to $\Omega(d)$. (See Theorem 3.6.14.)

Then, in Section 3.6.4.3 we prove two additional lower bounds, which hold in two more specific settings but have advantages over the foregoing bound. The first lower bound holds only when $d \leq q$ (i.e., when the corresponding Reed-Muller code has distance $\Omega(1)$); this lower bound is of the same form as in Theorem 3.6.14, but holds for higher degrees up to $d \leq n^{1-\Omega(1)}$ without complicated conditions on d and t (see Theorem 3.6.19). The second lower bound holds only over fields of constant size; this lower bound is of the stronger form $\Omega((d/t) \cdot \log(nt/d))$,[58] and holds for degrees d up to $\Omega(n)$, but only for value of $t \lessapprox \sqrt{d}$ (see Theorem 3.6.20).

3.6.4.1 Sampling from the seeds of a disperser

In this section we prove general results that use low-degree dispersers to reduce hitting arbitrary polynomials to hitting polynomials that vanish rarely (and thus deduce lower bounds for the latter); this follows the high-level explanations that were presented in Section 3.6.2.1. The following proposition specifies the reduction itself, and the subsequent corollary specifies the lower bounds that we can obtain using the reduction.

Proposition 3.6.11 (reducing hitting polynomials to hitting polynomials that vanish rarely by sampling from the seeds of a disperser)**.** *Let $m, d_0 \in \mathbb{N}$, let $\mathbb{F}$ be a field of size q, and let $\delta = \delta_{RM}(d_0, q)$. For $k < \log(q^n)$, let $\epsilon = 2^k/q^n$, let $\rho < 1 - \epsilon$, and let $r = \log_q(1/\rho)$. Assume that:*

1. *There exists a (k, δ)-disperser $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ of degree $d_{\mathsf{Disp}} \in \mathbb{N}$.*
2. *There exists a hitting-set $W \subseteq \mathbb{F}^n$ for polynomials $\mathbb{F}^n \to \mathbb{F}$ of degree $d = 2d_0 \cdot r \cdot d_{\mathsf{Disp}}$ that vanish on at most $\sqrt{\rho + \epsilon}$ of their inputs.*

Then, there exists a hitting-set $W_0 \subseteq \mathbb{F}^m$ for polynomials $\mathbb{F}^m \to \mathbb{F}$ of degree d_0 such that $|W_0| \leq |W| \cdot 2^\ell$.

Proof. For $L = 2^\ell$, let $W_0 = \{\mathsf{Disp}(z, i) : z \in W, i \in [L]\}$. We will prove that W_0 is a hitting-set for polynomials $\mathbb{F}^m \to \mathbb{F}$ of degree d_0.

To do so, fix any non-zero polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ of degree d_0. Let $V = \{x \in \mathbb{F}^m : f(x) = 0\}$ be the set of points on which f vanishes, and let $\mathrm{G} = \{z \in \mathbb{F}^n : \exists i \in [L], \mathsf{Disp}(z, i) \notin V\}$ be the set of inputs $z \in \mathbb{F}^n$ for Disp such that for some $i \in [L]$ it holds that f does not vanish on $\mathsf{Disp}(z, i)$. Note that G has density at least $1 - \epsilon$; this is the case since $|V|/q^m \leq 1 - \delta$ (and recall that δ is the distance of the corresponding Reed-Muller code and f is non-zero), and since Disp is a (k, δ)-disperser.

[58] Recall, from Corollary 3.6.13, that this is the lower is that would be obtained if there exists a linear disperser with optimal parameters.

Note that W_0 is a hitting-set for f if and only if $\Pr_{z\in W}[z \in \mathrm{G}] > 0$. We will prove that $\Pr_{z\in W}[z \in \mathrm{G}] > 0$ using Corollary 3.2.6. To construct the distribution $\mathbf{p}$ over polynomials in $\mathbb{F}^n \to \mathbb{F}$ needed for the hypothesis of the lemma, fix a multivalued OR polynomial $\mathtt{mvOR}\colon \mathbb{F}^r \to \mathbb{F}$ of degree less than $2r$ as in Proposition 2.3.8. Then, sampling $p \sim \mathbf{p}$ is equivalent to the following random process:

> Uniformly and independently choose $\alpha^{(1)}, ..., \alpha^{(r)} \in \mathbb{F}^L$, and output the polynomial $p(z) = \mathtt{mvOR}\left(\sum_{i\in[L]} \alpha_i^{(1)} \cdot f(\mathsf{Disp}(z,i)), ..., \sum_{i\in[L]} \alpha_i^{(r)} \cdot f(\mathsf{Disp}(z,i))\right)$.

Note that each $p \sim \mathbf{p}$ has degree less than $d = d_{\mathsf{Disp}} \cdot d_0 \cdot 2r$. Also note that for any $z \notin \mathrm{G}$ we have that $\Pr[\mathbf{p}(z) = 0] = 1$, whereas for any $z \in \mathrm{G}$ we have that $\Pr[\mathbf{p}(z) \neq 0] \geq 1 - q^{-r} = 1 - \rho$. Using Corollary 3.2.6 with parameters ϵ and ρ as in our hypothesis and with $\gamma = \mu = 0$, relying on the hypotheses that W is a hitting-set for polynomials that vanish on at most $\sqrt{\rho + \epsilon}$ of their inputs and that $\rho < 1 - \epsilon$, we deduce that $\Pr_{z\in W}[z \in \mathrm{G}] > 0$, as we wanted. ∎

Using the reduction from Proposition 3.6.11, and relying on the unconditional lower bound from Fact 2.4.12, we obtain the following result, which uses low-degree dispersers to deduce lower bounds on HSGs for polynomials that vanish rarely:

Corollary 3.6.12 (a lower bound by sampling from the seeds of a disperser)**.** *Let $m, d_0 \in \mathbb{N}$ such that $d_0 < m$, let $\mathbb{F}$ be a field of size q, and let $\delta = \delta_{RM}(d_0, q)$. For $t \in \mathbb{N}$ and $k = (n - 2t) \cdot \log(q)$, assume that there exists a linear (k, δ)-disperser $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$. Then, any hitting-set $W \subseteq \mathbb{F}^n$ for polynomials in $\mathbb{F}^n \to \mathbb{F}$ of degree $d = 4d_0 \cdot t$ that vanish on at most $\sqrt{2} \cdot q^{-t}$ of their inputs has size at least $\binom{m+d_0}{d_0} \cdot 2^{-\ell}$. In particular, the seed length for any such hitting-set is at least*

$$\Omega\left(\frac{d}{t} \cdot \log\left(\frac{m \cdot t}{d}\right)\right),$$

provided that $t \leq \frac{\log(mt/d)}{8\ell} \cdot d$.

Proof. We use Proposition 3.6.11 with the parameter values $\epsilon = \rho = q^{-2t} \leq 1/4$ (such that $r = 2t$) and $d_{\mathsf{Disp}} = 1$, and rely on the fact that any hitting-set $W_0 \subseteq \mathbb{F}^m$ for all polynomials $\mathbb{F}^m \to \mathbb{F}$ of degree d_0 has size at least $\binom{m+d_0}{d_0}$ (i.e., on Fact 2.4.12). The seed length (in bits) for sampling from the hitting-set is thus at least $d_0 \cdot \log(m/d_0) - \ell = \frac{d}{4t} \cdot \log(4mt/d) - \ell \geq \Omega((d/t) \cdot \log(mt/d))$, where the last inequality is due to the hypothesis that $\frac{d}{4t} \cdot \log(mt/d) \geq 2\ell$. ∎

Finally, note that if there exists a linear (k, δ)-disperser $\mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ with optimal parameters, then we get a lower bound of $\Omega((d/t) \cdot \log(nt/d))$ for essentially all settings of the parameters. That is:

Corollary 3.6.13 (lower bounds assuming an optimal linear disperser)**.** *Assume that for every $n, q, k \in \mathbb{N}$ and $\delta > 0$ there exists a* linear *(k, δ)-disperser* $\mathsf{Disp}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ *where $\ell = \log(n \cdot \log(q) - k) + \log(1/\delta) + O(1)$ and $m \cdot \log(q) = k + \ell - \log\log(1/\delta) - O(1)$. Then, for every constant $c > 1$ there exists a constant $\gamma > 0$ such that the following holds.*

Let $n, q, d, t \in \mathbb{N}$ such that $q \leq 2^{n^c}$, and $d < n/2$, and $t \leq \gamma \cdot n$, and $\frac{q-1}{\log(q)} \cdot \log(nt/d) \geq 1/\gamma$. Then, the seed length of any HSG for $\mathcal{P}_{n,q,d,\sqrt{2}\cdot q^{-t}}$ is at least $\Omega\left(\frac{d}{t} \cdot \log\left(\frac{n \cdot t}{d}\right)\right)$.

Proof. Let $d_0 = d/4t$, and let $a = d_0/(q-1)$ such that $\delta = \delta_{RM}(d_0, q) \geq q^{-a}$. When instantiating the hypothesized linear disperser with parameters n and $k = (n - 2t) \cdot \log(q)$ and $\delta = q^{-a}$, it has seed length $\ell = O(\log(t \cdot \log(q)) + (d/4t) \cdot (\log(q)/(q-1)))$ and output length $m = \Omega(n)$. Relying on Corollary 3.6.12, we get a lower bound of $\Omega\left((d/t) \cdot \log(n \cdot (t/d))\right)$, assuming that $d_0 < m$ (which holds since we assumed that $d < n/2$) and that $t \leq \frac{\log(nt/d)}{8\ell} \cdot d$. Thus, we just need to verify the latter condition.

We verify the condition by a case analysis. The first case is when $t \geq \sqrt{d/4(q-1)}$, which implies that the seed length is $\ell = O(\log(t \cdot \log(q)))$. The condition in this case holds since $\log(nt/d) = \Omega(\log(n))$ and $q \leq 2^{\mathrm{poly}(n)}$, which implies that $\frac{\log(nt/d)}{8\ell} = \Omega(1)$. The second case is when $t < \sqrt{d/4(q-1)}$, which implies that the seed length is $\ell = O((d/t) \cdot \log(q)/(q-1))$. The condition in this case holds if and only if $\frac{q-1}{\log(q)} \cdot \log(nt/d)$ is larger than a sufficiently large constant, which is our hypothesis. ∎

3.6.4.2 The main lower bound: Proof of Theorem 3.6.1

In this section we prove lower bounds that hold also when the degree is much larger than the field size (i.e., $d \gg q$). Specifically, we will prove the following, more general version of Theorem 3.6.1:

Theorem 3.6.14 (a lower bound using the Shaltiel-Umans linear disperser; a more general version of Theorem 3.6.1)**.** *For any two constants $\gamma > 0$ and $\gamma' > 0$ there exists a constant $\gamma'' > 0$ such that the following holds. Let $n, d, t, q \in \mathbb{N}$ such that $q \leq n^{1/\gamma'}$ is a prime power, $d \leq n/4$, and:*

- *(essentially all values of $\epsilon = q^{-t}$) $t \leq \gamma'' \cdot \frac{\log(nt/d)}{\log(n)} \cdot d$.*
- *(auxiliary condition that holds for typical settings) $\frac{q-1}{\log(q)} \cdot \log(nt/d) \geq 1/\gamma''$.*
- *(main condition: d/t is upper-bounded) $d/t \leq \gamma'' \cdot \min\left\{\frac{q-1}{\log(q)} \cdot n^{\gamma}, n^{1-(\gamma+\gamma')}\right\}$.*

Then, the seed length of any HSG for $\mathcal{P}_{n,q,d,\sqrt{2}\cdot q^{-t}}$ is at least $\Omega\left(\frac{d}{t} \cdot \log\left(\frac{n^{1-(\gamma+\gamma')} \cdot t}{d}\right)\right)$.

To deduce Theorem 3.6.1 from Theorem 3.6.14, note that if we are willing to assume that $d \leq n^{.49}$, then we can choose $\gamma = .499$ and $\gamma' > 0$ that is sufficiently small, and the three conditions in Theorem 3.6.14 hold for every $q \leq n^{1/\gamma'}$ and $t \leq \gamma'' \cdot d$.

To prove Theorem 3.6.14 we will instantiate Corollary 3.6.12 with a linear disperser that we will construct relying on the extractor of Shaltiel and Umans [SU05]. Recall that [SU05] constructed an extractor $\mathtt{Ext} : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ by first constructing what they called a q-ary extractor, whose output lies in a field of size $\mathrm{poly}(n)$ and only satisfies a relatively-weak unpredictability requirement, and then transforming the q-ary extractor to a standard extractor over the binary alphabet (the transformation follows an idea of Ta-Shma, Zuckerman, and Safra [TSZS06]).

We want to construct a low-degree disperser $\mathrm{Disp}: \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q$ where the field $\mathbb{F}_q$ is of size much smaller than $\mathrm{poly}(n)$ (i.e., $q \leq n^{\gamma'}$ for some small constant $\gamma' > 0$). To do so, we take as a starting-point their construction of a q_0-ary extractor from [SU05], where $q_0 = \mathrm{poly}(n)$, and then generalize their transformation of q_0-ary extractors to standard extractors (and in particular dispersers) such that the resulting extractor is both over the field $\mathbb{F}_q$, rather than over a binary alphabet, and linear.

Towards presenting the construction, let us first recall the definition of q_0-ary extractors and the main construction of such objects from [SU05].

Definition 3.6.15 (q_0-ary extractor)**.** *For $n, k, m, \ell \in \mathbb{N}$ and $\rho > 0$, and a prime power $q_0 \in \mathbb{N}$, we say that $\mathtt{Ext}_0 : \mathbb{F}_{q_0}^n \times \{0,1\}^\ell \to \mathbb{F}_{q_0}^m$ is a (k,ρ) q_0-ary extractor if for every random variable $\mathbf{x}$ over $\mathbb{F}_{q_0}^n$ with min-entropy at least k, and every $i \in [m]$, and every function $P : \mathbb{F}_{q_0}^{i-1} \to \mathbb{F}_{q_0}^{\rho^{-2}}$, it holds that $\Pr_{x\sim\mathbf{x}, u\sim\mathbf{u}_\ell}[P(\mathtt{Ext}_0(x,u)_1, ..., \mathtt{Ext}_0(x,u)_{i-1}) \ni \mathtt{Ext}_0(x,u)_i] \leq \rho$.*

Theorem 3.6.16 ([SU05, Thm 4.5, Item 1])**.** *There exists a universal constant $c > 1$ such that the following holds. Let $n_0, q_0, k, m, r, h \in \mathbb{N}$ and $\rho > 0$ such that q_0 is a prime power, and the following inequalities hold:*

1. *(Sufficiently large auxiliary parameters h and r) $n_0 \leq \binom{h+r-1}{r}$.*
2. *(Sufficiently large field) $q_0 \geq c \cdot \frac{(h\cdot r)^2}{\rho^4}$.*
3. *(Sufficiently small output length) $m \leq \frac{k - \log(1/\rho)}{c\cdot h\cdot r\cdot \log(q_0)}$.*

Then, there exists an $r \times r$ matrix A over $\mathbb{F}_{q_0}$ such that the following holds. Let $\mathtt{Ext}_0 : \mathbb{F}_{q_0}^{n_0} \times \{0,1\}^{r\cdot\log(q_0)} \to \mathbb{F}_{q_0}^m$ be defined by $\mathtt{Ext}_0(x,v) = p_x(A^1\cdot v) \circ p_x(A^2\cdot v) \circ ... \circ p_x(A^m \cdot v)$, where v is interpreted as an element in $\mathbb{F}_{q_0}^r$, and $p_x : \mathbb{F}_{q_0}^r \to \mathbb{F}_{q_0}$ is the r-variate polynomial of total degree $h-1$ whose coefficients are specified by x. Then, $\mathtt{Ext}_0$ is a (k,ρ) q_0-ary extractor.

Note that in [SU05] the input of the extractor is represented in binary and interpreted as n_0 elements in $\mathbb{F}_q$, whereas in Theorem 3.6.16 we considered the input as n_0 elements in $\mathbb{F}_q$. The two formulations are equivalent, since a random variable over $\mathbb{F}_{q_0}^{n_0}$ has min-entropy k if and only if the corresponding random variable over $\{0,1\}^{n_0\cdot\log(q_0)}$ has min-entropy k. Also note that [SU05, Lem 4.4] showed that A can be constructed in time $q_0^{O(r)}$ (by an exhaustive search over the field $\mathbb{F}_{(q_0)^r}$), and deduced that the extractor is efficiently computable; however, we will not use this property of the extractor.

We now present the transformation of q_0-ary extractors to standard extractors whose inputs and outputs are vectors over $\mathbb{F}_q$, where $q \ll q_0$; as mentioned above, the proof generalizes an idea from [TSZS06]. The intuition for this transformation is the following. Consider the output distribution of a q_0-ary extractor as consisting of blocks of elements from $\mathbb{F}_q$, where each block represents a single element from $\mathbb{F}_{q_0}$; by definition, the output distribution of a q_0-ary extractor is "next-element unpredicatable", and hence the distribution of elements from $\mathbb{F}_q$ is a *block source* (see, e.g., [Vad12, Section 6.3.1]). Following Nisan and Zuckerman [NZ96], we compose the q_0-ary extractor with a strong extractor over $\mathbb{F}_q$ that outputs a single element (and maps each block to a single element) and obtain an extractor over $\mathbb{F}_q$. We will specifically use a single-output extractor that is obtained from a *linear list-decodable code* (see, e.g., [TSZ04, Claim 4.1]), relying on well-known constructions of such codes.[59]

Proposition 3.6.17 (transforming a q_0-ary extractor into a standard extractor over $\mathbb{F}_q$)**.** *Let $\rho > 0$, let q be a prime power, let $q_0 = q^\Delta$ for some $\Delta \in \mathbb{N}$, and let $\mathfrak{C} : \mathbb{F}_q^\Delta \to \mathbb{F}_q^{\bar{\Delta}}$ be a $(1 - 1/q - \rho, \rho^{-2})$-list-decodable code. Assume that $\mathtt{Ext}_0 : \mathbb{F}_{q_0}^{n_0} \times \{0,1\}^{\ell_0} \to \mathbb{F}_{q_0}^m$ is a (k, ρ) q_0-ary extractor. Let $\mathtt{Ext} : \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$, where $n = n_0 \cdot \Delta$ and $\ell = \ell_0 + \log(\bar{\Delta})$, be defined by*

$$\mathtt{Ext}(x,(y,j)) = \mathfrak{C}(\mathtt{Ext}_0(\hat{x},y)_1)_j \circ \ldots \circ \mathfrak{C}(\mathtt{Ext}_0(\hat{x},y)_m)_j \ ,$$

where $\hat{x} \in \mathbb{F}_{q_0}^{n_0}$ is the vector that is represented by $x \in \mathbb{F}_q^{n_0 \cdot \Delta}$. Then, $\mathtt{Ext}$ is a $(k, 2qm \cdot \rho)$-extractor.

Proof. Assuming towards a contradiction that $\mathtt{Ext}$ is not a $(k, 2qm \cdot \rho)$-extractor, we will show that $\mathtt{Ext}_0$ is not a (k, ρ) q_0-ary extractor. For simplicity, throughout the argument we do not distinguish between $x \in \mathbb{F}_q^{n_0 \cdot \Delta}$ and $\hat{x} \in \mathbb{F}_{q_0}^{n_0}$.

Since $\mathtt{Ext}$ is not a $(k, 2qm \cdot \rho)$-extractor, there exists a random variable $\mathbf{x}$ over $\mathbb{F}_q^n$ with min-entropy at least k such that $\mathtt{Ext}(\mathbf{x}, \mathbf{u}_\ell)$ is $(2qm \cdot \rho)$-far from the uniform distribution over $\mathbb{F}_q^m$. By a standard argument showing that next-element unpredictability of a distribution implies that the distribution is close to uniform (see Appendix 3.6.6.1), there exists an index $i \in [m]$ and a function $f : \mathbb{F}_q^{i-1} \to \mathbb{F}_q$ such that

$$\Pr_{x \sim \mathbf{x}, (y,j) \sim \mathbf{u}_\ell} [f(\mathtt{Ext}(x,(y,j))_1, \ldots, \mathtt{Ext}(x,(y,j))_{i-1}) = \mathtt{Ext}(x,(y,j))_i] > 1/q + 2\rho \ . \quad (3.6.1)$$

For any fixed $(x,y) \in \mathbb{F}_q^n \times \{0,1\}^{\ell_0}$, let $c_{x,y}$ be the string that is obtained by encoding each of the first $i-1$ output elements of $\mathtt{Ext}_0(x,y)$ by the code $\mathfrak{C}$; that is, $c_{x,y} = \mathfrak{C}(\mathtt{Ext}_0(x,y)_1), \ldots, \mathfrak{C}(\mathtt{Ext}_0(x,y)_{i-1}) \in (\mathbb{F}_q^{\bar{\Delta}})^{i-1}$. Also, for any $j \in [\bar{\Delta}]$, let $c_{x,y}^{(j)} \in \mathbb{F}_q^{i-1}$ be the string that is obtained by projecting each of the $i-1$ symbols of $c_{x,y}$ into its j^{th} coordinate; that is, $c_{x,y}^{(j)} = \mathfrak{C}(\mathtt{Ext}_0(x,y)_1)_j, \ldots, \mathfrak{C}(\mathtt{Ext}_0(x,y)_{i-1})_j$. Note that

$$c_{x,y}^{(j)} = \mathtt{Ext}(x,(y,j))_1, \ldots, \mathtt{Ext}(x,(y,j))_{i-1}.$$

[59] In fact, since in our case the output of the q_0-ary extractor is not only unpredictable but also unpredictable by predictors that output a *list* of elements, we use a simpler proof that does not go through the notion of strong extractors.

It follows from Equation (3.6.1) by an averaging argument that for at least a ρ-fraction of the pairs $(x,y) \in \mathbb{F}_q^n \times \{0,1\}^\ell$ it holds that

$$\begin{aligned} 1/q + \rho &< \Pr_{j \in [\bar{\Delta}]}\left[f(\mathtt{Ext}(x,(y,j))_1, ..., \mathtt{Ext}(x,(y,j))_{i-1}) = \mathtt{Ext}(x,(y,j))_i\right] \\ &= \Pr_{j \in [\bar{\Delta}]}\left[f(c_{x,y}^{(j)}) = \mathfrak{C}(\mathtt{Ext}_0(x,y)_i)_j\right]; \end{aligned}$$

in other words, with probability at least ρ over choice of (x,y), for more than a $1/q + \rho$ fraction of the coordinates $j \in [\bar{\Delta}]$ it holds that $f(c_{x,y}^{(j)})$ correctly outputs the j^{th} coordinate of $\mathfrak{C}(\mathtt{Ext}_0(x,y)_i)$.

Let us now construct a predictor $P\colon \mathbb{F}_{q_0}^{i-1} \to \mathbb{F}_{q_0}^{\rho^{-2}}$ for $\mathtt{Ext}_0$ that succeeds with probability more than ρ. The predictor P gets $i-1$ inputs $\mathtt{Ext}_0(x,y)_1, ..., \mathtt{Ext}_0(x,y)_{i-1}$, and computes $r = f\left(c_{x,y}^{(1)}\right), ..., f\left(c_{x,y}^{(\bar{\Delta})}\right) \in \mathbb{F}_q^{\bar{\Delta}}$. We think of r as a possibly-corrupt codeword in the code $\mathfrak{C}$. Since $\mathfrak{C}$ is $(1-1/q-\rho, \rho^{-2})$-list-decodable, there are at most ρ^{-2} messages whose encoding is of distance at most $1-1/q-\rho$ from r; the predictor outputs this list. By the argument above, with probability at least ρ over choice of (x,y) it holds that r will be of distance less than $1-1/q-\rho$ from $\mathfrak{C}(\mathtt{Ext}_0(x,y)_i)$. For every such (x,y), the list that P outputs will contain $\mathtt{Ext}_0(x,y)_i$. ∎

We now combine Theorem 3.6.16 and Proposition 3.6.17 to obtain a linear (k,δ)-disperser $\mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ with output length $m = k/n^{\Omega(1)}$ and seed length $\ell = O(\log(n/\delta))$.

Theorem 3.6.18 (an adaptation of the Shaltiel-Umans extractor to a linear disperser over general finite fields)**.** *For any two constants $\gamma, \gamma' > 0$ the following holds. Let $n, k, q \in \mathbb{N}$ such that $k \geq n^{\gamma+\gamma'}$ and $q \leq n^{1/\gamma'}$, and let $\delta \geq 2^{-n^\gamma + \log(2qn)}$. Then, there exists a linear (k,δ)-disperser $\mathsf{Disp} : \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$, where $\ell = O_{\gamma'}(\log(n/\delta))$ and $m = \Omega_{\gamma'}\left(k/n^{\gamma+\gamma'}\right)$.*

Proof. For a sufficiently large universal constant $c \in \mathbb{N}$, we choose q_0 to be a power of q in the interval $[(nq/\delta)^c, (nq/\delta)^{2c}]$, denote $\Delta = \log_q(q_0) = O(\log(n/\delta))$, and let $n_0 = n/\Delta$. We also let $h = \left\lceil n^{\gamma'} \right\rceil$, let $r = O(1)$ be a sufficiently large constant, let $m = c_{\gamma'} \cdot k/n^{\gamma+\gamma'}$, where $c_{\gamma'} > 0$ is a sufficiently small constant that depends on γ', and let $\rho = \delta/2qm$. We instantiate Theorem 3.6.16 with the foregoing parameters, to obtain a q_0-ary (k,ρ)-extractor $\mathtt{Ext}_0\colon \mathbb{F}_{q_0}^{n_0} \times \{0,1\}^{O(\log(n))} \to \mathbb{F}_{q_0}^m$. (The conditions of Theorem 3.6.16 hold due to our hypothesized lower bounds for k and for δ.)

We now want to use Proposition 3.6.17 to transform $\mathtt{Ext}_0$ into a standard extractor. As a list-decodable code we use the concatenation of the Reed-Solomon code with the Hadamard code over $\mathbb{F}_q$, which yields a linear code $\mathbb{F}_q^\Delta \to \mathbb{F}_q^{\bar{\Delta}}$ with relative distance $1-1/q-\rho^2$ such that $\bar{\Delta} = O(\Delta/\rho^2)^2$.[60] By an appropriate version of the Johnson

[60] We use this specific code merely for simplicity, and since its sub-optimal parameters do not significantly affect the final parameters of the construction.

bound (see, e.g., [GS01, Thm 1]), the code is $(1-1/q-\rho,\rho^{-2})$-list-decodable. Using Proposition 3.6.17 with this code, we obtain a (k,δ)-extractor $\mathtt{Ext}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$, where $\ell = O(\log(n)) + \log(\bar{\Delta}) = O(\log(n/\delta))$.

Finally, let us verify that $\mathtt{Ext}$ is linear. Recall that for any fixed seed $(y,j) \in \{0,1\}^{r\cdot\log(q_0)+\log(\bar{\Delta})}$ and output location $i \in [m]$, we want to show that the function that outputs the i^{th} output element of $\mathtt{Ext}(x,(y,j))$ is linear. To see this, note that the i^{th} output element of $\mathtt{Ext}(x,(y,j))$ can be computed from $x \in \mathbb{F}_q^n$ by first computing a predetermined output element of $\mathtt{Ext}_0(x,y)$, which we denote by $z_{y,i}(x) \in \mathbb{F}_q^\Delta$, and then computing the j^{th} output element of $\mathfrak{C}(z_{y,i}(x))$, where $\mathfrak{C} : \mathbb{F}_q^\Delta \to \mathbb{F}_q^{\bar{\Delta}}$ is a linear code. Thus, it suffices to show that the mapping of $x \in \mathbb{F}_q^n$ to $z_{y,i} \in \mathbb{F}_q^\Delta$ is $\mathbb{F}_q$-linear; this is indeed the case since $z_{y,i}(x)$ is the evaluation of the multivariate polynomial p_x over $\mathbb{F}_{q_0}$ whose coefficients are described in x (i.e., each block of Δ elements in x describes a coefficient of p_x) at the fixed point in $\mathbb{F}_{q_0}^r$ described by y. ■

Finally, we deduce our lower bound from Theorem 3.6.14 using Corollary 3.6.12 with the linear disperser from Theorem 3.6.18.

of Theorem 3.6.14. Let $d_0 = d/4t$, and let $a = d_0/(q-1)$ such that $\delta = \delta_{RM}(d_0,q) \geq q^{-a}$. We instantiate the linear disperser from Theorem 3.6.18 with parameters n and $k = (n-2t)\cdot\log(q)$ and $\delta = q^{-a} \geq 2^{-n^\gamma+\log(2qn)}$, and with the parameters $\gamma > 0$ and $\gamma' > 0$. The conditions of Theorem 3.6.18 hold due to our hypotheses that $d/t \leq \gamma'' \cdot \frac{q-1}{\log(q)} \cdot n^\gamma$ (which implies that $\delta \geq 2^{-n^\gamma+\log(2qn)}$) and that $d \leq n/4$ (which implies that $k = \Omega(n)$). For these parameters, the disperser has seed length $\ell = O(\log(n/\delta)) = O(\log(n) + (d/4t)\cdot(\log(q)/(q-1)))$ and output length $m = \Omega(n^{1-(\gamma+\gamma')})$.

Relying on Corollary 3.6.12, we get a lower bound of $\Omega\left((d/t)\cdot\log(n^{1-(\gamma+\gamma')}\cdot(t/d))\right)$, assuming that $d_0 < m$ (which holds since $d/4t < \gamma''\cdot n^{1-(\gamma+\gamma')}$) and that $t \leq \frac{\log(nt/d)}{8\ell}\cdot d$. Thus, we just need to verify the latter condition.

We verify the condition by a case analysis. The first case is when $\log(n) > \frac{d\log(q)}{4t(q-1)}$, which implies that the seed length is $\ell = O(\log(n))$; then, the condition that we want holds due to our hypothesis $t \leq \gamma''\cdot\frac{\log(nt/d)}{\log(n)}\cdot d$. In the second case we have that $\frac{d\log(q)}{4t(q-1)} \geq \log(n)$, which implies that the seed length is $\ell = O\left(\frac{d\log(q)}{t(q-1)}\right)$; then, the condition holds since we assumed that $\frac{q-1}{\log(q)}\cdot\log(nt/d) \geq 1/\gamma''$. ■

3.6.4.3 Improved lower bounds in two special cases

In this section we extend Theorem 3.6.14 by proving the two additional lower bounds that were described in the beginning of Section 3.6.4. Recall that these lower bounds have advantages over the lower bound in Theorem 3.6.14 but hold only in two specific settings.

The first lower bound is for the setting of $d \leq q$. Recall, from Section 3.6.2, that this setting is relatively easier to handle, since the corresponding Reed-Muller code

has constant relative distance. To prove the lower bound we will instantiate Corollary 3.6.12 with the disperser from Theorem 3.6.16 used with the error parameter $\delta = \Omega(1)$.[61]

Theorem 3.6.19 (a lower bound when $d \leq q$)**.** *For any constant $\eta > 0$ there exists a constant $\eta' > 0$ such that following holds. Let $n, q, d, t \in \mathbb{N}$ such that q is a prime power, and $d/t \leq \min\{3q, \eta' \cdot n^{1-2\eta}\}$, and $t \leq \eta' \cdot d$. Then, the seed length of any HSG for $\mathcal{P}_{n,q,d,\sqrt{2}\cdot q^{-t}}$ is at least $\Omega\left(\frac{d}{t} \cdot \log\left(\frac{n^{1-\eta}\cdot t}{d}\right)\right)$.*

Proof. Let $d_0 = d/4t$, and note that $d_0 \leq (3/4) \cdot q$, which implies that $\delta = \delta_{RM}(d_0, q) \geq 1/4$. We instantiate the disperser from Theorem 3.6.16 with parameters n and $k = (n - 2t) \cdot \log(q)$ and $\delta = 1/4$, and with $\gamma = \gamma' = \eta/2$. For such parameters, this disperser has seed length $\ell = O(\log(n))$ and output length $m = \Omega(n^{1-\eta})$. The statement follows using Corollary 3.6.12 with the parameters m, q, d_0, t and with this disperser; the requirement that $d_0 < m$ is satisfied since $d/t \leq \eta' \cdot n^{1-2\eta} < m$, and the requirement that $t \leq \frac{\log(mt/d)}{8\ell} \cdot d$ is satisfied since $\log(mt/d) = \Omega(\log(n))$, relying on the hypothesis that $t/d \leq n^{1-2\eta}$. ■

The second lower bound holds only over fields of constant size. Recall that this lower bound is of the stronger form $\Omega((d/t) \cdot \log(nt/d))$ (as in Corollary 3.6.13), and holds even for high degrees up to $\Omega(n)$, and for every $t \lessapprox \sqrt{d}$. More accurately:

Theorem 3.6.20 (a lower bound using the local-expander disperser)**.** *For every constant prime power q there exists a constant $\alpha_q > 0$ such that the following holds. Let $n, d, t \in \mathbb{N}$ such that $2 \cdot (q-1) \leq d \leq n/2^{2(q-1)}$ and $t \leq \alpha_q \cdot \sqrt{\log(nt/d)} \cdot \sqrt{d}$. Then, the seed length of any HSG for $\mathcal{P}_{n,q,d,\sqrt{2}\cdot q^{-t}}$ is at least $\Omega\left(\frac{d}{t} \cdot \log\left(\frac{n\cdot t}{d}\right)\right)$.*

To prove Theorem 3.6.20, we will instantiate Corollary 3.6.12 with linear dispersers that can be obtained from the recent construction of linear 1-local expanders over a constant-sized alphabet by Goldreich [Gol16], following Viola and Wigderson [VW17]. Let us first recall the definition of linear 1-local functions and Goldreich's result:

Definition 3.6.21 (linear local functions)**.** *We say that a function $f\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ is* linear 1-local *if each output bit of f is an $\mathbb{F}_q$-linear function of a single input bit of f.*

Note that the composition of linear 1-local functions is linear 1-local. Then, Goldreich [Gol16], proved that there exist expanders over $\mathbb{F}_q^n$ whose neighbor functions are 1-local $\mathbb{F}_q$-linear functions. Specifically:

Theorem 3.6.22 (local expanders [Gol16])**.** *Let $\mathbb{F}_q$ be a field of constant size. Then, for any sufficiently large $n \in \mathbb{N}$ there exists an expander (i.e., a graph with a constant spectral gap) $G = ([q^n], E)$ of degree $\Delta = O_q(1)$ that satisfies the following. For each $i \in [\Delta]$, the i^{th} neighbor function $f_i\colon [q^n] \to [q^n]$ of the graph is a linear 1-local function.*

[61] Additional lower bounds for this setting, which admit different trade-offs between the lower bound itself and the requirements on d/t, can be proved by instantiating Corollary 3.6.12 with other dispersers (e.g., with the naive disperser or with the subspace sampler). For simplicity, we omit these statements.

We now use a standard transformation of expanders to extractors: The input to the extractor is a name of a vertex, the seed specifies the directions in a walk of suitable length, and the output is the final vertex in the corresponding walk (that starts from the input vertex and proceeds according to the seed). The crucial point is that for every fixed seed, the output of the extractor is obtained by applying fixed neighbor functions (which correspond to the walk specified in the seed) to the input; in particular, since the neighbor functions are linear, the resulting disperser is also linear.

Theorem 3.6.23 (expanders yield good extractors; see, e.g., Theorem 6.22 in [Vad12])**.** *For any $q, n \in \mathbb{N}$, let $G = ([q^n], E)$ be an expander (i.e., a graph with a constant spectral gap) of degree $\Delta = O(1)$. For $k < n \cdot \log(q)$ and $\delta > 0$, let $\mathsf{Disp}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^n$, where $\ell = r \cdot \log(\Delta)$ and $r = O(n \cdot \log(q) - k + \log(1/\delta))$, be defined as follows. For every $x \in \mathbb{F}_q^m$ and $w \in \{0,1\}^\ell$, consider the r-long walk on G that starts from x, and let $\mathsf{Disp}(x,w)$ be the final vertex in this walk. Then, Disp is a (k,δ)-disperser.*

Theorem 3.6.24 (a linear disperser from a local expander)**.** *Let $\mathbb{F}_q$ be a field of constant size, let $n \in \mathbb{N}$ be sufficiently large, and for $a, t \in \mathbb{N}$ let $k = (n-2t) \cdot \log(q)$ and $\delta = q^{-a}$. Then, there exists a linear (k,δ)-disperser $\mathsf{Disp}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^n$, where $\ell = O_q(t+a)$. Moreover, the function that maps x to $\{\mathsf{Disp}(x,w)\}_{w\in\{0,1\}^\ell}$ is linear 1-local.*

Proof. We use the disperser from Theorem 3.6.23, instantiated with the expander from Theorem 3.6.22, and with error parameter $\delta = q^{-a}$ and with $k = (n-2t) \cdot \log(q)$.

To show that the mapping $x \mapsto \{\mathsf{Disp}(x,w)\}_{w\in\{0,1\}^\ell}$ is linear 1-local, fix any $w \in [2^\ell]$, and let us focus on the w^{th} output element of Disp. Recall that the w^{th} output element is the final vertex in a walk of length r that starts at the input $x \in \mathbb{F}_q^n$ to Disp and whose steps are described by w. In particular, let $f_1, ..., f_\Delta$ be the neighbor functions of G, and let $(i_1, ..., i_r) \in [\Delta]^r$ be the r steps taken in the fixed walk w; then, $\mathsf{Disp}(x)_w = f_{i_r}(f_{i_{r-1}}(...(f_{i_1}(x))...))$. Since each of the neighbor functions is a linear 1-local function, their composition is also linear 1-local. Hence, for every $w \in \{0,1\}^\ell$ it holds that $\mathsf{Disp}(\cdot)_w$ is a linear 1-local function. ∎

We now prove Theorem 3.6.20 by instantiating Corollary 3.6.12 with the linear disperser from Theorem 3.6.24:

of Theorem 3.6.20. Let $d_0 = d/4t$, and let $a = d_0/(q-1)$ such that $\delta = \delta_{RM}(d_0, q) \geq q^{-a}$. When instantiating the disperser from Theorem 3.6.24 with parameters n and $k = (n-2t) \cdot \log(q)$ and $\delta = q^{-a}$, it has seed length $\ell = O_q(t+a)$. Relying on Corollary 3.6.12, we get a lower bound of $\Omega((d/t) \cdot \log(nt/d))$, assuming that $t \leq \frac{\log(nt/d)}{8\ell} \cdot d$. Thus, we just need to verify the latter condition.

Note that $t \leq \frac{\log(nt/d)}{8\ell} \cdot d$ if and only if $t \cdot (t+a) \leq c_q \cdot \log(nt/d) \cdot d$, where c_q is a constant that depends only on q. Since $t \cdot (t+a) = t^2 + d/4(q-1)$, it suffices to prove that

$$t^2 + d/4(q-1) \leq c_q \cdot \log(nt/d) \cdot d \iff$$
$$t \leq \sqrt{c_q} \cdot \sqrt{(\log(nt/d) - 1/4(q-1))} \cdot \sqrt{d}\,.$$

Finally, since $d \leq n/2^{2(q-1)}$ we have that $\log(nt/d) - 1/4(q-1) \geq \frac{1}{2} \cdot \log(nt/d)$. Hence, it suffices that $t \leq (\sqrt{c_q/2}) \cdot \sqrt{\log(nt/d)} \cdot \sqrt{d}$, which holds due to our hypothesis (using $\alpha_q = \sqrt{c_q/2}$). ■

3.6.5 Small sets with a large degree-d closure

In this section we establish a connection between the study of HSGs for polynomials that vanish rarely, and the study of small sets with large degree-d closures, which was recently initiated by Nie and Wang [NW15]. To do so let us first define the degree-d closure of a set $S \subseteq \mathbb{F}^n$:

Definition 3.6.25 (degree-d closure)**.** *Let $\mathbb{F}$ be a finite field, and let $n, d \in \mathbb{N}$. Then, for any $S \subseteq \mathbb{F}^n$, we define the* degree-d closure of S*, denoted $\mathtt{Cl}^{(\mathtt{d})}(S)$, by $\mathtt{Cl}^{(\mathtt{d})} = \{x \in \mathbb{F}^n : \forall p \in \mathcal{I}(S), p(x) = 0\}$, where $\mathcal{I}(S) = \{p : \mathbb{F}^n \to \mathbb{F} : \deg(p) = d \wedge \forall s \in S, p(s) = 0\}$.*

We now prove Theorem 3.6.5, which shows two reductions. Loosely speaking, we show that any set with degree-d closure of size q^{n-t} is a hitting-set for polynomials that vanish with probability at most q^{-t}; and we show that any hitting-set for polynomials that vanish with probability at most q^{-t} has degree-d' closure of size $q^{n-t}/2$, for d' that is not much smaller than d.

Theorem 3.6.26 (small sets with large closures are equivalent to hitting-sets for polynomials that vanish rarely; Theorem 3.6.5, restated)**.** *Let $\mathbb{F}$ be a field of size q, let $n \in \mathbb{N}$ and $t < d < n$, and let $S \subseteq \mathbb{F}^n$. Then,*

1. *If $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| > q^{n-t}$, then S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$.*
2. *If S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$, then $\left|\mathtt{Cl}^{(\mathtt{d}/2(\mathtt{t}+1))}(S)\right| > \frac{1}{2} \cdot q^{n-t}$.*

Proof. For the first statement, let $S \subseteq \mathbb{F}^n$ be such that $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| > q^{n-t}$. Then, every degree-$d$ polynomial that vanishes on S also vanishes on more than q^{n-t} of the inputs. It follows that S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$.

For the second statement, for $d' = d/2(t+1)$, assuming that $\left|\mathtt{Cl}^{(\mathtt{d}')}(S)\right| \leq \frac{1}{2} \cdot q^{n-t}$, we construct a degree-$d$ polynomial that vanishes on S and that vanishes on at most q^{n-t} inputs in $\mathbb{F}^n$ (and it follows that S is not a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$).

To construct the polynomial, let $T_1 = \mathbb{F}^n \setminus \mathtt{Cl}^{(\mathtt{d}')}(S)$. Note that for every $x \in T_1$ there exists a degree-d' polynomial p_x that vanishes on S, but does not vanish at x. We can thus construct a collection $\mathcal{P}_1$ of degree-d' polynomials such that for every $x \in T_1$ there exists a corresponding $p_x \in \mathcal{P}_1$ satisfying $p_x(x) \neq 0$. (Indeed, a single polynomial might "cover" two distinct inputs, i.e. $p_x = p_y$ for $x \neq y$.)

Now, consider the distribution $\mathbf{p}_1$ over polynomials $\mathbb{F}^n \to \mathbb{F}$ that is defined by

$$\mathbf{p}_1(z) = \sum_{x \in T_1} \mathbf{c}_x \cdot p_x(z) \ ,$$

where the coefficients $\mathbf{c}_x$ are uniformly and independently chosen in $\mathbb{F}$. Note that $\mathbf{p}_1$ is supported by polynomials of degree d' that vanish on S. Also note that for any fixed $z \in T_1$ we have that

$$\begin{aligned}\Pr[\mathbf{p}_1(z) = 0] &= \Pr\left[\sum_{x \in T_1} \mathbf{c}_x \cdot p_x(z) = 0\right] \\ &= \mathbb{E}_{\{c_x\}_{x \in T_1 \setminus \{z\}}}\left[\Pr\left[\mathbf{c}_z \cdot p_z(z) = -\sum_{x \in T_1 \setminus \{z\}} c_x \cdot p_x(z)\right]\right],\end{aligned}$$

which equals $1/q$ since $p_z(z) \neq 0$. Therefore, there exists a fixed polynomial p of degree d' that vanishes on S and on at most $1/q$ of the inputs in T_1.

We now repeat this step t additional times, while maintaining the invariant that for every $x \in T_i$ there exists a polynomial $p_x \in \mathcal{P}_i$ such that $p_x(x) \neq 0$. Specifically, for $i = 2, ..., t+1$, we let $T_i = T_{i-1} \cap \{x \in T_i : p_{i-1}(x) = 0\}$ and $\mathcal{P}_i = \mathcal{P}_{i-1} \setminus \{p_{i-1}\}$. Note that $|T_i| \leq |T_{i-1}|/q$, and that for every $x \in T_i$ there exists $p_x \in \mathcal{P}_i$ such that $p_x(x) \neq 0$. We again define a distribution $\mathbf{p}_i(z) = \sum_{x \in T_i} \mathbf{c}_x \cdot p_x(z)$, and using the same argument as above, we deduce that there exists a fixed polynomial p_i of degree d' that vanishes on S and on at most $1/q$ of the inputs in T_i.

After $t+1$ steps we obtain $t+1$ polynomials $p_1, ..., p_{t+1}$ of degree d' that vanish on S such that $\left|\{x \notin \mathtt{Cl}^{(\mathtt{d})}(S) : \forall i \in [t], p_i(x) = 0\}\right| \leq |T_1|/q^{t+1} \leq \frac{1}{2} \cdot q^{-t}$. Let $p : \mathbb{F}^n \to \mathbb{F}$ be the multivalued OR of $p_1, ..., p_{t+1}$, defined by $p(x) = \mathtt{mvOR}(p_1(x), ..., p_t(x))$. Note that $\deg(p) < 2(t+1) \cdot d' = d$, and that p vanishes on S. Thus, denoting $\delta = \left|\mathtt{Cl}^{(\mathtt{d}')}(S)\right|/q^n \leq \frac{1}{2} \cdot q^{-t}$, we have that

$$\Pr_{x \in \mathbb{F}^n}[p(x) = 0] = \delta + (1-\delta) \cdot q^{-(t+1)} < q^{-t},$$

which implies that $p \in \mathcal{P}_{n,q,d,q^{-t}}$. Hence, S is not a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$. ■

As mentioned in Section 3.6.1.3, we can obtain an upper-bound on the size of $\mathtt{Cl}^{(\mathtt{d})}(S)$ for any sufficiently-small set S, by combining Theorem 3.6.14 and the first item of Theorem 3.6.26. Specifically, we can deduce that for every $2 \leq q \leq \mathrm{poly}(n)$ and $d \leq n^{.49}$ and $t \leq \gamma \cdot d$ (where $\gamma > 0$ is a sufficiently small constant), any set S of size $|S| \leq n^{\gamma \cdot (d/t)}$ satisfies $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| \leq q^{n-t}$. However, this corollary is superseded by the upper-bound of [NW15], who showed that for any $S \subseteq \mathbb{F}^n$ it holds that $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| \leq \frac{|S|}{\binom{n+d}{d}} \cdot q^n$.

Indeed, since the problem of constructing small sets with large degree-d closures is at least as hard as the problem of constructing HSGs for polynomials that vanish rarely (due to the first item of Theorem 3.6.26), it might be inherent that a direct lower bound on the former problem is stronger than a lower bound that is obtained via a reduction from the latter problem.

3.6.6 Appendices for Section 3.6

3.6.6.1 Next-element unpredictability over large alphabets

Recall that, as proved by Yao [Yao82], if a distribution $\mathbf{w}$ over $\{0,1\}^m$ is next-bit unpredictable, then $\mathbf{w}$ is close to the uniform distribution. In this appendix we prove a generalized version of this claim that applies also to distributions over Σ^m where Σ is an alphabet of arbitrary size.

Proposition 3.6.27 (next-element unpredictability implies closeness to uniform, over arbitrary alphabets)**.** *Let Σ be a set of size $q = |\Sigma|$, let $\mathbf{w}$ be a distribution over Σ^m, and assume that the statistical distance between $\mathbf{w}$ and the uniform distribution on Σ^m, denoted $\mathbf{u}_m$, is at least $\rho > 0$. Then, there exists $i \in [m]$ and a function $P : \Sigma^{i-1} \to \Sigma$ such that $\Pr[\mathbf{w}_i = P(\mathbf{w}_1, ..., \mathbf{w}_{i-1})] > 1/q + \rho/qm$,.*

Proof. Let $\mathbf{h}^{(0)} = \mathbf{u}_n$, and for $i \in [m]$ let $\mathbf{h}^{(i)}$ be the distribution over Σ^m such that its first i elements are sampled from $\mathbf{w}$ and its last $m - i$ elements are sampled uniformly and independently. By a standard hybrid argument, for some $i \in [m]$ it holds that the statistical distance between $\mathbf{h}^{(i-1)}$ and $\mathbf{h}^{(i)}$ is at least ρ/m. Hence, there exists $T : \Sigma^i \to \{0,1\}$ such that

$$\Pr[T(\mathbf{h}^{(i)}_{1,...i}) = 1] - \Pr[T(\mathbf{h}^{(i-1)}_{1,...,i}) = 1] > \rho/m \ .$$

Now, for any $w_1, ..., w_{i-1} \in \Sigma^{i-1}$, let

$$P(w_1, ..., w_{i-1}) = \text{argmax}_{z \in \Sigma} \left\{ \Pr\left[\mathbf{w}_i = z | \mathbf{w}_{1,...,i-1} = w_{1,...,i-1}\right] \right\} \ .$$

Denote $\Pr_{w \sim \mathbf{w}}[w_i = P(w_{1,...,i-1})] \stackrel{\text{def}}{=\!=} (1/q + \delta)$, where $\delta \in [0,1]$. Our goal is to prove that $\delta > \rho/qm$. By the definition of P, for every $z \in \Sigma$ and $w_{1,...,i-1} \in \Sigma^{i-1}$ we have that

$$\mathbb{E}_{w \sim \mathbf{w}} \left[\Pr[\mathbf{w}_i = z | \mathbf{w}_{1,...,i-1} = w_{1,...,i-1}]\right] \leq 1/q + \delta \ .$$

Thus, we have that

$$\begin{aligned}
&\Pr[T(\mathbf{h}^{(i)}_{1,...i}) = 1] - \Pr[T(\mathbf{h}^{(i-1)}_{1,...,i}) = 1] \\
&\quad = \mathbb{E}_{u_{i+1,...,n} \sim \mathbf{u}_n, w_{1,...,i-1} \sim \mathbf{w}} \Big[\sum_{z \in \Sigma} \Pr[\mathbf{w}_i = z | \mathbf{w}_{1,...,i-1} = w_{1,...,i-1}] \cdot T(w_1, ..., w_{i-1}, z, u_{i+1}, ..., u_n) \\
&\qquad\quad - \frac{1}{q} \cdot \sum_{z \in \Sigma} T(w_1, ..., w_{i-1}, z, u_{i+1}, ..., u_n) \Big] \\
&\quad \leq \mathbb{E}_{u_{i+1,...,n} \sim \mathbf{u}_n, w_{1,...,i-1} \sim \mathbf{w}} \Big[\sum_{z \in \Sigma} (1/q + \delta) \cdot T(w_1, ..., w_{i-1}, z, u_{i+1}, ..., u_n) \\
&\qquad\quad - \frac{1}{q} \cdot T(w_1, ..., w_{i-1}, z, u_{i+1}, ..., u_n) \Big] \\
&\quad \leq q \cdot \delta \ ,
\end{aligned}$$

which implies that $\delta > \rho/qm$. ■

3.6.6.2 An alternative argument for lower bounds

In this appendix we describe an alternative argument for proving a lower bound on the size of hitting-sets for polynomials that vanish rarely; this argument was suggested to us by an anonymous reviewer. We note in advance that this argument is known to work only for prime fields (for reasons that will be explained below), and that our main reason for presenting it is since it is simple and elegant. For simplicity, we first present the argument only for the field $\mathbb{F}_2$.

Recall, from the "warm-up" in Section 3.6.2.1, that a lower bound on the seed length of any hitting-set generator for $\mathcal{P}_{n,q,d,t}$ can be proved quite easily (i.e., with the naive disperser and without "randomized tests") when the corresponding Reed-Muller code has constant relative distance. The main technical ingredient underlying the alternative argument is the existence of a *large subcode* of the Reed-Muller code that has constant relative distance; the existence of such a subcode can be deduced using the following lemma by Ben-Eliezer, Hod, and Lovett [BHL12]. Towards stating the lemma, we define the bias of a function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ to be $\mathrm{bias}(f) = \Pr_{x\in\mathbb{F}_2^n}[f(x) = 0] - \Pr_{x\in\mathbb{F}_2^n}[f(x) = 1] = 2\Pr_x[f(x) = 0] - 1$. The following is showed in [BHL12]:

Lemma 3.6.28 (a random $\mathbb{F}_2$-polynomial is unbiased [BHL12, Lemma 2])**.** *For every constant $\epsilon > 0$ there exist constants $\alpha, \beta > 0$ such that the following holds. For $n \in \mathbb{N}$ and $d \leq (1-\epsilon)\cdot n$, let $\mathbf{p}$ be a uniformly-chosen degree-d polynomial in $\mathbb{F}_2^n \to \mathbb{F}_2$. Then, it holds that*

$$\Pr\left[\left|\mathrm{bias}(\mathbf{p})\right| > 2^{-\alpha\cdot(n/d)}\right] \leq 2^{-\beta\cdot\binom{n+d}{d}}.$$

Loosely speaking, the fact that a random degree-d polynomial is unbiased implies that the difference between two random degree-d polynomials is unbiased, or in other words that two random degree-d polynomials disagree on $\approx 1/2$ of their inputs. Hence, when independently choosing many random degree-d polynomials, with high probability the subcode spanned by them has distance close to $1/2$; that is, there exists a large subcode of the Reed-Muller code with relative distance close to $1/2$. In more detail:

Corollary 3.6.29 (a large subcode of the Reed-Muller code with constant relative distance)**.** *For every $\epsilon > 0$ there exists $\gamma > 0$ such that the following holds. For every sufficiently large $n \in \mathbb{N}$ and $d \leq n^{.99}$ there exists a linear subcode of the $[n, d]$ Reed-Muller code over $\mathbb{F}_2$ that has dimension at least $\gamma\cdot\binom{n+d}{d}$ and relative distance at least $1/2 - \epsilon$.*

Proof. Fix $\epsilon > 0$, and let $\gamma > 0$ be sufficiently small. For two polynomials $f_1, f_2\colon \mathbb{F}_2^n \to \mathbb{F}_2$, let $\mathtt{agr}(f_1, f_2) = \Pr_{x\in\mathbb{F}_2^n}[f_1(x) = f_2(x)] = \Pr_x[f_1(x) - f_2(x) = 0] = \frac{1}{2} + \mathrm{bias}[f_1 - f_2]/2$. Denoting a uniformly-chosen degree-d polynomial $\mathbb{F}_2^n \to \mathbb{F}_2$ by $\mathbf{p}$, we choose $D' = \gamma\cdot\binom{n+d}{d}$ polynomials $\mathbf{b}_1, ..., \mathbf{b}_{D'} \sim \mathbf{p}$, and denote the subcode of the Reed-Muller code spanned by these polynomials by $\mathcal{C}' = \{\mathbf{f}_1, ..., \mathbf{f}_T\}$, for $T \leq 2^{D'}$.

First note that with high probability $T = 2^{D'}$, or in other words the $\mathbf{b}_i$-s are linearly independent. This is the case since if we choose the $\mathbf{b}_i$-s sequentially, then at each

iteration $i \in [D']$, the probability that $\mathbf{b}_i$ lies in the subspace spanned by the $i-1$ previously-chosen polynomials is at most $2^{(i-1)-\binom{n+d}{d}} < 2^{D'-\binom{n+d}{d}} = o(1/D')$.

Now, conditioned on the event that $T = 2^{D'}$, note that for every fixed $i \in [2^{D'}]$ it holds that $\mathbf{f}_i$ is uniformly distributed (i.e., its marginal distribution is $\mathbf{p}$). Thus, for every fixed $i \in [2^{D'}]$ we have that

$$\begin{aligned}
\Pr[\exists j \neq i : \mathtt{agr}(\mathbf{f}_i, \mathbf{f}_j) > 1/2 + \epsilon] &= \Pr[\exists j \neq i : \mathrm{bias}(\mathbf{f}_i - \mathbf{f}_j) > 2\epsilon] \\
&= \sum_{p \in \mathtt{supp}(\mathbf{p})} \Pr[\mathbf{f}_i = p] \cdot \Pr[\exists j \neq i : \mathrm{bias}(p - \mathbf{f}_j) > 2\epsilon] \\
&< 2^{D'} \cdot \Pr[\mathrm{bias}(\mathbf{p}) > 2\epsilon] \cdot \sum_{p \in \mathtt{supp}(\mathbf{p})} \Pr[\mathbf{f}_i = p] \\
&\leq 2^{(\gamma-\beta)\cdot\binom{n+d}{d}}, \qquad \text{(Lemma 3.6.28)}
\end{aligned}$$

where $\beta = \beta(\epsilon)$ and we used the fact that $2^{-\alpha\cdot(n/d)} \leq 2\epsilon'$ for every constant $\alpha = \alpha(\epsilon)$ and large enough n. Taking $\gamma < \beta$ to be a sufficiently small constnat, the above is $o(1)$. Therefore, with high probability over choice of $\mathbf{b}_1, ..., \mathbf{b}_{D'}$, the linear subcode induced by our choice has dimension D' and relative distance at least $1/2 - \epsilon'$. ∎

We now prove the lower bound. Loosely speaking, using the simple reduction that was described in the "warm-up" in Section 3.6.2.1, we show that if there exists a small hitting-set for degree-d polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish rarely then there exists a small hitting-set for the large subcode from Corollary 3.6.29.

Theorem 3.6.30 (a lower bound using subcodes of the Reed-Muller code)**.** *There exists a universal constant $\alpha > 0$ such that the following holds. For any sufficiently large $n \in \mathbb{N}$, $d \leq n^{.99}$, and $t \in \mathbb{N}$ such that $t < \alpha \cdot d$, the seed length of any hitting-set generator for $\mathcal{P}_{n,2,d,2^{-t}}$ is at least $\Omega((d/t) \cdot \log(n))$.*

Proof. Let $\delta > 0$ be a sufficiently small constant, let $n \in \mathbb{N}$ be sufficiently large, let $d \leq (1-\epsilon) \cdot n$, and let $t < d$. Assume towards a contradiction that there exists a hitting-set $S \subseteq \mathbb{F}_2^n$ for $\mathcal{P}_{n,2,d,2^{-t}}$ of size $\delta \cdot \binom{n+d/t}{d/t}$.

Now, let $d_0 = \lfloor d/4t \rfloor$, let $t' = 2t$, and let $m = \lfloor n/t' \rfloor$. Let $\mathcal{C}' \subseteq \mathbb{F}_2^m$ be a linear subcode of the Reed-Muller code $\mathbb{F}_2^m \to \mathbb{F}_2$ of degree d_0 that has dimension $\gamma \cdot \binom{m+d_0}{d_0}$ and relative distance at least .49, whose existence is guaranteed by Corollary 3.6.29.

We construct a hitting-set for $\mathcal{C}'$ as follows. For every polynomial $p \in \mathcal{C}'$, consider the polynomial $p' \colon \mathbb{F}_2^{m\cdot t'} \to \mathbb{F}_2$ such that $p'(z) = \mathtt{mvOR}(p(z^{(1)}), ..., p(z^{(t')}))$, where we think of $z = z^{(1)}, ..., z^{(t')}$ such that for each i it holds that $z^{(i)}$ is an m-bit string. Note that the degree of p' is less than $4d_0 \cdot t \leq d$, and that $\Pr_{z \in \mathbb{F}_2^{t'\cdot m}}[p'(z) = 0] = \Pr_{x \in \mathbb{F}_2^m}[p(x) = 0]^{t'} \leq 2^{-t}$. By our assumption that S is a hitting-set for $\mathcal{P}_{n,2,d,2^{-t}}$, there exists $z \in S$ such that $p'(z) \neq 0$, which implies that for some $i \in [t']$ it holds that $p(z^{(i)}) \neq 0$.

Thus, the set $S_0 = \{z^{(i)} : z \in S, i \in [t']\}$ is a hitting-set for $\mathcal{C}'$ of size at most $2t \cdot |S|$. Now, relying on Fact 2.4.12, any hitting-set for $\mathcal{C}'$ is of size at least $\dim(\mathcal{C}') = \gamma \cdot \binom{m+d_0}{d_0}$,

and hence $|S| \geq \gamma \cdot \binom{m+d_0}{d_0}/t$. The seed length for sampling from S is thus at least

$$\Omega\left(d_0 \cdot \log((m+d_0)/d_0) - \log(t)\right) = \Omega\left((d/t) \cdot \log(n/d) - \log(t)\right) ,$$

which simplifies to $\Omega((d/t) \cdot \log(n))$ relying on the hypotheses that $d \leq n^{.99}$ and $t \leq \alpha \cdot d$, for a sufficiently small universal constant $\alpha > 0$. ■

To generalize the foregoing argument to fields other than $\mathbb{F}_2$, note that the only place where we used the fact that the field is $\mathbb{F}_2$ is when deducing the existence of a large subcode of the Reed-Muller code (i.e., in Corollary 3.6.29, which relied on Lemma 3.6.28). The argument can be generalized to any prime field, relying on a generalization of Lemma 3.6.28 to arbitrary prime fields that was recently proved by Beame, Gharan, and Yang [BGY18]. However, we are not aware of an analogous result for non-prime fields.

3.7 Limitations of two "black-box" techniques

3.7.1 The main results

A main goal in the study of quantified derandomization is to use this framework in order to solve the standard derandomization problem (i.e., CAPP). Specifically, fix a circuit class $\mathcal{C}$, and assume that we constructed an algorithm that reduces the standard derandomization problem of $\mathcal{C}$ to quantified derandomization of $\mathcal{C}$ with $B^{\mathtt{red}}$ exceptional inputs (e.g., such an algorithm can get as input a circuit $C \in \mathcal{C}$ over m bits with at most $2^m/3$ exceptional inputs, and construct a circuit $C' \in \mathcal{C}$ over $n = \mathrm{poly}(m)$ bits with at most $B^{\mathtt{red}}(n)$ exceptional inputs, such that the most frequent output of C' equals the most frequent output of C). Also assume that we constructed an algorithm for quantified derandomization of $\mathcal{C}$ that can handle $B^{\mathtt{alg}}$ exceptional inputs. Then, if $B^{\mathtt{alg}}(n) \geq B^{\mathtt{red}}(n)$, we can combine both algorithms in the straightforward way to obtain an algorithm for standard derandomization of $\mathcal{C}$.

In this section we show *a limitation of two specific natural techniques* in quantified derandomization. One technique is used to reduce standard derandomization to quantified derandomization, and is based on error-reduction using seeded extractors that are computable by circuits in $\mathcal{C}$. The other technique is used to construct quantified derandomization algorithms, and is based on pseudorandom distributions of restrictions that simplify every circuit $C \in \mathcal{C}$, with high probability. The two foregoing techniques, which are described in detail in Section 3.7.2, only rely on "black-box" access to the input circuit C; that is, the algorithm (either for the reduction or for quantified derandomization) does not use the explicit description of C, beyond the guarantee that $C \in \mathcal{C}$ and the ability to evaluate C at arbitrarily-chosen inputs.

Informally, our main theorem asserts that the straightforward combination of the two foregoing techniques cannot suffice to yield a standard derandomization algorithm. This is the case since the function $B^{\mathtt{red}}$ for quantified derandomization to which

standard derandomization is reduced is *necessarily larger* than the function $B^{\mathtt{alg}}$ that the quantified derandomization algorithm can handle. That is,

Theorem 3.7.1 (a limitation of two "black-box" techniques in quantified derandomization; informal)**.** *Assume that there exists a reduction of standard derandomization of a class $\mathcal{C}$ to quantified derandomization of $\mathcal{C}$ with $B^{\mathtt{red}}$ exceptional inputs that is based on seeded extractors that are computable in $\mathcal{C}$. Also assume that there exists a quantified derandomization algorithm for $\mathcal{C}$ with $B^{\mathtt{alg}}$ exceptional inputs that is based on a distribution over restrictions that simplifies every $C \in \mathcal{C}$ to a constant, with high probability. Then, $B^{\mathtt{alg}}(n) < B^{\mathtt{red}}(n)$.*

This result is particularly meaningful for derandomization of $\mathcal{AC}^0$. Recall, from Section 3.3, that n this setting, we know of a reduction to quantified derandomization with $B^{\mathtt{red}}$ exceptional inputs and of a quantified derandomization algorithm with $B^{\mathtt{alg}}$ inputs such that $B^{\mathtt{red}}$ and $B^{\mathtt{alg}}$ are very close (i.e., both are of the form $2^{n/\mathrm{poly}\log(n)}$). However, the algorithms for both these results rely on the aforementioned "black-box" techniques, and therefore Theorem 3.7.1 implies that one cannot hope to obtain a standard derandomization algorithm for $\mathcal{AC}^0$ by merely improving the parameters of the underlying technical results. Nevertheless, for other circuit classes, results in quantified derandomization were obtained using different techniques, *which are not "black-box"*; in particular, such results were obtained for various subclasses of $\mathcal{AC}^0[\oplus]$ and for sparse $\mathcal{TC}^0$ circuits (see Section 3.7.2 for more details).

Organization. In Section 3.7.2 we describe the two "black-box" techniques that are the focus of the current text. In Section 3.7.3 we prove the main theorem (i.e., Theorem 3.7.1). In Section 3.7.4 we discuss several relaxations of the hypotheses of the main theorem (i.e., several natural modifications of the two "black-box" techniques) that do not seem sufficient to bypass the conclusion of the theorem.

3.7.2 Two black-box techniques for quantified derandomization

Throughout the text, whenever we refer to a class $\mathcal{C}$ of circuits, we will always think of $\mathcal{C}$ as a *set of circuits*, rather than a collection of such sets. That is, we consider $\mathcal{C} = \bigcup_{n\in\mathbb{N}} \mathcal{C}_n$, where $\mathcal{C}_n$ is some fixed set of circuits over n input bits. This is done merely for simplicity of presentation, and all results extend to the standard setting (i.e., when considering circuit families) in a natural way.

3.7.2.1 Error-reduction using a $\mathcal{C}$-computable sampler

The first technique that we discuss is used to reduce standard derandomization of a circuit class $\mathcal{C}$ to quantified derandomization of $\mathcal{C}$. This technique is based on *randomness-efficient error reduction*, using seeded extractors (equivalently, averaging samplers; see, e.g., [Vad12, Cor. 6.24], [Gol08, Apdx. D.4.1.2]) that are computable by $\mathcal{C}$-circuits. Let us now properly define this notion, while slightly diverging from Definition 2.5.5 (specifically, we fix the accuracy parameter of the sampler to be $1/10$,

we measure the *number* of "bad" inputs for the sampler rather than the fraction, and we account for the circuit class in which the sampler is computable).

Definition 3.7.2 ($\mathcal{C}$-computable sampler)**.** *For $B^{\mathtt{red}} : \mathbb{N} \to \mathbb{N}$, we say that a function* $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ *is a* Boolean sampler with $B^{\mathtt{red}}$ bad inputs *if it satisfies the following: For every $T \subseteq \{0,1\}^m$, for all but at most $B^{\mathtt{red}}$ of the inputs $x \in \{0,1\}^n$ it holds that $\Pr_{i \in \{0,1\}^s}[\mathsf{Samp}(x,i) \in T] \in |T|/2^m \pm 1/10$. For a circuit class $\mathcal{C}$, we say that* Samp *is* computable in $\mathcal{C}$ *if for every fixed $i \in \{0,1\}^s$, each output bit of the function $\mathsf{Samp}^{(i)}(x) = \mathsf{Samp}(x,i)$ is computable by a circuit in $\mathcal{C}$.*

Samplers that are computable in a circuit class $\mathcal{C}$ can be used for error-reduction of $\mathcal{C}$-circuits. Specifically, assume that for every $m \in \mathbb{N}$ we can efficiently construct circuits for the output bits of a $\mathcal{C}$-computable sampler $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ with $B^{\mathtt{red}}$ bad inputs. Then, given a circuit $C \in \mathcal{C}$ over m input bits, we can efficiently construct a circuit $C' : \{0,1\}^n \to \{0,1\}$ that gets input $x \in \{0,1\}^n$, computes the 2^s outputs of the sampler on x (each output is an m-bit string), evaluates C on each of these outputs, and outputs the majority of the evaluations of C; that is, $C'(x) = MAJ\left(\{C(\mathsf{Samp}(x,i))\}_{i \in \{0,1\}^s}\right)$. If C accepts (resp., rejects) all but at most $2^m/3$ of its inputs, then C' accepts (resp., rejects) all but at most $B^{\mathtt{red}}(n)$ of its inputs.

We typically want to minimize the overhead of C' with respect to the original circuit C, since we reduce the standard derandomization problem of C to quantified derandomization of C'. Hence, typical settings of the parameters are $n = \mathrm{poly}(m)$ and $s = O(\log(n))$, such that the size of C' is polynomial in the size of C. Also observe that the majority function in the definition of C' can be replaced by an "approximate majority" (i.e., a function that distinguishes between strings with relative Hamming weight ≤ 0.49 and strings with relative Hamming weight ≥ 0.51, as in [Ajt83; Vio09a]). For further discussion of the effect of the "overhead" when constructing C', see Section 3.7.3.

Observe that the algorithm above (that constructs the circuit C') uses the same sampler Samp, regardless of the input circuit $C \in \mathcal{C}$. Thus, in order to evaluate C' at any point, the algorithm does not need the explicit description of C, but rather only "black-box" access to C (i.e., the ability to evaluate C at arbitrarily-chosen points).

This approach has been used to reduce standard derandomization to quantified derandomization in the contexts of $\mathcal{AC}^0$ circuits (see Section 3.3, which uses the extractor of [CL16]), of $\mathcal{AC}^0[\oplus]$ circuits (see [GW14, Thm 1.4] and Section 3.4), and of $\mathcal{TC}^0$ circuits (see Section 3.5). A somewhat different approach for error-reduction was taken in [GW14, Thm 3.4 in the full version]; their approach is also insufficient to bypass the limitation in our main theorem, but is interesting in its own right (see further discussion in Section 3.7.4.2).

3.7.2.2 A quantified derandomization algorithm that uses pseudorandom restrictions

The second technique that we discuss can be used to construct a quantified derandomization algorithm. This technique relies on the existence of a distribution $\mathbf{S}_n$ over

subsets of $\{0,1\}^n$ (i.e., over "restrictions") such that for any $C \in \mathcal{C}$ over n input bits, with high probability over $S \sim \mathbf{S}_n$ it holds that $C|_S$ is constant. That is,

Definition 3.7.3 (simplifier sets)**.** *For $B^{\mathtt{alg}} : \mathbb{N} \to \mathbb{N}$, we say that a distribution $\mathbf{S}_n$ over subsets of $\{0,1\}^n$ is* a distribution of simplifier sets of size more than $B^{\mathtt{alg}}$ for $\mathcal{C}$ *if the following conditions hold:*

1. *Every subset S in the support of $\mathbf{S}_n$ is of size $|S| > B^{\mathtt{alg}}(n)$.*
2. *For every $C \in \mathcal{C}$ over n input bits, $\Pr_{S \sim \mathbf{S}_n}[C|_S$ is constant$] > 1/2$.*

To see why simplifier sets are useful for quantified derandomization, let $C \in \mathcal{C}$ be a circuit over n bits with $B^{\mathtt{alg}}(n)$ exceptional inputs. Then, with probability more than $1/2$ over $S \sim \mathbf{S}_n$ it holds that $C|_S$ is a constant function; and since $B^{\mathtt{alg}}(n) < |S|$, whenever $C|_S$ is a constant function, this constant equals the most frequent output of C. Now, assume that we can efficiently sample a succint representation of a set $S \sim \mathbf{S}_n$ using only a few (say, $O(\log(n))$) random bits, and that this representation allows to efficiently find some input $x \in S$. In this case, we can solve the quantified derandomization problem as follows: We enumerate the choices of $S \sim \mathbf{S}_n$, evaluate C on an (arbitrary) input in each choice of S, and output the majority value among the evaluations of C. Note that the only information about C that the algorithm in this approach uses, other than the fact that $C \in \mathcal{C}$, is the ability to evaluate C on arbitrarily-chosen inputs.

This ("black-box") approach has been used to construct quantified derandomization algorithms for $\mathcal{AC}^0$ (see [GW14, Thm 1.3] and Section 3.3) and for $\mathcal{TC}^0$ circuits of depth two (see Section 3.5.5). Nevertheless, for other circuit classes, *pseudorandom restriction algorithms that are "non-black-box"* have been constructed and used for quantified derandomization; these classes include various subclasses of $\mathcal{AC}^0[\oplus]$ (see Section 3.4) and sparse $\mathcal{TC}^0$ circuits (see Section 3.5).

3.7.3 Proof of the main theorem

Towards formally stating and proving Theorem 3.7.1, let us now carefully track how the combination of the two techniques from Section 3.7.2 works. We are interested in standard derandomization of a circuit class $\mathcal{C}$. That is, we are given a circuit $C \in \mathcal{C}$ over m input bits, and want to distinguish between the case that C accepts all but at most $2^m/3$ of its inputs and the case that C rejects all but at most $2^m/3$ of its inputs.

To do so, we fix some sampler $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ with $B^{\mathtt{red}}$ bad inputs, and consider the "error-reduced" circuit $C' : \{0,1\}^n \to \{0,1\}$ such that $C'(x) = MAJ\left(\{C(\mathsf{Samp}(x,i))\}_{i \in \{0,1\}^s}\right)$. [62] We think of the circuit C' as belonging to some new circuit class, which we denote by $\widehat{\mathcal{C}}$; for example, we can define $\widehat{\mathcal{C}}$ to be the class of circuits of the form $C''(x) = \Phi\left(\{C_0(\mathsf{Samp}(x,i)\}_{i \in \{0,1\}^s}\right)$, where $C_0 \in \mathcal{C}$ and Φ is one of several "simple composition" functions (the majority function being one of them).

[62] Recall that the majority function can also be replaced by "approximate majority".

Indeed, it might be the case that $\widehat{\mathcal{C}} \subseteq \mathcal{C}$, if $\mathcal{C}$ is closed to the overhead involved in constructing circuits such as C', but we do not assume so. However, we assume that Samp is computable in $\widehat{\mathcal{C}}$, which holds under reasonable definitions of $\widehat{\mathcal{C}}$. [63]

Now we are interested in quantified derandomization of the class $\widehat{\mathcal{C}}$ with $B^{\mathtt{red}}$ bad inputs, using simplifier sets. However, the following lemma asserts that in any distribution of simplifier sets of size more than $B^{\mathtt{alg}}$ for a class that can compute Samp (and in particular for $\widehat{\mathcal{C}}$), the size of the sets satisfies $B^{\mathtt{alg}}(n) < B^{\mathtt{red}}(n)$. When reading the lemma's statement, we encourage the reader to think of a sampler $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ such that $s = O(\log(n))$ and $m = n^{\Omega(1)}$.

Lemma 3.7.4 (simplifier sets and samplers)**.** *Let $\widehat{\mathcal{C}}$ be a circuit class, and let $B^{\mathtt{red}} : \mathbb{N} \to \mathbb{N}$ and $B^{\mathtt{alg}} : \mathbb{N} \to \mathbb{N}$. For $n \in \mathbb{N}$ and $s, m \in \mathbb{N}$ such that $2^{s+3m/4} \leq 2^m/5$, assume that there exists a Boolean sampler $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ with $B^{\mathtt{red}}$ bad inputs that is computable in $\widehat{\mathcal{C}}$. Also assume that there exists a distribution $\mathbf{S}_n$ of simplifier sets of size more than $B^{\mathtt{alg}}$ for $\widehat{\mathcal{C}}$. Then $B^{\mathtt{alg}}(n) < B^{\mathtt{red}}(n)$.*

Proof. Assuming that a distribution $\mathbf{S}_n$ as in the hypothesis exists, we will construct a set $T \subseteq \{0,1\}^m$ such that for more than $B^{\mathtt{alg}}(n)$ inputs $x \in \{0,1\}^n$, the set T is "over-sampled" by the sampler Samp with input x; that is, for every such x it holds that $\Pr_{i \in \{0,1\}^s}[\mathsf{Samp}(x,i) \in T] > |T|/2^m + 1/10$. Thus, the number $B^{\mathtt{red}}(n)$ of "bad" inputs for the sampler is larger than $B^{\mathtt{alg}}(n)$.

In order to construct $T \subseteq \{0,1\}^m$, we will first fix a single set $S \subseteq \{0,1\}^n$ of size $|S| > B^{\mathtt{alg}}(n)$ of inputs for the sampler; these inputs will later on be the ones that will cause the sampler to "over-sample" T. To do so, for any fixed choice of $S \sim \mathbf{S}_n$, let us call an index $i \in \{0,1\}^s$ $\mathtt{bad\ under}$ S if at least a quarter of the output bits of $\mathsf{Samp}^{(i)}{\restriction_S}$ are constant functions (i.e., when restricting the function $\mathsf{Samp}^{(i)}(x) = \mathsf{Samp}(x,i)$ to the set S, at least $m/4$ of the m output bits of $\mathsf{Samp}^{(i)}{\restriction_S}$ become constant functions). For any $i \in \{0,1\}^s$, recall that each output bit of $\mathsf{Samp}^{(i)}$ can be computed by a $\widehat{\mathcal{C}}$-circuit, and therefore (since $\mathbf{S}_n$ simplifies $\widehat{\mathcal{C}}$) the expected number of output bits of $\mathsf{Samp}^{(i)}$ that become constant under $S \sim \mathbf{S}_n$ is more than half. Thus, there exists some set $S \sim \mathbf{S}_n$ such that at least one third of the indices $i \in \{0,1\}^s$ are bad under S. [64] We now fix any such set S, and note that $|S| > B^{\mathtt{alg}}(n)$ (since $\mathbf{S}_n$ is of size more than $B^{\mathtt{alg}}$).

Let us now construct $T \subseteq \{0,1\}^m$ that is "over-sampled" by the sampler when given any input $x \in S$. To do so, let $\mathcal{B} \subseteq \{0,1\}^s$ be the set of bad indices under S. For any $i \in \mathcal{B}$, let $\Phi_i \subseteq [m]$ be the set of output bits of $\mathsf{Samp}^{(i)}{\restriction_S}$ that are constant; that is, $\Phi_i = \left\{ j \in [m] : \exists \sigma_j \in \{0,1\}, (\mathsf{Samp}^{(i)}{\restriction_S})_j \equiv \sigma_j \right\}$ and $|\Phi_i| \geq m/4$ (since i is bad under S). Also, let Q_i be the subcube of $\{0,1\}^m$ corresponding to the non-constant output

[63] For example, if $\widehat{\mathcal{C}}$ is indeed defined as all circuits of the form $\Phi\left(\{C_0(\mathsf{Samp}(x,i)\}_{i \in \{0,1\}^s}\right)$, where in particular C_0 and Φ can be any pair of dictator functions, then Samp is computable in $\widehat{\mathcal{C}}$.

[64] For any $i \in \{0,1\}^s$, denote by $\alpha_i(S)$ the random variable that is the number of output bits of $\mathsf{Samp}^{(i)}{\restriction_S}$ that are constant functions. Then, we have that $\Pr_{S \sim \mathbf{S}_n}[\alpha_i(S) \geq m/4] \geq 1/3$ (otherwise $\mathbb{E}_{S \sim \mathbf{S}_n}[\alpha_i(S)] < \frac{1}{3} \cdot m + \frac{2}{3} \cdot (m/4) = m/2$). Thus, the expected number of bad indices under $S \sim \mathbf{S}_n$ is at least $2^s/3$.

bits of $\mathsf{Samp}^{(i)}\restriction_S$; that is, $Q_i = \left\{z \in \{0,1\}^m : \forall j \in \Phi_i, z_j = (\mathsf{Samp}^{(i)}\restriction_S)_j\right\}$. We define T to be the union of the subcubes Q_i for all $i \in \mathcal{B}$; that is, $T = \bigcup_{i\in\mathcal{B}} Q_i$.

Note that $|T| < 2^m/5$, since for every $i \in \mathcal{B}$ it holds that $|Q_i| \leq 2^{3m/4}$ (and since we assumed that $2^s \cdot 2^{3m/4} < 2^m/5$). On the other hand, for every $x \in S$, the probability over $i \in \{0,1\}^s$ that $\mathsf{Samp}(x,i) \in T$ is lower bounded by the probability that $i \in \mathcal{B}$, which is at least $1/3$. Therefore, for any $x \in S$ it holds that $\Pr_{i\in\{0,1\}^s}[\mathsf{Samp}(x,i) \in T] \geq 1/3 > |T|/2^m + 1/10$. ∎

We mention that lemmas similar to Lemma 3.7.4 were proved for the specific setting of $\mathcal{AC}^0$ circuits (i.e., when $\mathcal{C} = \mathcal{AC}^0$) in [Vio05, Thm 6.4] and [GVW15, Thm 5.4], using a different proof strategy.[65] Relying on Lemma 3.7.4 and on the discussion that preceded the lemma's statement, we can now formally state our main theorem:

Theorem 3.7.5 (a limitation of two "black-box" techniques in quantified derandomization; Theorem 3.7.1, restated)**.** *Let $B^{\mathtt{red}} : \mathbb{N} \to \mathbb{N}$ and $B^{\mathtt{alg}} : \mathbb{N} \to \mathbb{N}$. For $n, s, m \in \mathbb{N}$ such that $2^{s+3m/4} \leq 2^m/5$, let $C : \{0,1\}^m \to \{0,1\}$, and let $\mathsf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ be a Boolean sampler with $B^{\mathtt{red}}$ bad inputs. Let $\widehat{\mathcal{C}}$ be any circuit class that can compute the function $C'(x) = MAJ\left(\{C(\mathsf{Samp}(x,i))\}_{i\in\{0,1\}^s}\right)$ and such that Samp is computable in $\widehat{\mathcal{C}}$. Then, for any distribution $\mathbf{S}_n$ of simplifier sets of size more than $B^{\mathtt{alg}}$ for $\widehat{\mathcal{C}}$ it holds that $B^{\mathtt{alg}}(n) < B^{\mathtt{red}}(n)$.*

We stress that Theorem 3.7.5 holds *regardless of the class $\mathcal{C}$* for which we wanted to solve the standard derandomization problem. This is the case since the limitation in Theorem 3.7.5 only relies on the hypothesis that Samp is computable in $\widehat{\mathcal{C}}$, and not on the fact that the circuit C' (which depends on C) is computable in $\widehat{\mathcal{C}}$.

Indeed, we did not use the fact that $\widehat{\mathcal{C}}$ contains a circuit (i.e., C') that computes a function that is "more complicated" than (the output bits of) Samp. Intuitively, we expect that "complicated" circuits will require simplifier sets that are smaller than simplifier sets for "simpler" circuits (e.g., Håstad's switching lemma [Hås87] yields simplifier sets of size $2^{\Omega(n/\log^{d-1}(n))}$ for circuits of depth d). In particular, in typical situations we expect that simplifier sets for $\widehat{\mathcal{C}}$ will need to be even smaller than simplifier sets for Samp. In such situations, the upper bound on $B^{\mathtt{alg}}(n)$ in Theorem 3.7.5 is not tight; that is, in such situations $B^{\mathtt{red}}$ and $B^{\mathtt{alg}}$ are far apart by significantly more than just a single bit (as is asserted in the theorem).

[65] Instead of relying directly on the existence of simplifier sets, they relied on the *low average sensitivity* of $\mathcal{AC}^0$ circuits; this property follows from a stronger notion of simplifier sets, which in particular are subcubes (i.e., it follows from Håstad's switching lemma; see [LMN93; Bop97; Tal17]). In comparison, our proof is simpler, and also applies to classes of functions with *high sensitivity* that have distributions of simplifier sets (recall that the simplifier sets in Definition 3.7.3 are not necessarily subcubes).

3.7.4 Strengthenings of the main theorem: Natural relaxations that do not suffice to bypass the limitation in Theorem 3.7.5

Following Theorem 3.7.5, the main question we are faced with is *which relaxations of the hypotheses* are sufficient to avoid the conclusion of the theorem, and thus to bypass the limitation arising from it. We now mention a few natural relaxations that *do not* seem sufficient to bypass this limitation.

3.7.4.1 Simplifier sets that simplify $\mathcal{C}$-circuits to non-constant functions

In Definition 3.7.3 we assumed that for every $C \in \mathcal{C}$, with high probability over $S \sim \mathbf{S}_n$ it holds that $C|_S$ is *constant*. We now note that if we assume that $C|_S$ simplifies to a "simple" (non-constant) function, then we can still obtain a corresponding quantified derandomization algorithm (with a mild loss in the parameters); but that in *some* natural settings, this relaxation does not suffice to bypass the limitations in Theorem 3.7.5.

Let us first see why this relaxation still suffices for quantified derandomization. Fix a circuit $C : \{0,1\}^n \to \{0,1\}$ with at most $B^{\mathtt{alg}}(n)$ exceptional inputs. Assume that there exists a distribution $\mathbf{S}_n$ over subsets $S \subseteq \{0,1\}^n$ of size at least $3 \cdot B^{\mathtt{alg}}(n)$ such that $\Pr_{S \sim \mathbf{S}_n}[C|_S \in \mathcal{C}_{Simple}] > 1/2$, where $\mathcal{C}_{Simple}$ is some class of "simple" functions. Further assume that an algorithm can efficiently sample a succint representation of $S \sim \mathbf{S}_n$ using few random bits, and that given a representation of S such that $C|_S \in \mathcal{C}_{Simple}$, the algorithm can efficiently approximate the acceptance probability of $C|_S$, up to error $1/10$. Note that for a strict majority of the choices of $S \sim \mathbf{S}_n$, the acceptance probability of $C|_S$ is either at most $1/3$ or at least $2/3$ (since $|S| > 3 \cdot B^{\mathtt{alg}}(n)$ for any $S \sim \mathbf{S}_n$), and we can distinguish between the two cases by estimating the acceptance probability of $C|_S \in \mathcal{C}_{Simple}$. Thus, a quantified derandomization algorithm can enumerate the choices of $S \sim \mathbf{S}_n$, decide for each choice whether the acceptance probability of $C|_S$ is at most $1/3$ or at least $2/3$, and rule according to a majority vote.[66]

Nevertheless, in some cases, this relaxation does not seem sufficient to bypass the limitation in Theorem 3.7.5. This is since for *some* natural classes $\mathcal{C}_{Simple}$ of "very simple" functions, a random restriction simplifies every $C \in \mathcal{C}_{Simple}$ to a constant function, with high probability; for example, this holds for constant-depth circuits [Hås87] and for linear threshold functions [KW16]. In these cases, the existence of $\mathbf{S}_n$ as above implies the existence of $\mathbf{S}'_n$ that meets the stronger definition (i.e., Definition 3.7.3), with a quantitative loss in the parameter $B^{\mathtt{alg}}$ that depends on $\mathcal{C}_{Simple}$ (i.e., the loss is induced by the random restriction that turns functions in $\mathcal{C}_{Simple}$ to constant functions).

3.7.4.2 A sampler that only samples $\mathcal{C}$-events

Our requirement from the sampler in Definition 3.7.2 was information-theoretic: For *any set* $T \subseteq \{0,1\}^m$, we required that for all but $B^{\mathtt{red}}$ inputs, the sampler will hit T

[66] The algorithm may not be able to correctly decide whether the acceptance probability of $C|_S$ is at most $1/3$ or at least $2/3$ when $C|_S \notin \mathcal{C}_{Simple}$, but the latter event only happens in the minority of the choices $S \sim \mathbf{S}_n$.

with an approximately correct probability (i.e., $\Pr_{i\in\{0,1\}^s}[\mathsf{Samp}(x,i) \in T] \in |T|/2^m \pm 1/10$). However, since we only want to use the sampler to approximate the acceptance probability of a circuit $C \in \mathcal{C}$, one may consider a relaxation in wish we only require that the sampler "appropriately samples" sets T that are decidable by $\mathcal{C}$ circuits. That is,

Definition 3.7.6 (sampler for $\mathcal{C}$-events)**.** *We say that a function* $\mathsf{Samp}_{\mathcal{C}} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ *is a* Boolean sampler for $\mathcal{C}$-events with $B^{\mathtt{red}}$ bad inputs *if it satisfies the following: For every* $T \subseteq \{0,1\}^m$ *such that* $T = C^{-1}(1)$ *for some* $C \in \mathcal{C}$*, for all but at most* $B^{\mathtt{red}}$ *of the inputs* $x \in \{0,1\}^n$ *it holds that* $\Pr_{i\in\{0,1\}^s}[\mathsf{Samp}(x,i) \in T] \in |T|/2^m \pm 1/10$.

The point that we wish to make is that in many natural settings, the relaxation in Definition 3.7.6 does not suffice to bypass the limitation in Theorem 3.7.5. This is the case because the "over-sampled" set $T \subseteq \{0,1\}^m$ that was constructed in the proof of Theorem 3.7.5 can be decided by a circuit that is a *DNF of size at most* 2^s. [67] (Using the notation of the proof, T is the union of $|\mathcal{B}| \leq 2^s$ subcubes $Q_1, ..., Q_{|\mathcal{B}|} \subseteq \{0,1\}^m$.) In particular, if the initial circuit C belongs to a class $\mathcal{C}$ that contains DNFs of size 2^s, then the relaxation in Definition 3.7.6 does not suffice to bypass the limitation in Theorem 3.7.5.

Detour. The notion of a sampler for $\mathcal{C}$-events might be of independent interest. The main point is that potentially, in some settings, the relaxation embodied in the definition of samplers for $\mathcal{C}$-events might allow to construct such samplers with better parameters than the parameters of information-theoretic samplers (i.e., as in Definition 3.7.2). For an example of a construction of a sampler for $\mathcal{AC}^0$-events, see [GW14, Thm 3.4 in the full version]: They constructed an $\mathcal{AC}^0$-computable sampler for $\mathcal{AC}^0$-events with $2^{n/\operatorname{poly}\log(n)}$ bad inputs and $m = n^{\Omega(1)}$.

We note, however, that this construction from [GW14] was later superseded by a construction of an $\mathcal{AC}^0$-computable sampler in the *information-theoretic* sense (i.e., as in Definition 3.7.2) that also has $2^{n/\operatorname{poly}\log(n)}$ bad inputs, and that has $m = n/\operatorname{poly}\log(n)$; see [CL16, Thms. 1.5 & 1.7]. Moreover, this construction, coupled with Lemma 3.7.4 and with Håstad's switching lemma [Hås87], implies that $\mathcal{AC}^0$-computable samplers for $\mathcal{AC}^0$-events *cannot* have significantly better parameters than $\mathcal{AC}^0$-computable samplers in the information-theoretic sense (i.e., as in Definition 3.7.2). This is the case since Håstad's switching lemma yields a distribution of simplifier sets of size $2^{\Omega(n/\log^{d-2}(n))}$ for depth-d circuits, and thus Lemma 3.7.4 implies that any sampler computable by depth-d circuits (even if it is a sampler only for DNF-events) must have at least $2^{\Omega(n/\log^{d-2}(n))}$ bad inputs;[68] and the number of bad inputs in the information-theoretic

[67] Recall that a typical setting of the parameters is $s = O(\log(n))$, and therefore the size of such a DNF is $2^s = \operatorname{poly}(n)$.

[68] The lower bound on the number of bad inputs for $\mathcal{AC}^0$-computable samplers for $\mathcal{AC}^0$-events can also be derived using the proofs of [Vio05; GVW15]: In their proofs, the "over-sampled" set can also be decided by a DNF of size that is at most exponential in the seed length of the extractor.

constructions in [CL16] is already $2^{\Omega(n/\log^{d-10}(n))}$ (or $2^{n/\log^{\Omega(d)}(n)}$, if one wishes to maximize the output length m), which nearly matches this lower-bound.

3.7.4.3 Samplers that are not efficiently computable

To combine the two techniques from Section 3.7.2 into a single algorithm, we need an efficient *uniform* algorithm that can compute $\mathsf{Samp}(x, i)$ for arbitrarily-chosen $x \in \{0,1\}^n$ and $i \in \{0,1\}^s$; that is, we need the sampler not only to be computable in the class $\mathcal{C}$, but also to be efficiently computable by a uniform algorithm.

We note, however, that the limitation in Theorem 3.7.5 holds even if we use a sampler that is not necessarily efficiently computable by a uniform algorithm. This is the case since the argument in Theorem 3.7.5 did not rely on such a hypothesis regarding the sampler, and thus holds also for "non-uniform" samplers.

Chapter 4

Derandomization and Lower Bounds

4.1 Introduction

Connections between the $prBPP = prP$ conjecture and lower bounds for algorithms and for circuits have been gradually developing for decades. Following the classical "hardness-to-randomness" paradigm [Yao82; BM84], a line-of-works initiated by Nisan and Wigderson [Nis91; NW94] proved that lower bounds for *non-uniform circuits* yield derandomization algorithms for $prBPP$. Moreover, this statement "scales smoothly", in the sense that stronger lower bounds yield quicker derandomization algorithms (see [Uma03] for an essentially optimal trade-off). At the extreme, if there is a function in $\mathcal{E}$ that cannot be computed by circuits of size $2^{\epsilon \cdot n}$ (for some constant $\epsilon > 0$) even infinitely-often, then $prBPP = prP$ (this was proved in [IW99]).

The other direction of the implication above has been gradually developing in the last two decades, in works showing that any derandomization of $prBPP$ (and even of $\mathcal{MA}$) implies circuit lower bounds that are currently unknown. The first work to imply such a result is [BFT98], and this was observed by Impagliazzo, Kabanets, and Wigderson [IKW02]. A subsequent line-of-works (including [IKW02] and following it) showed that even very *weak* derandomizations of $prBPP$ suffice to prove circuit lower bounds that are currently unknown (for the state-of-the-art results, following Murray and Williams [MW18], see Section 4.2). Nevertheless, the circuit lower bounds that we currently know to be *implied* by derandomizations of $prBPP$ are weaker than the lower bounds that would suffice (using the "hardness-to-randomness" results above) to deduce a deterministic derandomization of $prBPP$.

Alongside these works, a different line-of-works referred to as *uniform "hardness-to-randomness"* relates lower bounds for *uniform* probabilistic algorithms to *average-case* derandomization of $prBPP$ (for a precise definition of the latter term, see Section 4.3). This study was initiated by Impagliazzo and Wigderson [IW98], and for the state-of-the-art results, see Section 4.3. In comparison with the line-of-works described above (i.e., to non-uniform "hardness-to-randomness"), the known results that de-

duce derandomization from lower bounds incur significantly more overheads, and it is widely-believed that these results can be further improved.

In this thesis we include three contributions to the study of the connections between the $prBPP$ conjecture and lower bounds. The first contribution, described in Section 4.2, asserts that if $prBPP = prP$ then there exists a function that is computable in $NTIME[n^{\omega(1)}]$ (for essentially any super-constant function $\omega(1)$, e.g. $\log^*(n)$) that cannot be computed by polynomial-sized circuits. This result follows a recent breakthrough of Murray and Williams [MW18], and significantly improves on the previously-known lower bound (that referred to $NEXP$ rather than to $NTIME[n^{\omega(1)}]$); see Section 4.2 for details. This contribution is based on the work [Tel19b].

The second contribution, described in Section 4.3, significantly strengthens the average-case derandomization of BPP that is known to follow from lower bounds for uniform probabilistic algorithm (i.e., for $BPTIME$). We establish for the first time that average-case derandomization of BPP in nearly-polynomial-time (i.e., in time $n^{\mathrm{polyloglog}(n)} = 2^{\tilde{O}(\log(n))}$) follows from uniform hardness assumptions for $BPTIME$. Specifically, under the hypothesis that the Totally Quantified Boolean Formula (TQBF) problem cannot be solved by probabilistic algorithms that run in time $2^{n/\mathrm{polylog}(n)}$, we construct a pseudorandom generator for uniform circuits with *almost-exponential* stretch (i.e., with seed length $\tilde{O}(\log(n))$); see Section 4.3 for precise details. This contribution is based on a joint work with Lijie Chen, Ron Rothblum, and Eylon Yogev [CRT+19].

The third contribution, described in Section 4.4, refers to the question of whether any deterministic (worst-case) derandomization of $prBPP$ is *equivalent* to corresponding circuit lower bounds. A particular consequence of such an equivalence would be that derandomization of $prBPP$ necessitates constructing pseudorandom generators. Loosely speaking, we show that to answer this question in the affirmative, it suffices to prove a very weak version of the *non-deterministic exponential-time hypothesis*; and we also show that this very weak version is necessary to prove a slightly stronger conclusion that we deduce from it. Specifically, this very weak version asserts that there is a problem in E that cannot be solved by $NTIME[T]$-uniform circuits of size S, for relatively-small values of $S \ll T \ll 2^n$; see Section 4.4 for precise details. This contribution is also based on the aforementioned joint work with Lijie Chen, Ron Rothblum, and Eylon Yogev [CRT+19].

4.2 If $prBPP = prP$ then "almost NP" is not contained in P/poly

4.2.1 The main results

It has been known for at least two decades that the $prBPP = prP$ conjecture is intimately related to *circuit lower bounds*; that is, to lower bounds for non-uniform models of computation. Specifically:

- On the one hand, *any proof of sufficiently strong circuit lower bounds would also prove that* $prBPP = prP$. Specifically, if there is a function in $\mathcal{E}$ that requires exponential-sized circuits, then $prBPP = prP$ (see [IW99], which relies on the classical hardness-randomness paradigm [Yao82; BM84; NW94]).
- On the other hand, *any proof that* $prP = prBPP$ *implies long-sought circuit lower bounds*. As a prominent example, if $prBPP = prP$ then there exists a function in $NEXP$ that cannot be computed by any polynomial-sized circuit family [BFT98].[1] In fact, the latter circuit lower bound follows even from much weaker hypotheses, such as $MA \neq NEXP$ (see, e.g., [IKW02; Wil13]).

Informally, following a recent breakthrough by Murray and Williams [MW18], the main result in this section considerably strengthens the known connection between the conjecture that $prBPP = prP$ and circuit lower bounds.

The starting point for this result is the observation that an immediate corollary of a result from the recent work of Murray and Williams [MW18, Thm 1.2] is the following: *If* $prBPP = prP$*, then there exists a function in* $NTIME[n^{\text{poly}\log(n)}]$ (rather than $NEXP$) *that cannot be computed by any polynomial-sized circuit family*. This is a dramatic (almost exponential) strengthening of previously-known results (i.e., of [BFT98; IKW02]), and we believe that it is a fundamental result that is worth spelling out and highlighting. Furthermore, this result can even be further strengthened. In particular, by using the proof approach of [MW18] while instantiating their technical tools with different parameters, we get the following:

Theorem 4.2.1 (main theorem; informal)**.** *If* $prBPP = prP$*, then, for essentially any super-constant function* $f(n) = \omega(1)$*, there exists a set in* $NTIME[n^{f(n)}] \setminus P/\text{poly}$.

One might a-priori hope to strengthen the conclusion of Theorem 4.2.1 by improving the time bound in the non-deterministic class; that is, to prove that "if $prBPP = prP$, then $NP \not\subseteq P/\text{poly}$ (and $P \neq NP$)". However, such a result cannot be proved without *unconditionally* proving that $P \neq NP$, since any proof of the conditional statement "$prBPP = prP \implies P \neq NP$" would unconditionally imply that $P \neq NP$ (see Proposition 4.2.5). Therefore, the conclusion of Theorem 4.2.1 is optimal in this sense.

Theorem 4.2.1 is a special case of a more general "derandomization implies lower bounds" result that follows using the technical tools of Murray and Williams [MW18], and in particular their new "easy witness lemma" (see Section 4.2.2.1 for details on the latter). In this general result, the lower bound in the conclusion can be parameterized:

Theorem 4.2.2 (a generalized version of Theorem 4.2.1; informal, see Corollary 4.2.11)**.** *There exists* $\epsilon > 0$ *such that for any time-computable* $s : \mathbb{N} \to \mathbb{N}$ *satisfying* $n < s(n) < 2^{\epsilon \cdot n}$ *it holds that*

$$prBPP = prP \implies NTIME[s' \circ s' \circ s'] \not\subseteq SIZE[s] \,,$$

where $s' = \text{poly}(s(O(n)))$.

[1] In [BFT98] it is shown, unconditionally, that $MAEXP \not\subseteq P/\text{poly}$. Thus, under the hypothesis $prBPP \subseteq prNP$ we have that $MAEXP = NEXP \not\subseteq P/\text{poly}$ (see [IKW02, Rmk. 26]).

Indeed, Theorem 4.2.1 follows as a special case of Theorem 4.2.2 by using $s(n) = n^{\omega(1)}$ (in which case $s' \circ s' \circ s' = n^{\omega(1)}$ and $\mathcal{SIZE}[s] \supset \mathcal{P}/\mathrm{poly}$; see Corollary 4.2.13). The hypothesis of Theorem 4.2.2 can also be significantly relaxed, since its proof relies on Williams' [Wil13] celebrated proof strategy, which is well-known to support such relaxations. Recall that, loosely speaking, Williams' proof strategy shows that certain "non-trivial" circuit-analysis algorithms imply circuit lower bounds. In our case, Theorem 4.2.2 holds, for example, under the hypothesis that there exists a "non-trivial" algorithm for the Circuit Acceptance Probability Problem (CAPP) (i.e., an algorithm that approximates the acceptance probability of a circuit of size m with v variables in time $2^{.99 \cdot v} \cdot \mathrm{poly}(m)$); Theorem 4.2.2 also holds under the hypothesis that $pr\text{-}co\mathcal{RP} \subseteq pr\mathcal{NP}$. See the end of Section 4.2.3.1 for details of possible relaxations.

Theorem 4.2.2 may be compared to the following result of Kinne, van Melkebeek, and Shaltiel [KMS12, Thm. 9], which builds on the well-known result of Kabanets and Impagliazzo [KI04]: If $\mathcal{BPP} = \mathcal{P}$ (i.e., the "non-promise" version of the $\mathcal{BPP} = \mathcal{P}$ conjecture holds), then either the *permanent function* of $\{0,1\}$-matrices over $\mathbb{Z}$ does not have polynomial-sized arithmetic circuits, or for essentially any $s : \mathbb{N} \to \mathbb{N}$ it holds that $\mathcal{NTIME}[s^{O(1)}] \not\subseteq \mathcal{SIZE}[s]$. Indeed, in this result from [KMS12] the hypothesis is weaker than in Theorem 4.2.2, since it only refers to the "non-promise" conjecture $\mathcal{BPP} = \mathcal{P}$ (rather than to $pr\mathcal{BPP} = pr\mathcal{P}$); whereas the conclusion is not a circuit lower bound, but rather a disjunction of two circuit lower bounds.[2] Nevertheless, in the lower bound $\mathcal{NTIME}[s^{O(1)}] \not\subseteq \mathcal{SIZE}[s]$, the time-complexity of the "hard" function is just $s^{O(1)}$, rather than $s^{O(1)} \circ s^{O(1)} \circ s^{O(1)}$ as in Theorem 4.2.2.

4.2.1.1 A technical strengthening of Theorem 4.2.2

Observe that the lower bound in the conclusion of Theorem 4.2.2 becomes trivial when s is half-exponential or larger (i.e., when $s(s(n)) \geq 2^n$), because there are three compositions of s in the time-bound for the "hard" function.[3] The following improvement removes this limitation: We prove that if $pr\mathcal{BPP} = pr\mathcal{P}$, then there exists a function in $\mathcal{NTIME}[s^{O(1)} \circ s^{O(1)}]$ that cannot be computed by circuits of size s.

Moreover, we also improve the concluded lower bound by showing that size-s circuits fail to compute the "hard" function on a "dense" set of input lengths (the conclusion in Theorem 4.2.2 only guarantees failure on infinitely-many input lengths). Specifically, for $s_I(n) = \mathrm{poly}(s(\mathrm{poly}(n)))$, we conclude that size-$s$ circuits fail to compute the "hard" function on an input length in any interval of the form $[n, s_I(n)]$.

[2] Similarly to Theorem 4.2.2, the result in [KMS12] also follows from the weaker hypothesis $co\mathcal{RP} \subseteq \mathcal{NP}$. Also, Jansen and Santhanam [JS12] showed how to remove the disjunction in the conclusion, at the cost of deducing a weaker lower bound: Loosely speaking, they showed that if $\mathcal{BPP} = \mathcal{P}$, then polynomial-sized arithmetic circuits cannot compute the class of polynomials such that each bit in the binary representation of the output of the polynomial can be computed in $\mathcal{NEXP} \cap co\mathcal{NEXP}$ (see [JS12] for precise details).

[3] Specifically, when $s(s(n)) \geq 2^n$ we have that $\mathcal{NTIME}[s^{O(1)} \circ s^{O(1)} \circ s^{O(1)}] \supseteq \mathcal{NTIME}[2^{\mathrm{poly}(s(n))}]$, whereas $\mathcal{DTIME}[2^{\mathrm{poly}(s(n))}] \not\subseteq \mathcal{SIZE}[s]$ holds unconditionally (by a diagonalization argument).

Similarly to Theorem 4.2.2, the foregoing (stronger) conclusions follow also from the hypothesis $pr\text{-}co\mathcal{RP} \subseteq pr\mathcal{NP}$ (which is weaker than $pr\mathcal{BPP} = pr\mathcal{P}$):

Theorem 4.2.3 (strengthening the conclusion of Theorem 4.2.2; informal, see Theorem 4.2.19)**.** *There exists $\epsilon > 0$ such that for any time-computable $s : \mathbb{N} \to \mathbb{N}$ satisfying $n < s(n) < 2^{\epsilon \cdot n}$ it holds that*

$$pr\text{-}co\mathcal{RP} \subseteq pr\mathcal{NP} \implies \mathcal{NTIME}[s' \circ s'] \not\subseteq \mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s] \ ,$$

where $s' = \mathrm{poly}(s)$, and $\mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s]$ is the class of problems such that there exists a size-s circuit that, for infinitely-many intervals of length $s_I(n) = \mathrm{poly}(s(n^2))$, solves the problem on some input length in the interval.

The proof of Theorem 4.2.3 does not follow the proof approach of Murray and Williams, and in particular does not use their new "easy witness lemma". Nevertheless, the proof crucially relies on one of their technical results, namely their strengthening of Santhanam's circuit lower bound [San09]. See Section 4.2.2.2 for further details.

In Appendix 4.2.5.1 we present an alternative and relatively simple proof of a weaker form of Theorem 4.2.3, which does not include the guarantee of failure in every "small" interval. This proof does not use the results of Murray and Williams, but is based only on (a generalization of) the well-known circuit lower bound of Santhanam [San09]. The proof strategy for this result was suggested to us by Igor Oliveira after a preliminary version of the work [Tel19b] appeared online, and also serves as the proof strategy for the improved results in Section 4.2.2.2.

4.2.1.2 The meaning of the results in this section

What is the meaning of the statement "$pr\mathcal{BPP} = pr\mathcal{P} \implies \mathcal{NTIME}[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$"? Note that the lower bound $\mathcal{NTIME}[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$ asserts that polynomial-sized circuits cannot simulate *both "slightly" super-polynomial running time and non-determinism*.[4]

Thus, on the one hand, one may view the lower bound $\mathcal{NTIME}[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$ as a weaker form of $\mathcal{NP} \not\subseteq \mathcal{P}/\mathrm{poly}$. Hence, Theorem 4.2.1 can be interpreted as saying that proving that $pr\mathcal{BPP} = pr\mathcal{P}$ is as hard as proving a lower bound that is essentially a precursor of $\mathcal{NP} \subseteq \mathcal{P}/\mathrm{poly}$. From this perspective the conclusion of the theorem is essentially optimal, since (as mentioned after the statement of Theorem 4.2.1), we cannot strengthen the conclusion of the theorem to $\mathcal{NP} \not\subseteq \mathcal{P}/\mathrm{poly}$ without unconditionally proving that $\mathcal{P} \neq \mathcal{NP}$.

On the other hand, as pointed out by Ryan Williams, one can alternatively view the statement $\mathcal{NTIME}[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$ as a weaker form of the statement $\mathcal{DTIME}[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$. The latter statement asserts that polynomial-sized circuits cannot simulate algorithms with superpolynomial running time. From this perspective, Theorem 4.2.1

[4] This lower bound can be viewed as a significant strengthening of the (unconditionally-known) lower bound $\Sigma_3[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$, which asserts that polynomial-sized circuits cannot simulate both super-polynomial running time and "several levels" of non-determinism/alterations. (The proof of the lower bound is a diagonalization argument a-la Kannan's theorem; see, e.g., [Juk12, Lem. 20.12].)

implies that proving that $prBPP = prP$ is as hard as proving a weak form of a "strengthened time-hierarchy theorem" (in which we compare uniform algorithms to non-uniform circuits).

Continuing the latter view, it seems instructive to compare the lower bounds implied by $prBPP = prP$ to the lower bounds that are known to *imply* $prBPP = prP$ (using the results of Impagliazzo and Wigderson [IW99]). Specifically, being slightly informal,[5] we have that:

$$\forall s(n) < 2^{\epsilon \cdot n},\ \mathcal{DTIME}[\mathrm{poly}(s)] \not\subseteq \mathtt{i.o.}\text{-}\mathcal{SIZE}[s] \tag{4.2.1}$$

$$\Downarrow \quad \text{(by [IW99])}$$

$$prBPP = prP$$

$$\Downarrow \quad \text{(by Thm 4.2.3)}$$

$$\forall s(n) < 2^{\epsilon \cdot n},\ \mathcal{NTIME}[s' \circ s'] \not\subseteq \mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s] \tag{4.2.2}$$

where $\epsilon > 0$, s', and s_I are defined as in Theorem 4.2.3. This comparative perspective suggests the following interpretation of the results in this section:

> The lower bounds implied by $prBPP = prP$ are now significantly stronger (compared to the lower bounds that were previously known to be implied by this conjecture); but they are nevertheless still weaker than the lower bounds that are known to *imply* that $prBPP = prP$.

Is $prBPP = prP$ equivalent to a specific circuit lower bound? The circuit lower bounds implied in Eq. (4.2.2) hold not only when $prBPP = prP$, but also under the (intuitively) weaker hypothesis $pr\text{-}coRP \subseteq prNP$. Therefore, one might suspect that the conclusion in Theorem 4.2.2 can be strengthened. Recall that the question of whether specific derandomization results are *equivalent* to *specific* circuit lower bounds has been raised several times in the past (see, e.g., [IKW02] and [TV07, Sec. 1.1]). We thus propose the following natural conjecture (we have found no explicit prior mentions of this conjecture in the literature):

Conjecture 4.2.4 ($prBPP = prP$ is equivalent to the [IW99] lower bounds)**.** *The statement that $prBPP = prP$ is equivalent to the statement that for some $\epsilon > 0$ and every $s(n) < 2^{\epsilon \cdot n}$ it holds that $\mathcal{DTIME}[\mathrm{poly}(s)] \not\subseteq \mathtt{i.o.}\text{-}\mathcal{SIZE}[s]$.*

The most important gap between Theorem 4.2.3 and Conjecture 4.2.4 is that in Theorem 4.2.3, the lower bounds implied by $prBPP = prP$ are against *non-deterministic* classes. Note that even a modest first step towards proving Conjecture 4.2.4, namely proving that $prBPP = prP \implies \mathcal{EXP} \not\subseteq P/\mathrm{poly}$, already implies that any polynomial-time derandomization of $prBPP$ requires pseudorandom generators (see [BFN+93]).

[5] The informality is by ignoring time-computability constraints on s.

4.2.1.3 Organization

In Section 4.2.2 we present high-level overviews of the proofs of our main theorems. In Section 2 we present preliminary definitions. In Section 4.2.3 we prove Theorems 4.2.1 and 4.2.2, and in Section 4.2.4 we prove Theorem 4.2.3.

4.2.2 Overviews of the proofs

In Section 4.2.2.1 we present an overview of the proof of Theorem 4.2.2, and in Section 4.2.2.2 we present an overview of the proof of Theorem 4.2.3. Since the proof approaches for the two theorems are very different, one may read Section 4.2.2.2 without first reading Section 4.2.2.1.

4.2.2.1 Proof overview for Theorem 4.2.2

The proof of Theorem 4.2.2 follows the approach used by Murray and Williams [MW18], which is based on the celebrated proof strategy of Williams [Wil13]. The main new component in [MW18] is a new "easy witness lemma", which allows for flexible scaling of the parameters in the original proof strategy of Williams (see below; this new lemma improves the original easy witness lemma of [IKW02]). Murray and Williams stated consequences with two *specific parameter settings*. We extend their result by stating a *general* (parametrized) "derandomization implies lower bounds" result that uses this proof approach with the new easy witness lemma (see Theorem 4.2.10), and deduce Theorems 4.2.1 and 4.2.2 as special cases.

Let us now overview the proof of Theorem 4.2.2. The point of the overview is to describe how the (well-known) proof strategy of Williams can be instantiated with the new easy witness lemma for general parameters in order to deduce Theorem 4.2.2. The starting point for the proof is the Circuit Acceptance Probability Problem (or CAPP, in short): Given as input the description of a Boolean circuit C, the problem is to distinguish between the case that the acceptance probability of C is at least $2/3$ and the case that the acceptance probability of C is at most $1/3$. It is well-known that a deterministic polynomial-time algorithm for CAPP exists if and only if $prBPP = prP$ (see Proposition 2.4.2). The current argument relies on the much weaker hypothesis that CAPP for circuits of size m with v input variables can be solved in time $2^{.99 \cdot v} \cdot \mathrm{poly}(m)$; for simplicity, let us assume that the CAPP algorithm runs in time $2^{.99 \cdot v} \cdot m^2$.

Fix any time-computable function $n < s(n) < 2^{\epsilon \cdot n}$, where $\epsilon > 0$ is a universal constant. Denoting $t = s^{O(1)} \circ s^{O(1)} \circ s^{O(1)}$, our goal is to prove that $\mathcal{NTIME}[t] \not\subset \mathcal{SIZE}[s]$. (The definition of t in this high-level overview is slightly informal; see Definition 4.2.8 and Corollary 4.2.11 for precise details.) To do so, assume towards a contradiction that $\mathcal{NTIME}[t] \subseteq \mathcal{SIZE}[s]$, and let $t_0(n) = t(n)^\delta$, where $\delta > 0$ is sufficiently small. We will construct, for any $L \in \mathcal{NTIME}[t_0]$, a non-deterministic machine that decides L in time $t_0^{1-\Omega(1)}$; this will contradict the non-deterministic time hierarchy [Coo72].

The new easy witness lemma asserts that if $\mathcal{NTIME}[t] \subseteq \mathcal{SIZE}[s]$ where $t = t_0^{1/\delta}$, then for every $L' \in \mathcal{NTIME}[(t_0)^2]$, every $(t_0)^2$-time verifier V for L' and every $x \in L'$, there exists a circuit $P_x \in \mathcal{SIZE}[t_0^{.001}]$ that encodes a witness π_x such that $V(x, \pi_x)$ accepts.[6] (Again, our parameters in the overview are informal; see Lemma 4.2.9 for a statement that uses precise parameters.) The point is that witnesses for the verifier V are a-priori of size $(t_0)^2$, but the lemma asserts that (under the hypothesis) every $x \in L'$ has a witness that can be concisely represented by a circuit of much smaller size $t_0^{.001}$. We note that the main "bottleneck" in the proof that requires using $t = s^{O(1)} \circ s^{O(1)} \circ s^{O(1)}$ (rather than, say, $t = \mathrm{poly}(s)$) is the new "easy witness lemma".

Let us now construct the non-deterministic machine for $L \in \mathcal{NTIME}[t_0]$, relying on the existence of the foregoing "compressible" witnesses. We first fix a PCP system for L with a verifier V that runs in time $t_V = \mathrm{poly}(n, \log(t_0))$ and uses $\ell = \log(t_0) + O(\log\log(t_0))$ random bits. (For concreteness, we use the PCP of Ben-Sasson and Viola [BSV14], but previous ones such as [BGH+05] also suffice for the proof.) Using the new easy witness lemma, for every $x \in L$ there exists a circuit of size $t_0(|x|)^{.001}$ that encodes a valid proof for x in this PCP system.[7]

Now, given input $x \in \{0,1\}^n$, the non-deterministic machine M first guesses a circuit P_x of size $t_0(n)^{.001}$, in the hope that such a circuit encodes a valid proof for x. Then, the machine constructs a circuit $C_x^{P_x}$ that, when given $r \in \{0,1\}^\ell$ as input, simulates the execution of V on x using randomness r when V is given oracle access to the witness P_x (i.e., $C_x^{P_x}(r) = V^{P_x}(x, r)$). Finally, the machine M uses the CAPP algorithm on the circuit $C_x^{P_x}$ to determine whether the verifier is accepts x with high probability over r or rejects x with high probability over r.

Note that if $x \in L$, then for *some* guess of P_x it holds that $C_x^{P_x}$ has acceptance probability one, and thus the machine M will accept x. On the other hand, if $x \notin L$, then for *any* guess of P_x it holds that $C_x^{P_x}$ has low acceptance probability (corresponding to the soundness of the PCP verifier), and thus the machine M will reject x.

The point is that all the operations of the machine happened in time much shorter than $t_0(n)$. Specifically, the size of P_x is $t_0(n)^{.001}$, and the size of $C_x^{P_x}$ is $m < t_V(n) \cdot t_0(n)^{.001} < t_0(n)^{.002}$; thus, guessing P_x and constructing $C_x^{P_x}$ can be done in time $\mathrm{poly}(m) \ll \sqrt{t_0(n)}$. Now, note that $C_x^{P_x}$ has $\ell = \log(t_0) + O(\log\log(t_0))$ variables; thus, when the CAPP algorithm is given $C_x^{P_x}$ it runs in time

$$2^{.99 \cdot \ell} \cdot m^2 < t_0(n)^{.995} \cdot \left(t_0(n)^{.002}\right)^2 = (t_0(n))^{1-\Omega(1)} ,$$

and we get a contradiction.

[6] A circuit $P_x : \{0,1\}^{\log(|\pi_x|)} \to \{0,1\}$ encodes a string π_x if for every $i \in [|\pi_x|]$ it holds that $P_x(i)$ is the i^{th} bit of π_x (equivalently, π_x is the truth-table of P_x).

[7] To apply the easy witness lemma, consider the deterministic verifier V' that, when given input and a proof, enumerates the random coins of V and decides by a majority vote. This verifier runs in time $2^\ell \cdot t_V < (t_0)^2$, so we can apply the lemma to L with this verifier.

As mentioned in the introduction, the hypothesis in this proof strategy can be further relaxed in various (known) ways. For details of these relaxations, see the statement of Theorem 4.2.10 and the remark following the theorem's proof.

4.2.2.2 Proof overview for Theorem 4.2.3

The proof of Theorem 4.2.3 is very different than the proof of Theorem 4.2.2, and in particular does *not* rely on the proof strategy of Williams [Wil13] or on an "easy witness lemma". As a first step, let us prove a statement that is weaker than that of Theorem 4.2.3: We prove that if $pr\text{-}co\mathcal{RP} \subseteq pr\mathcal{NP}$, then for essentially any $s : \mathbb{N} \to \mathbb{N}$ it holds that $\mathcal{NTIME}[s \circ s] \not\subseteq \mathcal{SIZE}[s]$ (without claiming that failure happens in every "small" interval).

Recall that a standard approach to prove "derandomization implies lower bounds" theorems is to rely on unconditional lower bounds for $\mathcal{MA}$ protocols: Specifically, if we start from a lower bound $\mathcal{MATIME}[t] \not\subseteq \mathcal{SIZE}[s]$, for some $t > s$, and assume a derandomization hypothesis $\mathcal{MATIME}[t] \subseteq \mathcal{NTIME}[t']$, for some $t' \geq t$, then under the derandomization hypothesis we have that $\mathcal{NTIME}[t'] \not\subseteq \mathcal{SIZE}[s]$. In our case, we can rely on a generalization of the lower bounds of Santhanam [San09] for $\mathcal{MA}$ protocols with non-uniform advice: For $t \approx s \circ s$ and a small function $\ell = O(\log(s))$ it holds that $\mathcal{MATIME}[t]/\ell \not\subseteq \mathcal{SIZE}[s]$ (see Appendix 4.2.5.1 for a proof of this statement).[8] Now, our derandomization hypothesis implies that $\mathcal{MATIME}[t]/\ell \subseteq \mathcal{NTIME}[t']/\ell$, where $t' = \text{poly}(t)$, and thus we can conclude that $\mathcal{NTIME}[t']/\ell \not\subseteq \mathcal{SIZE}[s]$.

The second observation is that if $\mathcal{NTIME}[\text{poly}(t)]/\ell \not\subseteq \mathcal{SIZE}[s]$, then $\mathcal{NTIME}[\text{poly}(t)]$ (without the advice) is not contained in $\mathcal{SIZE}[s']$, for s' that is moderately smaller than s. To see this, assume towards a contradiction that $\mathcal{NTIME}[\text{poly}(t)] \subseteq \mathcal{SIZE}[s']$. For any $S \in \mathcal{NTIME}[\text{poly}(t)]/\ell$ we construct a family of size-s circuits that decides S. Consider a non-deterministic machine M that decides S with advice $\{a_n\}$, and let S^{adv} be the set of pairs (x, σ) such that $|\sigma| = \ell(|x|)$ and M (non-deterministically) accepts x when given advice σ. Note that S^{adv} can be decided by a non-determinstic machine that simulates M (and requires no advice), and thus, by our hypothesis, S^{adv} can be solved by a circuit family $\{C_n\}$ of size s'. By hard-wiring the "good" advice a_n into each C_n, we obtain a circuit family $\{C'_n\}$ of size s' that decides S. Note that the size of the circuit is still s', but it is now a function of a smaller input length, since we "hard-wired" the advice in place of input bits; however, since the advice is relatively short (i.e., $\ell = O(\log(s))$), the new size function, denoted s, is not much larger than s' (see Proposition 4.2.18). The crucial point is that the foregoing "advice elimination" argument only follows through *after the derandomization step* (i.e., for $\mathcal{NTIME}$ and not for $\mathcal{MATIME}$). This is because when dealing with probabilistic machines, it is not clear how to define S^{adv} in a way that will allow a probabilistic machine without

[8] In fact, this lower bound can be improved by reducing the advice length ℓ from logarithmic to a single bit. But since our proof of Theorem 4.2.3 (rather than the weaker version) will later use the lower bound of [MW18], in which $\ell = O(\log(s))$, let us assume at this point already that ℓ is logarithmic.

advice to decide it (since a probabilistic machine that is given a "wrong" advice might not "distinctly" accept or reject some inputs).

This proves the weaker version of Theorem 4.2.3, which does not assert that every circuit of size-s' fails to compute the hard function in any "small" interval. To prove the stronger version, our starting point is the strengthening by Murray and Williams [MW18] of the lower bound from [San09] for MA protocols with advice: They showed unconditionally that there exists $S \in \mathcal{MATIME}[t]/\ell$ (where $\ell = O(\log(s))$ as above) such that $S \notin \mathtt{i.o.}_{[\mathrm{poly}(s)]}\text{-}\mathcal{SIZE}[s]$ (see Theorem 4.2.15).

Going through the proof above, note that the first step (i.e., the derandomization step) preserves the failure of small circuits in every "small" interval; and thus we need to show that the second step (i.e., the "advice elimination" argument) also preserves this property. The source of trouble is that now our "towards-a-contradiction" hypothesis only implies that $S^{\mathtt{adv}} \in \mathtt{i.o.}_{[\mathrm{poly}(s)]}\text{-}\mathcal{SIZE}[s]$, which only guarantees the existence of an infinite "dense" set $I \subseteq \mathbb{N}$ of input lengths for which $S^{\mathtt{adv}}$ has small circuits. In particular, we have no guarantee that every $n \in I$ is of the form $m + \ell(m)$, which is what we need to deduce that $S_m = S \cap \{0,1\}^m$ has small circuits. To overcome this problem, we "embed" all pairs (x, σ) such that $|\sigma| = \ell(|x|)$ and $|x| + |\sigma| < n$ into $\{0,1\}^n$, and define $S^{\mathtt{adv}}_n = S^{\mathtt{adv}} \cap \{0,1\}^n$ such that deciding $S^{\mathtt{adv}}_n$ allows to determine the output of M on (x, σ) for all pairs satisfying $|x| + |\sigma| < n$. Thus, for any $n \in I$, a circuit of size $s(n)$ that decides $S^{\mathtt{adv}}_n$ allows us to solve S_m where $m + \ell(m) < n$. And similarly to above, since the advice is relatively small (i.e., $\ell(m) = O(\log(s(m))) < m$), both the size $s(n)$ of the circuit and the interval length $\mathrm{poly}(s(n))$ in which failure is guaranteed are not too large as a function of m. For precise details see Proposition 4.2.18.

4.2.2.3 A barrier for proving "$prBPP = prP \implies P \neq NP$"

We note that it is impossible to prove the statement "if $prP = prBPP$ then $P \neq NP$" without *unconditionally* proving that $P \neq NP$.

Proposition 4.2.5 (a barrier for "derandomization implies lower bounds" statements)**.** *If the conditional statement "$prBPP = prP \implies P \neq NP$" holds, then $P \neq NP$.*

Proof. Assume towards a contradiction that $P = NP$. Then, the polynomial-time hierarchy collapses to P, and similarly the promise-problem version of the polynomial-time hierarchy collapses to prP.[9] Now, since $prBPP$ is contained in the promise-problem version of the polynomial-time hierarchy (e.g., by adapting the well-known argument of Lautemann [Lau83]), it follows that $prBPP = prP$. Finally, we can use

[9] To see that this is the case, let $\Pi = (Y, N) \subseteq \{0,1\}^* \times \{0,1\}^*$ be a promise problem in $pr\Sigma_k$, for some $k \in \mathbb{N}$. Then, there exists a polynomial-time algorithm A such that for every $x \in Y$ it holds that $\exists y_1, \forall y_2, ..., y_k : A(x, y_1, ..., y_k) = 1$, and for every $x \in N$ it does not hold that $\exists y_1, \forall y_2, ..., y_k : A(x, y_1, ..., y_k) = 1$. We define a set $S = S_A$ that consists of all strings x such that $\exists y_1, \forall y_2, ..., y_k : A(x, y_1, ..., y_k) = 1$. Note that $S \supseteq Y$, and that $S \cap N = \emptyset$, and that $S \in \Sigma_k$ (using the algorithm A). By our assumption that the polynomial-time hierarchy collapses, there exists a polynomial-time algorithm A' that decides S. It follows that A' solves the problem Π.

the hypothesized conditional statement to deduce that $\mathcal{P} \neq \mathcal{NP}$, which is a contradiction. ■

4.2.3 Proof of Theorems 4.2.1 and 4.2.2

We will first prove a general and parametrized "derandomization implies lower bounds" theorem. This theorem is obtained by using the proof strategy of Williams [Wil13] with general parameters, while leveraging the new easy witness lemma of Murray and Williams [MW18]. We then prove Theorems 4.2.1 and 4.2.2 as corollaries.

In the proofs we will use the new "easy witness lemma" of Murray and Williams [MW18]. Towards stating it, let us recall the definition of witness circuits for a proof system.

Definition 4.2.6 (verifiers and witnesses)**.** *Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible, non-decreasing function, and let $L \subseteq \{0,1\}^*$. An algorithm $V(x,y)$ is a t-*time verifier *for L if V runs in time at most $t(|x|)$ and satisfies the following: For all strings x it holds that $x \in L$ if and only if there exists* a witness *y such that $V(x,y)$ accepts.*

Definition 4.2.7 (witness circuits)**.** *Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible, non-decreasing function, let $w : \mathbb{N} \to \mathbb{N}$, and let $L \subseteq \{0,1\}^*$. We say that a t-time verifier V has* witness circuits of size *w if for every $x \in L$ there exists a witness y_x such that $V(x,y_x)$ accepts and there exists a circuit $C_{y_x} : \{0,1\}^{\log(|y_x|)} \to \{0,1\}$ of size $w(|x|)$ such that $C_{y_x}(i)$ is the i^{th} bit of y_x. We say that $\mathcal{NTIME}[t]$* has witness circuits of size *w if for every $L \in \mathcal{NTIME}[t]$, every t-time verifier for L has witness circuits of size w.*

Loosely speaking, the new easy witness lemma of [MW18] asserts that for any two functions $t(n) \gg s(n)$ with sufficient "gap" between them, if $\mathcal{NTIME}[\mathrm{poly}(t)] \subseteq \mathcal{SIZE}[s]$, then $\mathcal{NTIME}[t]$ has witness circuits of size $\hat{s}$, where $\hat{s}(n) > s(n)$ is the function s with some "overhead". To more conveniently account for the exact parameters, we introduce some auxiliary technical notation:

Definition 4.2.8 (sufficiently gapped functions)**.** *Let $\gamma, \gamma', \gamma'' \in \mathbb{N}$ be universal constants.*[10] *For any function $s : \mathbb{N} \to \mathbb{N}$, let $s' : \mathbb{N} \to \mathbb{N}$ be the function $s'(n) = (s(\gamma \cdot n))^{\gamma}$, and let $\hat{s} : \mathbb{N} \to \mathbb{N}$ be the function $\hat{s}(n) = (s'(s'(s'(n))))^{\gamma'}$. We say that two functions $s, t : \mathbb{N} \to \mathbb{N}$ are* sufficiently gapped *if both functions are increasing and time-constructible, and s' is also time-constructible, and $s(n) < 2^{n/\gamma}/n$, and $t(n) \geq (\hat{s}(n))^{\gamma''}$.*

Lemma 4.2.9 (easy witnesses for small nondeterministic time [MW17, Lem 4.1])**.** *Let $s, t : \mathbb{N} \to \mathbb{N}$ be sufficiently gapped functions, and assume that $\mathcal{NTIME}[O(t(n))^{\gamma}] \subset \mathcal{SIZE}[s]$, where γ is the constant from Definition 4.2.8. Then, $\mathcal{NTIME}[t]$ has witness circuits of size $\hat{s}$.*

[10] Specifically, the values of these constants are $\gamma = e$ and $\gamma' = 2g$ and $\gamma'' = d$, where e, g, and d are the universal constants from Lemma 4.1 in [MW17].

4.2.3.1 A parametrized "derandomization implies lower bounds" theorem

Loosely speaking, in the following theorem statement we assume that CAPP can be solved in non-deterministic time $T(m,v)$, and deduce that for any two functions $t(n) \gg s(n)$ such that $T\Big(\text{poly}(n,\hat{s}(n),\log(t(n))),\ \log(t(n))\Big) \ll t(n)$ it holds that $\mathcal{NTIME}[\text{poly}(t(n))]$ does not have circuits of size $s(n)$.

Theorem 4.2.10 (derandomization implies lower bounds, with general parameters)**.** *There exist constants $c, c' \in \mathbb{N}$ and $\alpha < 1$ such that the following holds. For $T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, assume that $(1, 1/3)$-CAPP on circuits of size m with at most v input variables can be solved in non-deterministic time $T(m,v)$. Let $s, t : \mathbb{N} \to \mathbb{N}$ be sufficiently gapped functions such that $s(n) > n$ and for some constant $\epsilon > 0$ and any constant $\alpha > 0$ it holds that*

$$T\Big((n \cdot \hat{s}(n) \cdot \log(t(n)))^c,\ \ \alpha \cdot \log(t(n))\Big) \leq t(n)^{(1-\epsilon)\cdot\alpha}\ ,$$

where $\hat{s}$ is defined as in Definition 4.2.8. Then, $\mathcal{NTIME}[t(n)^{c'}] \not\subseteq \mathcal{SIZE}[s(n)]$.

Proof. The starting point of the proof is the non-deterministic time hierarchy [Coo72]: For an appropriate function $t' = t'(n)$ (that will be determined in a moment), there exists a set $L \in \mathcal{NTIME}[t']$ that cannot be decided by non-deterministic machines running in time $(t')^{1-\Omega(1)}$. Specifically, for a sufficiently small constant $\alpha > 0$, let $t'(n) = (t(n))^{(1-\epsilon/2)\cdot\alpha}$, and let $L \in \mathcal{NTIME}[t'] \setminus \mathcal{NTIME}\left[(t')^{\frac{1-\epsilon}{1-\epsilon/2}}\right]$.[11] Now, for a sufficiently large constant c', assume towards a contradiction that $\mathcal{NTIME}[t(n)^{c'}] \subseteq \mathcal{SIZE}[s(n)]$. Our goal is to construct a non-deterministic machine that decides L in time $(t')^{\frac{1-\epsilon}{1-\epsilon/2}}$, which will yield a contradiction.

To do so, consider the PCP verifier of [BSV14] for L, denoted by V. On inputs of length n, the verifier V runs in time $\text{poly}(n, \log(t'(n)))$, uses $\ell = \log(t'(n)) + O(\log\log(t'(n)))$ bits of randomness, and has perfect completeness and soundness (much) lower than $1/3$. [12] Furthermore, using the hypothesis that $\mathcal{NTIME}[t(n)^{c'}] \subseteq \mathcal{SIZE}[s(n)]$ and the "easy witness lemma" (i.e., Lemma 4.2.9), for every $x \in L$ there exists a circuit $P_x \in \mathcal{SIZE}[\hat{s}(n)]$ such that $\Pr_r[V^{P_x}(x,r) \text{ accepts}] = 1$. (We actually apply Lemma 4.2.9 to the deterministic verifier V' that enumerates the random coins of V, which runs in time $2^\ell \cdot \text{poly}(n, \log(t')) = \text{poly}(t') = \text{poly}(t)$. We can use the lemma since we assumed that $\mathcal{NTIME}[t(n)^{c'}] \subseteq \mathcal{SIZE}[s(n)]$, for a sufficiently large c'.)

Given input $x \in \{0,1\}^n$, the non-deterministic machine M acts as follows. The machine non-deterministically guesses a (description of a) circuit P_x of size $\hat{s}(n)$, and constructs a circuit $C_x^{P_x} : \{0,1\}^\ell \to \{0,1\}$ such that $C_x^{P_x}(r) = V^{P_x}(x,r)$. Then, the machine feeds the description of $C_x^{P_x}$ as input to the machine M_{CAPP} that solves CAPP

[11] Such a function exists by standard non-deterministic time hierarchy theorems (e.g., [Coo72]), since $t'(n) > n^{\Omega(1)}$, which implies that the gap between t' and $(t')^{1-\Omega(1)}$ is sufficiently large.

[12] Note that the only upper-bound that we need on the number of oracle queries issued by V is the trivial bound given by the running time of V.

in non-deterministic time T and exists by the hypothesis, and outputs the decision of $M_{\texttt{CAPP}}$. By the properties of the PCP verifier and of $M_{\texttt{CAPP}}$, if $x \in L$ then for some guess of P_x and for some non-deterministic choices of $M_{\texttt{CAPP}}$, the machine M will accept x; and if $x \notin L$, then for any guess of P_x and any non-deterministic choices of $M_{\texttt{CAPP}}$, the machine M will reject x.

To conclude let us upper-bound the running-time of the machine M. The circuit $C_x^{P_x}$ has $\ell = \log(t') + O(\log\log(t')) < \alpha \cdot \log(t)$ input bits, and its size is $m(n) = \text{poly}(n, \log(t')) \cdot \hat{s}(n)$; thus, its representation size is $\text{poly}(m(n))$. Therefore, the circuit $C_x^{P_x}$ can be constructed in time $\text{poly}(m(n))$, and the CAPP algorithm runs in time $T(m(n), \ell)$. The total running-time of the non-deterministic machine M is thus at most $T\left((n \cdot \hat{s}(n) \cdot \log(t(n)))^c, \alpha \cdot \log(t)\right)$, for some constant c. By our hypothesized upper-bound on T, the running time of M is at most $t(n)^{(1-\epsilon)\cdot\alpha} = (t')^{\frac{1-\epsilon}{1-\epsilon/2}}$, which yields a contradiction. ■

Additional relaxations of the hypothesis in Theorem 4.2.10. Since the proof of Theorem 4.2.10 relies on the strategy of [Wil13], it is well-known that the hypothesis of the theorem can be further relaxed. First, we do not have to *unconditionally* assume that the non-deterministic machine for CAPP exists, and it suffices to assume that the machine exists *under the hypothesis that* $\mathcal{NTIME}[t(n)^{c'}] \subseteq \mathcal{SIZE}[s(n)]$ (this is the case since we are only using the existence of the machine to contradict the latter hypothesis). And secondly, the non-deterministic machine that solves CAPP can use (sub-linearly many) bits of non-uniform advice; this follows by using a strengthened non-deterministic time hierarchy theorem, which was proved by Fortnow and Santhanam [FS16] (see [MW18, Remark 1] for details).

4.2.3.2 Theorems 4.2.1 and 4.2.2 as corollaries

We now prove Theorem 4.2.2 as a corollary of Theorem 4.2.10. As detailed in Section 4.2.2.1, we start from the hypothesis that $(1, 1/3)$-CAPP can be solved in non-deterministic time $T(m, v) = 2^{.99 \cdot v} \cdot \text{poly}(m)$ (which is weaker than the hypothesis $pr\mathcal{BPP} = pr\mathcal{P}$). The proof amounts to verifying that, given such a CAPP algorithm, the hypothesis of Theorem 4.2.10 holds for essentially any s and $t \approx s^{O(1)} \circ s^{O(1)} \circ s^{O(1)}$.

Corollary 4.2.11 (Theorem 4.2.2, restated)**.** *Assume that* $(1, 1/3)$-CAPP *can be solved in non-deterministic time* $T(m, v) \leq 2^{(1-\epsilon)\cdot v} \cdot \text{poly}(m)$*, for some constant* $\epsilon > 0$*. Then, there exists a constant* $k \in \mathbb{N}$ *such that for any two sufficiently gapped functions* $s : \mathbb{N} \to \mathbb{N}$ *and* $t : \mathbb{N} \to \mathbb{N}$ *it holds that* $\mathcal{NTIME}[t(n)^k] \not\subseteq \mathcal{SIZE}[s]$*.*

Proof. Let $k' > 1$ be such that $T(m, v) \leq 2^{(1-\epsilon)\cdot v} \cdot m^{k'}$. We invoke Theorem 4.2.10 with the sufficiently gapped functions s and $t_1(n) = t(n)^{k''}$, where $k'' > 1$ is a sufficiently

large constant that depends on k'. Note that for any $\alpha > 0$ it holds that

$$\begin{aligned} T\Big((n \cdot \hat{s}(n) \cdot \log(t_1(n)))^c,\ \ \alpha \cdot \log(t_1(n))\Big) & \\ &\leq (n \cdot \log(t_1(n)))^{c \cdot k'} \cdot (t_1(n))^{\epsilon/2} \cdot t_1(n)^{(1-\epsilon)\cdot\alpha} && (\hat{s}(n)^{c\cdot k'} < t_1(n)^{\epsilon/2}) \\ &\leq (t_1(n))^{1-\epsilon/3}\,, && (n^{c\cdot k'} < s(n)^{c\cdot k'} < t_1(n)^{\epsilon/12}) \end{aligned}$$

where both inequalities relied on the hypothesis that k'' is sufficiently large. Thus, we conclude that $\mathcal{NTIME}[t_2] \not\subseteq \mathcal{SIZE}[s]$, where $t_2(n) = t_1(n)^{c'} = t(n)^{c' \cdot k''}$. ■

Finally, we prove Theorem 4.2.1 as a corollary of Corollary 4.2.11. Recall that the conclusion in Theorem 4.2.1 is that $\mathcal{NTIME}[n^{f(n)}] \not\subseteq \mathcal{P}/\text{poly}$ for "essentially" any super-constant function f. We now specify exactly what this means. Our goal is to deduce that $\mathcal{NTIME}[n^{f(n)}] \not\subseteq \mathcal{SIZE}[n^{g(n)}]$, where $g(n) \ll f(n)$ and $g(n) = \omega(1)$. Therefore, the proof works for any f such that a suitable g exists. We note in advance that this minor technical detail imposes no meaningful restrictions on f (see next).

Definition 4.2.12 (admissible functions)**.** *We say that a function $f : \mathbb{N} \to \mathbb{N}$ is* admissible *if f is super-constant (i.e. $f(n) = \omega(1)$), and if there exists another super-constant function $g : \mathbb{N} \to \mathbb{N}$ that satisfies the following: The function g is super-constant, and $t(n) = n^{f(n)}$ and $s(n) = n^{g(n)}$ are sufficiently gapped, and $\hat{s}(n) = n^{o(f(n))}$.*

Essentially any increasing function $f(n) = \omega(1)$ such that $f(n) \leq n$ is admissible, where the only additional constraints that the admissibility condition imposes are time-constructibility of various auxiliary functions (we require t and s to be sufficiently gapped, which enforces time-constructibility constraints); for a precise (and tedious) discussion, see Appendix 4.2.5.2. We can now formally state Theorem 4.2.1 and prove it:

Corollary 4.2.13 (Theorem 4.2.1, restated)**.** *Assume that $(1, 1/3)$-CAPP can be solved in non-deterministic time $T(m, v) \leq 2^{(1-\epsilon)\cdot v} \cdot \text{poly}(m)$, for some constant $\epsilon > 0$. Then, for every admissible function f there exists a set in $\mathcal{NTIME}[n^{O(f(n))}] \setminus \mathcal{P}/\text{poly}$.*

Proof. Since f is admissible, there exists a function g that satisfies the requirements of Definition 4.2.12. We thus invoke Corollary 4.2.11 with the functions $t(n) = n^{f(n)}$ and $s(n) = n^{g(n)}$, and conclude that there exists a set in $\mathcal{NTIME}[n^{O(f(n))}] \setminus \mathcal{SIZE}[n^{g(n)}]$. Since $g(n) = \omega(1)$, the latter set does not belong to $\mathcal{P}/\text{poly}$. ■

4.2.4 Proof of Theorem 4.2.3

In this section we prove Theorem 4.2.3. Throughout the section, for a set $S \subseteq \{0,1\}^*$ and $n \in \mathbb{N}$, we denote $S_n = S \cap \{0,1\}^n$.

Recall that the conclusion in Theorem 4.2.3 is that there exists a set S such that for every polynomial-sized circuit family and sufficiently large $n \in \mathbb{N}$, the family fails to decide S on some input length in the interval $[n, s_I(n)]$. Our actual conclusion will be

slightly stronger: We will conclude that for every sufficiently large $n \in \mathbb{N}$, the circuit family fails to decide S on at least one of the "end-points" of the interval; that is, it fails either on input length n, or on input length $s_I(n)$ (or on both).

This leads us to the following definition. Intuitively, the following definition asserts that $S \in \mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s]$ if there exists a circuit family of size s that, on infinitely-many input lengths $n \in \mathbb{N}$, manages to decide S correctly both on inputs of length n and on inputs of length $s_I(n)$. Indeed, it follows that if $S \notin \mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s]$, then every circuit family $\{C_n\}$ of size s that tries to decide S fails, for every sufficiently large $n \in \mathbb{N}$, either on inputs on size n or on inputs of size $s_I(n)$ (or on both).

Definition 4.2.14 (a stronger notion of infinitely-often computation)**.** *For $s, s_I : \mathbb{N} \to \mathbb{N}$ and $S \subseteq \{0,1\}^*$, we say that $S \in \mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s]$ if there exists an infinite set $\mathcal{N} \subseteq \mathbb{N}$ and a circuit family $\{C_n\}_{n\in\mathbb{N}}$ of size at most s such that for every $n \in \mathcal{N}$, it holds that:*

1. *The circuit $C_n : \{0,1\}^n \to \{0,1\}$ computes S_n.*
2. *The circuit $C_{s_I(n)} : \{0,1\}^{s_I(n)} \to \{0,1\}$ computes $S_{s_I(n)}$.*

Definition 4.2.14 is reminiscent of the notion of robust simulations by Fortnow and Santhanam [FS17], but the two definitions have significant differences. Recall that a set $S \subseteq \{0,1\}^*$ can be *robustly simulated* by a circuit family if for every polynomial p there are infinitely-many integers n such that the family correctly decides S on all input lengths in the interval $[n, p(n)]$. In contrast, in Definition 4.2.14 we consider a *fixed* interval length s_I (which may also be super-polynomial), but only require the circuit to decide S on the end-points of the interval.

The starting point of the proof of Theorem 4.2.3 is Murray and Williams' [MW17, Thm 3.1] strengthening of Santhanam's [San09] circuit lower bound. Following [MW18], we say that a function $s : \mathbb{N} \to \mathbb{N}$ is a circuit-size function if s is increasing, time-constructible, and for all sufficiently large $n \in \mathbb{N}$ it holds that $s(n) < 2^n/(2n)$.

Theorem 4.2.15 (Murray and Williams' [T]hm 3.1)**.** *mw17 strengthening of Santhanam's [San09] lower bound] Let s be a super-linear circuit-size function, and let $t = \mathrm{poly}(s(\mathrm{poly}(s)))$ (for sufficiently large polynomials that do not depend on s). Then, there exists a set $S \in \mathcal{MATIME}_0[t]/O(\log(s))$ such that $S \notin \mathtt{i.o.}_{[\mathrm{poly}(s)]}\text{-}\mathcal{SIZE}[s]$.*

Note that in [MW18] the "hard" set S is stated to be in $\mathcal{MATIME}[t]/O(\log(s))$, rather than in $\mathcal{MATIME}_0[t]/O(\log(s))$ (i.e., the verifier in [MW18] has two-sided error). However, using Theorem 2.3.5, we can assume that $S \in \mathcal{MATIME}_0[t]/O(\log(s))$, where $t = \mathrm{poly}(s(\mathrm{poly}(s)))$.

As mentioned in Section 4.2.2.2, the first observation in the proof is that if $pr\text{-}co\mathcal{RP} \subseteq pr\mathcal{NP}$ then we can derandomize $\mathcal{MA}$ verifiers that receive non-uniform advice.

Proposition 4.2.16 (derandomization of $\mathcal{MA}$ with advice)**.** *If $pr\text{-}co\mathcal{RP} \subseteq pr\mathcal{NP}$, then for any $t, \ell : \mathbb{N} \to \mathbb{N}$ such that t is time-constructible it holds that $\mathcal{MATIME}[t]/\ell \subseteq \mathcal{NTIME}[\mathrm{poly}(t)]/\ell$.*

Proof. Note that $(1,1/3)$-CAPP is in $pr\text{-}co\mathcal{RP}$. Thus, relying on the hypothesis that $pr\text{-}co\mathcal{RP} \subseteq pr\mathcal{NP}$, there exists a non-deterministic polynomial-time machine M_{CAPP} that gets as input a Boolean circuit C and satisfies the following: If the acceptance probability of C is one, then for some non-deterministic choices M_{CAPP} accepts; and if the acceptance probability of C is at most $1/3$ then M_{CAPP} rejects, regardless of the non-deterministic choices.

Now, let S be a set in $\mathcal{MATIME}_0[t]/\ell$, let V be an $\mathcal{MATIME}_0[t]/\ell$ verifier for S, and let $\{a_n\}_{n\in\mathbb{N}}$ be a sequence of "good" advice that allows V to decide S. We want to construct a non-deterministic machine M that runs in time $\mathrm{poly}(t)$ and decides S with ℓ bits of non-uniform advice. Given input $x \in \{0,1\}^n$ and advice a_n, the machine M guesses a witness $w \in \{0,1\}^{t(n)}$, and constructs a circuit $C = C_{V,x,w,a_n} : \{0,1\}^{t(n)} \to \{0,1\}$ that gets as input $r \in \{0,1\}^{t(n)}$ and computes $V(x,w,r,a_n)$. Then, the machine M feeds C to the machine M_{CAPP}, and outputs the decision of M_{CAPP}. The running time of the machine M is dominated by the running time of M_{CAPP}, which is at most $\mathrm{poly}(t(n))$. Now, since a_n is the "good" advice for V, if $x \in S$ then there exists w such that the acceptance probability of C is one, which means that there exist non-deterministic choices for M_{CAPP} such that M_{CAPP} will accept C; on the other hand, if $x \notin S$ then for any w the acceptance probability of C is at most $1/3$, which means that for any non-deterministic choices for M_{CAPP} it holds that M_{CAPP} rejects C. ■

The second observation in the proof is that if $\mathcal{NTIME}[t]/\ell$ is not contained in a non-uniform class of circuits, then $\mathcal{NTIME}[O(t)]$ (i.e., without non-uniform advice) is also not contained in a (related) non-uniform class of circuits. Moreover, this assertion still holds if the "separation" between the classes is in the sense of Definition 4.2.14.

We first prove a simpler form of this statement, which showcases the main idea but is much less cumbersome. In the following statement, we only consider a single bit of advice, and do not refer to separations in the sense of Definition 4.2.14.

Proposition 4.2.17 (eliminating the advice)**.** *Let $s_0, s, t : \mathbb{N} \to \mathbb{N}$ such that t is increasing, and for all sufficiently large $n \in \mathbb{N}$ it holds that $s_0(n) \geq s(n+1)$. If $\mathcal{NTIME}[t]/1 \not\subseteq \mathcal{SIZE}[s_0]$, then $\mathcal{NTIME}[O(t)] \not\subseteq \mathcal{SIZE}[s]$.*

Proof. We prove the contrapositive statement: If $\mathcal{NTIME}[O(t)] \subseteq \mathcal{SIZE}[s]$, then $\mathcal{NTIME}[t]/1 \subseteq \mathcal{SIZE}[s_0]$. To do so, fix any $S \in \mathcal{NTIME}[t]/1$, and let us construct a circuit family of size s_0 that decides S.

To construct the circuit family we consider an auxiliary set S^{adv}, which is defined as follows. Let M be a t-time non-deterministic machine and let $\{a_n\}$ be a sequence of advice bits such that M correctly decides S when given advice $\{a_n\}$. Let S^{adv} be the set of pairs (x,σ), where $x \in \{0,1\}^*$ and $\sigma \in \{0,1\}$, such that M (non-deterministically) accepts x when given advice σ. Note that $S^{\text{adv}} \in \mathcal{NTIME}[O(t)]$, because a non-deterministic machine that gets input (x,σ) simulate the machine M on input x with advice σ and decide according to the output of M.

Relying on the hypothesis that $\mathcal{NTIME}[O(t)] \subseteq \mathcal{SIZE}[s]$, there exists a circuit family $\{C_n\}$ of size s such that each C_n decides S^{adv}_n. By hard-wiring the "correct"

advice bit a_n in place of the last input bit into every C_n, we obtain a circuit family $\{C'_n\}$ such that each C'_n decides S_n, and its size is at most $s(n+1) \leq s_0(n)$. ■

The following proposition is a stronger form of Proposition 4.2.17, which considers possibly long advice strings, and refers to separations in the sense of Definition 4.2.14.

Proposition 4.2.18 (eliminating the advice). *Let $s_0, s, \ell, t, s_I : \mathbb{N} \to \mathbb{N}$ be functions such that t is super-linear and increasing, and s_I, s_0 and s are increasing, and the mapping $1^n \mapsto 1^{\ell(n)}$ is computable in time $O(n + \ell(n))$. Assume that for every sufficiently large $n \in \mathbb{N}$ it holds that $\ell(n) < n/2$ and $s_0(n) \geq s(2n)$ and $s_0(s_I(n)) \geq s(2s_I(2n))$. Further assume that $\mathcal{NTIME}[t]/\ell \not\subseteq \mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s_0]$. Then, $\mathcal{NTIME}[O(t)] \not\subseteq \mathtt{i.o.}_{[2s_I]}\text{-}\mathcal{SIZE}[s]$.*

We comment that a statement that is more general than the one in Proposition 4.2.18 can be proved, foregoing some of the requirements (e.g., on ℓ) while allowing potential degradation in the parameters of the conclusion. Since the statement of Proposition 4.2.18 suffices for our parameter setting, and for simplicity, we avoid such generalizations.

Proof of Proposition 4.2.18. Assuming that $\mathcal{NTIME}[O(t)] \subseteq \mathtt{i.o.}_{[2s_I]}\text{-}\mathcal{SIZE}[s]$, we prove that $\mathcal{NTIME}[t]/\ell \subseteq \mathtt{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s_0]$. Fixing any $S \in \mathcal{NTIME}[t]/\ell$, let us construct a circuit family of size s_0 that decides S infinitely-often on inputs of length n and $s_I(n)$.

We first define a set $S^{\mathtt{adv}}$ as follows. Let M be a t-time non-deterministic machine and let $\{a_n\}$ be a sequence of "good" advice strings of length $|a_n| = \ell(n)$ such that M correctly decides S when given advice $\{a_n\}$. For every $n \in \mathbb{N}$, the set $S^{\mathtt{adv}}_n$ will include representations of all pairs (x, σ), where $|\sigma| = \ell(|x|)$ and $|x| + 2|\sigma| < n$, such that M accepts x when given advice σ. Specifically, we define $S^{\mathtt{adv}}_n$ to be the set of all n-bit strings of the form $1^t 0^{|\sigma|} 1 x \sigma$, where $t = n - (|x| + 2|\sigma| + 1)$, such that M accepts x when given advice σ.[13]

Note that $S^{\mathtt{adv}} \in \mathcal{NTIME}[O(t)]$. This is the case since a non-deterministic machine that gets input $z \in \{0,1\}^n$ can first verify that z can be parsed as $z = 1^t 0^{|\sigma|} 1 x \sigma$ such that $|\sigma| = \ell(|x|)$ (and reject z if the parsing fails); and then simulate the machine M on input x with advice σ, in time $O(t(|x|)) = O(t(n))$, and decide according to the output of M. Now, since we assume that $\mathcal{NTIME}[O(t)] \subseteq \mathtt{i.o.}_{[2s_I]}\text{-}\mathcal{SIZE}[s]$, there exists an infinite set $I \subseteq \mathbb{N}$ and a circuit family $\{C_n\}$ of size s such that for every $n \in I$:

1. $C_n : \{0,1\}^n \to \{0,1\}$ correctly computes $S^{\mathtt{adv}}_n$; and
2. $C_{2s_I(n)} : \{0,1\}^{2s_I(n)} \to \{0,1\}$ correctly computes $S^{\mathtt{adv}}_{2s_I(n)}$.

We transform $\{C_n\}$ into a circuit family of size s_0 that decides S infinitely-often on inputs of length both n and $s_I(n)$. To do so, we rely on the following simple claim:

Claim 4.2.18.1. *Let $n, m \in \mathbb{N}$ such that $m + 2\ell(m) < n$. Assume that there exists a circuit of size $s(n)$ that decides $S^{\mathtt{adv}}_n$. Then, there exists a circuit of size $s(n)$ that decides S_m.*

[13] The $0^{|\sigma|}$ term facilitates the parsing of the suffix of the n-bit string as a pair $x\sigma$.

Proof. Let C_n be the circuit of size $s(n)$ for S^{adv}_n. The circuit C_m for S_m is obtained by hard-wiring into C_n the "correct" advice a_m instead of the last $\ell(m)$ input bits, and the correct initial padding $1^{n-m-2\ell(m)-1}0^{\ell(m)}1$ instead of the first $n-m-\ell(m)$ input bits. □

For every $n \in I$, let $m = m(n)$ be the largest integer such that $m + 2\ell(m) + 1 \leq n$. Let $I' = \{m(n)\}_{n\in\mathbb{N}}$, and note that I' is infinite. For every sufficiently large $m \in I'$, relying on the fact that $m = m(n)$ for some $n \in I$ and on Claim 4.2.18.1, we have that:

1. There exists a circuit $C_m : \{0,1\}^m \to \{0,1\}$ of size $s(n) \leq s_0(\lceil n/2 \rceil) \leq s_0(m)$ that decides S_m. (We relied on the fact that $m \geq n/2$, since $\ell(m) < m/2$.)
2. There exists a circuit $C_{s_I(m)} : \{0,1\}^{s_I(m)} \to \{0,1\}$ of size $s_0(s_I(m))$ that decides $S_{s_I(m)}$. To see this, recall that there exists a circuit $C_{2s_I(n)}$ of size $s(2s_I(n))$ that decides $S^{\text{adv}}_{2s_I(n)}$. We can invoke Claim 4.2.18.1 because $s_I(m) + 2\ell(s_I(m)) < 2s_I(m) < 2s_I(n)$. Also, relying on the fact that $m \geq n/2$ and on the hypotheses regarding s_0, s and s_I, we have that $s(2s_I(n)) \leq s(2s_I(2m)) \leq s_0(s_I(m))$.

It follows that $S \in \text{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s_0]$. ■

We now combine the foregoing ingredients into a proof of Theorem 4.2.3.

Theorem 4.2.19 (Theorem 4.2.3, restated)**.** *There exists a constant $\epsilon > 0$ such that the following holds.*

- *Let $s : \mathbb{N} \to \mathbb{N}$ be an increasing, super-linear and time-constructible function such that for all sufficiently large $n \in \mathbb{N}$ it holds that $s(n) \leq 2^{\epsilon \cdot n}$ and that $s(2n) \leq s(n)^2$.*
- *Let $t = \text{poly}(s(\text{poly}(s)))$, for sufficiently large polynomials (that do not depend on s).*
- *Let $s_0 : \mathbb{N} \to \mathbb{N}$ be an increasing and time-constructible function such that for all sufficiently large $n \in \mathbb{N}$ it holds that $s_0(n) \geq s(n^2)^2$ and that $s_0(2n) \leq s_0(n)^2$.*

Assume that $prBPP \subseteq prNP$. Then, $NTIME[t] \not\subseteq \text{i.o.}_{[\text{poly}(s_0)]}\text{-}\mathcal{SIZE}[s]$.

Note that if the function s in Theorem 4.2.19 satisfies $s(n^2) < s(n)^k$, for a sufficiently large constant $k \in \mathbb{N}$, then we can use the function $s_0(n) = s(n)^{2k}$, and deduce that $NTIME[t] \not\subseteq \text{i.o.}_{[\text{poly}(s)]}\text{-}\mathcal{SIZE}[s]$.

Proof of Theorem 4.2.19. Let $t_0 = \text{poly}(s_0(\text{poly}(s_0)))$, for sufficiently large polynomials, and let $\ell = O(\log(s_0))$ (the universal constant hidden in the O-notation is the one from Theorem 4.2.15). By Theorem 4.2.15, there exists $S \in \mathcal{MATIME}[t_0]/\ell \setminus \text{i.o.}_{[(s_0)^c]}\text{-}\mathcal{SIZE}[s_0]$, for a sufficiently large constant $c \in \mathbb{N}$. By Proposition 4.2.16, and relying on the hypothesis that $prBPP \subseteq prNP$, it holds that $S \in \mathcal{NTIME}[\text{poly}(t_0)]/\ell \setminus \text{i.o.}_{[(s_0)^c]}\text{-}\mathcal{SIZE}[s_0]$.

We now want to use Proposition 4.2.18 to deduce that $\mathcal{NTIME}[\text{poly}(t_0)]$ is not contained in $\text{i.o.}_{[\text{poly}(s_0)]}\text{-}\mathcal{SIZE}[s]$, and thus we need to verify that the functions ℓ, s,

s_0, and $(s_0)^c$ satisfy the hypothesis of Proposition 4.2.18. This is indeed the case since for all sufficiently large $n \in \mathbb{N}$ it holds that $\ell(n) < n/2$ (assuming that ϵ is sufficiently small); and since $s_0(n) > s(n^2)^2 \geq s(2n)$, and $s(2s_0(2n)^c) \leq s(s_0(n)^{2c})^2 \leq s_0(s_0(n)^c)$. ∎

As mentioned before the statement of Theorem 4.2.19, the "gap" between the input lengths n and $s_I(n) = \text{poly}(s_0(n))$ (on which any size-s circuit family is guaranteed to fail) in Theorem 4.2.19 is larger than the function s that bounds the size of the circuits. This is no coincidence: If the gap function s_I would have been *significantly smaller* than the bound s on the circuit size, then we would have obtained an "almost-everywhere" lower bound (for circuits of size about $s(s_I^{-1})$).[14]

4.2.5 Appendices for Section 4.2

4.2.5.1 An alternative proof of Theorem 4.2.2

In this section we present an alternative proof of Theorem 4.2.1, which does not rely on the work of Murray and Williams [MW18], but rather on the work of Santhanam [San09]. The idea for this alternative proof was suggested to us by Igor Oliveira.

The structure of this alternative proof is very similar to the proof of Theorem 4.2.3 (which was described in Section 4.2.2.2), but uses as a starting point a generalization of the circuit lower bound proved by Santhanam [San09], instead of its subsequent strengthening by Murray and Williams [MW18]. Specifically, the starting point of the proof is the following:

Theorem 4.2.20 (a generalization of [San09, Thm. 1])**.** *Let $s : \mathbb{N} \to \mathbb{N}$ be an increasing, super-linear and time-computable function such that for all sufficiently large $n \in \mathbb{N}$ it holds that $s(3n) \leq s(n)^3$. Then, for $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) = \text{poly}(s(\text{poly}(s(n))))$ it holds that $\mathcal{MATIME}[t]/1 \not\subseteq \mathcal{SIZE}[s]$.*

The proof of Theorem 4.2.20 imitates the original argument from [San09], but uses more general parameters. We include the full proof for completeness, but since it requires no new significant ideas, we defer its presentation to the end of the appendix. The alternative proof of Theorem 4.2.1 follows by combining Theorem 4.2.20, Proposition 4.2.16 (instantiated with the value $\ell = 1$), and Proposition 4.2.17.

Theorem 4.2.21 (Theorem 4.2.1, an alternative technical statement)**.** *Let $s : \mathbb{N} \to \mathbb{N}$ be an increasing, super-linear and time-computable function such that for all sufficiently large $n \in$*

[14] To see this, assume that $S \notin \text{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s]$, for $s_I \ll s$. We define a set S^{emb} by "embedding" all strings in S of length $n-1$ and $s_I^{-1}(n-1)$ into $\{0,1\}^n$: For each $n \in \mathbb{N}$, let S_n^{emb} consist of all n-bit strings $0^{n-|x|}1x$ such that $x \in S$. Since $S \notin \text{i.o.}_{[s_I]}\text{-}\mathcal{SIZE}[s]$, for every sufficiently large $n \in \mathbb{N}$ the circuit complexity of S_n^{emb} is larger either than $s(n-1)$ or than $s(s_I^{-1}(n-1))$. In natural cases where $s(s_I^{-1}(n-1)) < s(n-1)$, we obtain an "almost-everywhere" lower bound for circuits of size about $s(s_I^{-1})$.

$\mathbb{N}$ it holds that $s(3n) \leq s(n)^3$, and let $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) = \mathrm{poly}(s(\mathrm{poly}(s(n))))$, for sufficiently large polynomials. Assume that $prBPP = prP$. Then, $NTIME[t] \not\subseteq SIZE[s]$.

Proof. Let $s_0 = s^3$, and let $t_0 = \mathrm{poly}(s_0(\mathrm{poly}(s_0)))$, for sufficiently large polynomials. According to Theorem 4.2.20, there exists a set S in $MATIME[t_0]/1$ such that $S \notin SIZE[s_0]$. By Proposition 4.2.16, and relying on the hypothesis that $prBPP = prP$, it holds that $S \in NTIME[t_1]/1 \setminus SIZE[s_0]$, where $t_1 = \mathrm{poly}(t_0)$. Using Proposition 4.2.17, it holds that $NTIME[t] \not\subseteq SIZE[s_1]$, where $t = O(t_1) = \mathrm{poly}(s(\mathrm{poly}(s)))$ and $s_1(n) = s_0(n-1)$. Finally, since s is increasing and $s(n) \leq s(\lceil n/3 \rceil)^3$, we have that $s_1(n) = s_0(n-1) \geq s_0(\lceil n/3 \rceil) \geq s(n)$, and hence $NTIME[t] \not\subseteq SIZE[s]$. ■

It is just left to detail the proof of Theorem 4.2.20. The first technical ingredient in the proof is the $PSPACE$-complete set of Trevisan and Vadhan [TV07]. We use this set, but instead of relying on the fact that the set is $PSPACE$-complete, we will use padding to claim that the set is complete for $DSPACE[n^{\omega(1)}]$ under $n^{\omega(1)}$-time reductions.

Lemma 4.2.22 (scaling the $PSPACE$-complete set of [TV07])**.** *There exists a set $L^{\mathrm{TV}} \subseteq \{0,1\}^*$ and a probabilistic polynomial-time oracle Turing machine M that satisfy the following:*

1. *Let $t : \mathbb{N} \to \mathbb{N}$ be a super-linear, time-computable function. Then, for every set $L \in DSPACE[t]$ there exists a deterministic Turing machine R_L that runs in time $\mathrm{poly}(t)$ such that for every $x \in \{0,1\}^*$ it holds that $x \in L \iff R_L(x) \in L^{\mathrm{TV}}$.*
2. *On input $x \in \{0,1\}^*$, the machine M only issues queries of length $|x|$.*
3. *For any $x \in L^{\mathrm{TV}}$ it holds that $\Pr[M^{\mathbf{1}_{L^{\mathrm{TV}}}}(x) = 1] = 1$, where $\mathbf{1}_{L^{\mathrm{TV}}} : \{0,1\}^n \to \{0,1\}$ is the indicator function of $L^{\mathrm{TV}} \cap \{0,1\}^n$.*
4. *For any $x \notin L^{\mathrm{TV}}$ and any $f : \{0,1\}^n \to \{0,1\}$ it holds that $\Pr[M^f(x) = 0] \geq 2/3$.*

Proof. We take L^{TV} to be the $PSPACE$-complete set from [San09, Lem. 12], which is the same set constructed in [TV07]. Items (2) – (4) follow immediately from the original statement in [San09].[15] Item (1) follows since L^{TV} is $PSPACE$-complete, and using a padding argument. Specifically, for any t and L, consider the machine R_L that combines a reduction of L to $L' = \{(x, 1^t) : x \in L\}$ with a reduction of L' to L^{TV}. The first reduction maps $x \mapsto (x, 1^t)$, and since $L' \in PSPACE$, there exists a second reduction of L' to L^{TV} that can be computed in time $\mathrm{poly}(t + |x|) < \mathrm{poly}(t)$ (the inequality is since t is super-linear). ■

Proof of Theorem 4.2.20. Let $t_0 : \mathbb{N} \to \mathbb{N}$ such that $t_0(n) = s^4(n)$, and let $t_1 = \mathrm{poly}(t_0)$ and $t = t_2 = \mathrm{poly}(t_0(\mathrm{poly}(t_0)))$, for sufficiently large polynomials. Let L^{TV} be the set from Lemma 4.2.22. Our goal is to prove that there exists a set in $MATIME[t_2]/1$ that is not in $SIZE[t_0^{1/4}]$. The proof proceeds by a case analysis.

[15] The original statement asserts that any $x \notin L^{\mathrm{TV}}$ is rejected with probability at least $1/2$ (rather than $2/3$), but this probability can be amplified to $2/3$ using standard error-reduction.

Case 1: $L^{\mathtt{TV}} \in \mathcal{SIZE}[t_0]$**.** By a standard diagonalization argument, there exists a set $L^{\mathtt{diag}} \in \mathcal{DSPACE}[t_1] \setminus \mathcal{SIZE}[t_0]$. [16] Our main goal now will be to prove that $\mathcal{DSPACE}[t_1] \subseteq \mathcal{MATIME}[t_2]$, which will imply that $L^{\mathtt{diag}} \in \mathcal{MATIME}[t_2] \setminus \mathcal{SIZE}[t_0]$. (Indeed, in this case we are proving a stronger result, since the $\mathcal{MA}$ verifiers do not need advice, and since the circuits are of size t_0 rather than $s = t_0^{1/4}$.)

To do so, let $L \in \mathcal{DSPACE}[t_1]$, and consider the following $\mathcal{MA}$ verifier for L. On input $x \in \{0,1\}^n$, the verifier computes $x' = R_L(x)$, where R_L is the machine from Lemma 4.2.22. Note that $n' = |x'| \leq \mathrm{poly}(t_1(n))$, and that $x \in L \iff x' \in L^{\mathtt{TV}}$. Now, the verifier parses the witness $w \in \{0,1\}^{\mathrm{poly}(t_0(n'))}$ as a description of a circuit $C : \{0,1\}^{n'} \to \{0,1\}$ of size $t_0(n')$, and runs the machine M from Lemma 4.2.22 on input x', while answering each oracle query of M using the circuit C.

Note that, since $L^{\mathtt{TV}} \in \mathcal{SIZE}[t_0]$, there exists a circuit C over n' input bits of size $t_0(n')$ that correctly computes $L^{\mathtt{TV}}$ on inputs of length n'. Therefore, by Lemma 4.2.22, when $x \in L$ there exists a witness such that the verifier accepts x with probability one, whereas the verifier rejects any $x \notin L$ with probability at least $2/3$, regardless of the witness. The total running time of the verifier is dominated by the time it takes to simulate M using the circuit C, which is at most $\mathrm{poly}(n') \cdot \mathrm{poly}(t_0(n')) \leq t_2(n)$.

Case 2: $L^{\mathtt{TV}} \notin \mathcal{SIZE}[t_0]$**.** In this case we show an explicit set $L^{\mathtt{pad}}$, which will be a padded version of $L^{\mathtt{TV}}$, such that $L^{\mathtt{pad}}$ can be decided in $\mathcal{MATIME}[t_2]$ with one bit of advice, but cannot be decided by circuits of size $s = t_0^{1/4}$. To do so, let $\mathtt{sz}_{\mathtt{TV}} : \mathbb{N} \to \mathbb{N}$ be such that $\mathtt{sz}_{\mathtt{TV}}(n)$ is the minimum circuit size for $L_n^{\mathtt{TV}} = L^{\mathtt{TV}} \cap \{0,1\}^n$. Also, for any integer m, let $p(m) = 2^{\lfloor \log(m) \rfloor}$ be the largest power of two that is not larger than m, and let $n(m) = m - p(m)$. We think of $n(m)$ as the "effective input length" indicated by m, and on $p(m)$ as the length of padding. We define the set $L^{\mathtt{pad}}$ as follows:

$$L^{\mathtt{pad}} = \Big\{ (x, 1^p) : \; x \in L^{\mathtt{TV}}, \text{ and } |x| = n(|x| + p), \\ \text{and } t_0(|x| + p) \leq \mathtt{sz}_{\mathtt{TV}}(|x|)^3 < t_0(|x| + 2p) \Big\} .$$

Let us first see that $L^{\mathtt{pad}}$ cannot be decided by circuits of size $t_0^{1/4}$. Assume towards a contradiction that there exists a circuit family $\{C_m\}$ of size $t_0^{1/4}$ that decides $L_m^{\mathtt{pad}}$ correctly for every m. Since $L^{\mathtt{TV}} \notin \mathcal{SIZE}[t_0]$, there exists an infinite set $I \subseteq \mathbb{N}$ such that for every $n \in I$ it holds that $\mathtt{sz}_{\mathtt{TV}}(n) > t_0(n)$. For a sufficiently large $n \in I$, we will construct a circuit $C'_n : \{0,1\}^n \to \{0,1\}$ of size less than $\mathtt{sz}_{\mathtt{TV}}(n)$ that computes $L_n^{\mathtt{TV}}$, which yields a contradiction to the definition of $\mathtt{sz}_{\mathtt{TV}}$.

Specifically, consider the circuit $C'_n : \{0,1\}^n \to \{0,1\}$ that acts as follows. Let p be a power of two such that $t_0(n+p) \leq \mathtt{sz}_{\mathtt{TV}}{}^3(n) < t_0(n+2p)$; there exists such a p since $t_0(n + 2^{\lceil \log(n) \rceil}) \leq t_0(n)^3 < \mathtt{sz}_{\mathtt{TV}}{}^3(n)$. The value of this p is hard-coded into C'_n. Given

[16] For example, $L^{\mathtt{diag}} = \{x : C_{|x|}(x) = 1\}$, where C_n is the lexicographically-first circuit over n bits of size at most $t_0^2(n)$ that decides a set whose circuit complexity is more than $t_0(n)$. The proof that $L^{\mathtt{diag}} \in \mathcal{DSPACE}[t_1]$ follows the well-known idea used in Kannan's theorem (see, e.g., [Juk12, Lem. 20.12]).

$x \in \{0,1\}^n$, the circuit C'_n pads x with 1^p, simulates the circuit C_m on $(x, 1^p)$ (where $m = n + p$), and outputs $C_m(x, 1^p)$. By the definition of $L^{\mathtt{pad}}$ it holds that C'_n correctly computes $L^{\mathtt{TV}}_n$. The size of C'_n is dominated by the size of C_m, and is thus at most $O(t_0(n+p)^{1/4}) = o(t_0(n+p)^{1/3})$. Since $t_0(n+p)^{1/3} \leq \mathtt{sz}_{\mathtt{TV}}(n)$ and n is sufficiently large, the size of C'_n is less than $\mathtt{sz}_{\mathtt{TV}}(n)$, which yields a contradiction.

Let us now see that $L^{\mathtt{pad}}$ can be decided by an MA verifier that runs in time t_2 and uses one bit of advice. Given an input z of length m, the advice bit is set to one if and only if $L^{\mathtt{pad}}_m \neq \emptyset$; if the advice is zero, the verifier immediately rejects. Otherwise, the verifier computes $n = n(m)$ and $p = p(m)$, and parses the input z as $(x, 1^p)$ where $|x| = n$ (if the verifier fails to parse the input, it immediately rejects). The verifier parses the witness $w \in \{0,1\}^{\mathrm{poly}(t_0(n+2p))}$ as a circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most $t_0(n+2p)^{1/3}$, and emulates the machine M from Lemma 4.2.22 on input x, answering each oracle query of M using the circuit C. The verifier outputs the decision of M.

Since $\mathtt{sz}_{\mathtt{TV}}(n) < t_0(n+2p)^{1/3}$, there exists a circuit C of size at most $t_0(n+2p)^{1/3}$ that computes $L^{\mathtt{TV}}_n$. For any $z \in L^{\mathtt{pad}}$, when the witness represents this circuit, the verifier accepts z with probability one. Also, for any $z \notin L^{\mathtt{TV}}$, the verifier rejects x with probability $2/3$, regardless of the witness. Finally, note that the running time of the verifier is dominated by the time that it takes to run the machine M while simulating the oracle answers, which is at most $\mathrm{poly}(n) \cdot \mathrm{poly}(t_0(2m)) \leq t_2(m)$. ■

4.2.5.2 Sufficient conditions for admissibility

The point of the current appendix is to show that essentially any increasing function $f(n) = \omega(1)$ such that $f(n) \leq n$ is admissible (in the sense of Definition 4.2.12).

Claim 4.2.23. *Let $f(n) = \omega(1)$ be any increasing function such that $f(n) \leq n$ for all n, and $t(n) = n^{f(n)}$ is time-constructible, and $s(n) = n^{\log(f(\log(n)))}$ is time-constructible, and $s'(n)$ is time-constructible. Then, f is admissible.*

Proof. Let $g(n) = \log(f(\log(n)))$ and let $s(n) = n^{g(n)}$. We need to verify that g is super-constant (which holds because f is super-constant), and that t and s are sufficiently gapped, and that $\hat{s}(n) = n^{o(f(n))}$. To see that t and s are sufficiently gapped, first note that both functions are increasing (since f is increasing, and hence g is also increasing) and are time-constructible, as is s' (we assumed time-constructibility in the hypothesis). Also note that $s(n) \leq n^{\log\log(n)} < 2^{n/\gamma}/n$.

Thus, it is left to verify that $\hat{s}(n) = n^{o(f(n))}$. The proof of this fact amounts to the following elementary calculation. First note that

$$s'(n) = (s(\gamma \cdot n))^{\gamma} = (\gamma \cdot n)^{\gamma \cdot \log(f(\log(\gamma \cdot n)))} < n^{\log^2(f(\log^2(n)))} .$$

Thus, for any function $k = k(n)$ and constant $c \geq 2$ such that $k(n) \leq \log^c(f(\log^{3c}(n)))$ (which in particular implies that $k(n) \leq \log^c(n)$), we have that

$$s'(n^k) < n^{k \cdot \log^2(f(\log^2(n^k)))} \leq n^{\log^{2c}(f(\log^{3c}(n)))} . \tag{4.2.3}$$

In particular, using Eq. (4.2.3) with $k(n) = \log^2(f(\log^2(n)))$ and $c = 2$, we deduce that $s'(s'(n)) < n^{\log^4(f(\log^6(n)))}$. Then, using Eq. (4.2.3) again with $k(n) = \log^4(f(\log^6(n)))$ and $c = 4$, we deduce that $s'(s'(s'(n))) < n^{\log^8(f(\log^{12}(n)))}$. Therefore, we have that $\hat{s}(n) < n^{\gamma' \cdot \log^8(f(\log^{12}(n)))} < n^{\gamma' \cdot \text{poly}\log(f(n))} = n^{o(f(n))}$. ■

4.3 Uniform lower bounds and average-case derandomization

4.3.1 The main results

Assume that some function in $\mathcal{PSPACE}$ cannot be computed by uniform probabilistic algorithms that run in time $2^{n/\text{polylog}(n)}$. Intuitively, using "hardness-to-randomness" results, we expect that such a strong lower bound would imply a strong derandomization result. For context, recall that in *non-uniform* hardness-to-randomness results (following [NW94]), lower bounds for non-uniform circuits yield pseudorandom generators (PRGs) that "fool" non-uniform distinguishers. Moreover, these results "scale smoothly" such that lower bounds for larger circuits yield PRGs with longer stretch (see [Uma03] for an essentially optimal trade-off); at the extreme, if $\mathcal{E}$ is hard almost-always for exponential-sized circuits, then we obtain PRGs with exponential stretch and deduce that $pr\mathcal{BPP} = pr\mathcal{P}$ (see [IW99]).

The key problem, however, is that the long line-of-works concerning *uniform* "hardness-to-randomness" did not yield such smooth trade-offs so far (see [IW98; CNS99; Kab01; Lu01; GSTS03; TV07; SU07; GV08; Gol11; CIS18]). Ideally, given an exponential lower bound for uniform probabilistic algorithms (such as $\mathcal{E} \not\subseteq \texttt{i.o.}\mathcal{BPTIME}[2^{\epsilon \cdot n}]$) we would like to deduce that there exists a PRG with exponential stretch for uniform circuits, and consequently that $\mathcal{BPP} = \mathcal{P}$ in "average-case".[17] However, prior to the current work, the state-of-the-art (by Trevisan and Vadhan [TV07]) could at best yield PRGs with *sub-exponential stretch* (i.e., with seed length $\text{polylog}(n)$), even if the hypothesis refers to an exponential lower bound. Moreover, the best currently-known PRG only works infinitely-often, even when we assume that the "hard" function cannot be computed by probabilistic algorithms on almost all input lengths.

Previous works bypassed these two obstacles in various indirect ways. Goldreich [Gol11] relied on the (much) stronger hypothesis $pr\mathcal{BPP} = pr\mathcal{P}$ to construct an "almost-always" PRG with exponential stretch for uniform circuits. Similarly, Carmosino, Impagliazzo, and Sabin [CIS18] relied on hypotheses from *fine-grained complexity* (recall that these are qualitatively strong, and implied by the "strong" version of rETH, i.e. by rSETH) to bypass both obstacles and derandomize $\mathcal{BPP}$ "almost-

[17] Throughout the paper, when we say that a PRG is ϵ-pseudorandom for *uniform circuits*, we mean that for every efficiently-samplable distribution over circuits, the probability over choice of circuit that the circuit distinguishes the output of the PRG from uniform with advantage more than ϵ is at most ϵ (see Definitions 2.4.14 and 2.4.11). The existence of such PRGs implies an "average-case" derandomization of $\mathcal{BPP}$ in the following sense: For every $L \in \mathcal{BPP}$ there exists an efficient deterministic algorithm D such that every probabilistic algorithm that gets input 1^n and tries to find $x \in \{0,1\}^n$ such that $D(x) \neq L(x)$ has a small probability of success (see, e.g., [Gol11, Prop. 4.4]).

always" on average-case in polynomial time; however, their derandomization does not rely on a PRG construction, and satisfies a weaker notion of average-case derandomization than the notion that we use.[18] Gutfreund and Vadhan [GV08] bypassed the "almost-always" barrier by deducing (subexponential-time) derandomization of $\mathcal{RP}$ rather than of $\mathcal{BPP}$ (see details below). Lastly, a line-of-works dealing with uniform "hardness-to-randomness" for $\mathcal{AM}$ (rather than for $\mathcal{BPP}$) was able to bypass both obstacles in this context (see, e.g., [Lu01; GSTS03; SU07]).

In this work we tackle both obstacles directly. First, we establish for the first time that hardness assumptions for $\mathcal{BPTIME}$ yield a pseudorandom generator for uniform circuits with *near-exponential stretch* (i.e., with seed length $\tilde{O}(\log(n))$), which can be used for average-case derandomization of $\mathcal{BPP}$ in nearly-polynomial-time (i.e., in time $2^{\tilde{O}(\log(n))} = n^{\mathrm{loglog}(n)^{O(1)}}$). Specifically, we start from the hypothesis that the Totally Quantified Boolean Formula (TQBF) problem cannot be solved by probabilistic algorithms that run in time $2^{n/\mathrm{polylog}(n)}$; this hypothesis is *weaker* than rETH (since 3-SAT reduces to TQBF with a linear overhead). Under this hypothesis, we show that there exists a PRG for uniform circuits with seed length $\tilde{O}(\log(n))$ that is computable in time $2^{\tilde{O}(\log(n))} = n^{\mathrm{loglog}(n)^{O(1)}}$.

Theorem 4.3.1 (rETH $\Rightarrow$ PRG with almost-exponential stretch for uniform circuits; informal)**.** *Suppose that there exists $T(n) = 2^{n/\mathrm{polylog}(n)}$ such that $\mathtt{TQBF} \notin \mathcal{BPTIME}[T]$. Then, for every $t(n) = n^{\mathrm{polyloglog}(n)}$, there exists a PRG that has seed length $\tilde{O}(\log(n))$, runs in time $n^{\mathrm{polyloglog}(n)}$, and is infinitely-often $(1/t)$-pseudorandom for every distribution over circuits that can be sampled in time t with $\log(t)$ bits of non-uniform advice.*

The proof of Theorem 4.3.1 is based on careful refinements of the proof framework of [IW98], using new technical tools that we construct. The latter tools significantly refine and strengthen the technical tools that were used by [TV07] to obtain the previously-best uniform hardness-to-randomness tradeoff. For high-level overviews of the proof of Theorem 4.3.1 (and of the new constructions), see Section 4.3.2.

Overcoming the "infinitely-often" barrier. The hypothesis in Theorem 4.3.1 is that any probabilistic algorithm that runs in time $2^{n/\mathrm{polylog}(n)}$ fails to compute TQBF *infinitely-often*, and the corresponding conclusion is that the PRG "fools" uniform circuits only *infinitely-often*. This is identical to all previous uniform "hardness-to-randomness" results that used the [IW98] proof framework.[19]

Gutfreund and Vadhan [GV08, Sec 6] showed one way to overcome this "infinitely-often" barrier, by deducing almost-always average-case derandomization of $\mathcal{RP}$ (rather than of $\mathcal{BPP}$) under an almost-always lower bound hypothesis; as in previous results, their derandomization is relatively slow (i.e., it works in sub-exponential time). Combining their ideas with the techniques underlying Theorem 4.3.1, we prove that un-

[18] Specifically, they deduce an average-case derandomization of $\mathcal{BPP}$ with respect to the *uniform* distribution, rather than with respect to every polynomial-time-samplable distribution.

[19] Other proof strategies (which use different hypotheses) were able to support an "almost-always" conclusion, albeit not necessarily a PRG, from an "almost-always" hypothesis (see [GSTS03; CIS18]).

der the hypothesis that rETH holds almost-always, $\mathcal{RP}$ can be derandomized almost-always in average-case and in (nearly-)polynomial time (see Theorem 4.3.16).

In addition, their techniques can be adapted to yield an almost-always PRG (from an almost-always lower bound hypothesis) that uses $O(\log(n))$ bits of non-uniform advice. We are able to significantly improve this: Assuming that every probabilistic algorithm that runs in time $2^{n/\text{polylog}(n)}$ fails to decide TQBF on almost all input lengths, we prove that $\mathcal{BPP}$ can be derandomized in average-case and almost-always, using only a *triply-logarithmic* number (i.e., $O(\text{logloglog}(n))$) of advice bits.

Theorem 4.3.2 (aa-rETH $\Rightarrow$ almost-always derandomization in time $n^{\text{polyloglog}(n)}$; informal)**.** *Assume that for some $T(n) = 2^{n/\text{polylog}(n)}$ it holds that $\texttt{TQBF} \notin \texttt{i.o.}\mathcal{BPTIME}[T]$, and let $t(n) = n^{\text{polyloglog}(n)}$. Then, for every $L \in \mathcal{BPTIME}[t]$ and every distribution ensemble $\mathcal{X} = \{\mathcal{X}_n \subset \{0,1\}^n\}$ such that $x \sim \mathcal{X}_n$ can be sampled in time $t(n)$, there exists a deterministic algorithm $D = D_{\mathcal{X}}$ that runs in time $n^{\text{polyloglog}(n)}$ and uses $O(\text{logloglog}(n))$ bits of non-uniform advice such that for* almost all input lengths $n \in \mathbb{N}$ *it holds that* $\Pr_{x\sim\mathcal{X}_n}[D(x) \neq L(x)] < 1/t(n)$.

Remark: Non-deterministic extensions. We note that "scaled-up" versions of Theorems 4.3.1 and 4.3.2 for *non-deterministic settings* follow easily from known results; that is, assuming lower bounds for non-deterministic uniform algorithms, we can deduce strong derandomization of corresponding non-deterministic classes. First, from the hypothesis MAETH[20] we can deduce strong circuit lower bounds, and hence also *worst-case* derandomization of $pr\mathcal{BPP}$ and of $pr\mathcal{MA}$ (this uses relatively standard Karp-Lipton-style arguments, following [BFN+93]; see Appendix 4.3.6.1 for details and for a related result). Similarly, as shown by Gutfreund, Shaltiel, and Ta-Shma [GSTS03], a suitable variant of AMETH implies an average-case derandomization of $\mathcal{AM}$.

4.3.2 Proof overviews

Recall that in typical "hardness-to-randomness" results, a PRG is based on a hard function, and the proof amounts to showing that an efficient distinguisher for the PRG can be transformed to an efficient algorithm or circuit that computes the hard function.

In high-level, our proof strategy follows this paradigm, using the classic approach of Impagliazzo and Wigderson [IW98]. Their approach for transforming a distinguisher to an algorithm for the hard function works only when the hard function $f^{\texttt{ws}}\colon \{0,1\}^* \to \{0,1\}^*$ is well-structured; the precise meaning of the term "well-structured" differs across different follow-up works, and in the current proof it will also take on a new meaning, but for now let us intuitively think of $f^{\texttt{ws}}$ as downward self-reducible and as having properties akin to random self-reducibility. Instantiating the Nisan-Wigderson PRG with a suitable encoding $\texttt{ECC}(f^{\texttt{ws}})$ of $f^{\texttt{ws}}$ (again, the precise

[20] Note that indeed a non-deterministic analogue of rETH is MAETH (or, arguably, AMETH), rather than NETH, due to the use of randomness. Also recall that, while the "strong" version of MAETH is false (see [Wil16]), there is currently no evidence against the "non-strong" version MAETH.

requirements from ECC differ across works), our goal is to show that if the PRG with stretch $t(n)$ does not "fool" uniform distinguishers even infinitely-often, then $f^{\mathtt{ws}}$ is computable in probabilistic time $t'(n) > t(n)$.

The key challenge underlying this approach is the *significant overheads* in the proof, which increase the time complexity t' of computing $f^{\mathtt{ws}}$. In the original proof of [IW98] this time was roughly $t'(n) \approx t(t(n))$, and the state-of-the-art (prior to the results described here), by Trevisan and Vadhan [TV07] (following [CNS99]), yielded $t'(n) = \mathrm{poly}(t(\mathrm{poly}(n)))$. Since the relevant functions $f^{\mathtt{ws}}$ in all works are computable in $\mathcal{E}$, proofs with such an overhead can yield at most a sub-exponential stretch $t(n) = 2^{n^{\Omega(1)}}$.

As mentioned in Section 4.3.1, Goldreich [Gol11] bypassed this difficulty by using the stronger hypothesis $pr\mathcal{BPP} = pr\mathcal{P}$, whereas Carmosino, Impagliazzo, and Sabin [CIS18] bypassed this difficulty by using hypotheses from fine-grained complexity (that are implied by the "strong" version of rETH, i.e. by rSETH). In contrast, we take a brute-force approach: We reduce *all* of the polynomial overheads in the input length to *polylogarithmic overheads* in the input length. That is, we will show that for carefully-constructed $f^{\mathtt{ws}}$ and suitably-chosen ECC (and with some variations in the proof), if the PRG instantiated with $\mathtt{ECC}(f^{\mathtt{ws}})$ for stretch t does not "fool" uniform distinguishers infinitely-often, then $f^{\mathtt{ws}}$ can be computed in time $t'(n) = t(\widetilde{O}(n))^{O(1)}$.

4.3.2.1 The well-structured function $f^{\mathtt{ws}}$

Following Trevisan and Vadhan [TV07], our $f^{\mathtt{ws}}$ is an artificial $\mathcal{PSPACE}$-complete problem that we carefully construct. Their goal was to construct $f^{\mathtt{ws}}$ that will be simultaneously downward self-reducible and randomly self-reducible. They achieved this by constructing a function based on the proof of $\mathcal{IP} = \mathcal{PSPACE}$ [LFK+92; Sha92]: Loosely speaking, at input length $N = \mathrm{poly}(n)$ the function gets as input a 3-SAT formula φ over n variables, and outputs $P^{(\varphi,N)}(\varphi) = Q_1 \circ Q_2 \circ ... \circ Q_{\mathrm{poly}(n)} P^{(\varphi)}$, where $P^{(\varphi)}$ is an arithmetization of φ and the Q_i's are arithmetic operators from the $\mathcal{IP} = \mathcal{PSPACE}$ proof such that $P^{(\varphi,N)}(\varphi) = \mathtt{TQBF}(\varphi)$; and at input length $N - i$, the function gets input (φ, w) and outputs $P^{(\varphi,N-i)}(\varphi, w)$, where $P^{(\varphi,N-i)}$ is the polynomial that applies one less operator to $P^{(\varphi)}$ than $P^{(\varphi,N-i+1)}$ and fixes some input variables for $P^{(\varphi)}$ according to w. Since $f^{\mathtt{ws}}$ consists of low-degree polynomials, it is randomly self-reducible; and since each $P^{(\varphi,N-i)}$ is obtained by applying a simple operator to $P^{(\varphi,N-(i-1))}$, the function $f^{\mathtt{ws}}$ is also downward self-reducible.

Going through their proof (with needed adaptations for our "high-end" parameter setting), we encounter four different polynomial overheads in the input length. The first and obvious one is that inputs of length n are mapped to inputs of length $N = \mathrm{poly}(n)$, corresponding to the number of polynomials in the $\mathcal{IP} = \mathcal{PSPACE}$ protocol. The other polynomial overheads in the input length come from their reduction of TQBF to an intermediate problem that takes both φ and w as part of the input and is still amenable to arithmetization,[21] from the field size that is required for the strong

[21] Recall that the standard arithmetization of 3-SAT is a polynomial that depends on the input formula, whereas we want a single polynomial that gets both a formula and the assignment as input.

random self-reducibility that is needed in our parameter setting (see below), and from the way the $\text{poly}(n)$ polynomials are combined into a single Boolean function.

The main challenge is to eliminate all of the foregoing overheads *simultaneously*, rather than separately. We will achieve this by presenting a construction of a suitable $f^{\mathtt{ws}}$, which is a refinement of their construction, and constitutes the main technical part in the proof of Theorem 4.3.1. We now outline (very briefly) the key points underlying the construction; for a detailed overview we refer the reader to Section 4.3.3. After the following brief outline, we will explain how we use $f^{\mathtt{ws}}$ to prove Theorem 4.3.1.

Our main idea is to use an $\mathcal{IP} = \mathcal{PSPACE}$ protocol with $\text{polylog}(n)$ rounds instead of $\text{poly}(n)$ rounds, so that the first overhead (i.e., the additive overhead in the input length caused by the number of operators) will be only $\text{polylog}(n)$ instead of $\text{poly}(n)$. Indeed, in such a protocol the verification time in each round is high, and therefore our downward self-reducibility algorithm is relatively slow and makes many queries; but we will be able to afford this in our proof (since eventually we only need to solve TQBF in time $2^{n/\text{polylog}(n)}$). While implementing this idea, we define a different intermediate problem that is both amenable to arithmetization and reducible from TQBF in quasilinear time (see Claim 4.3.9.1); we move to an arithmetic setting that will support the strong random self-reducibility that we want (see details below), and arithmetize the intermediate problem in this setting (see Claim 4.3.9.2); we show how to execute arithmetic operators in a "batch" in this arithmetic setting (see Claim 4.3.9.3); and we combine the resulting collection of polynomials into a single Boolean function. We stress that we are "paying" for all the optimizations above, by the fact that the associated algorithms (for downward self-reducibility and for our notion of random self-reducibility that will be described next) now run in time $2^{n/\text{polylog}(n)}$, instead of in polynomial time; but again, we will be able to afford this.

We obtain a function $f^{\mathtt{ws}}$ with the following properties: First, $f^{\mathtt{ws}}$ is computable in linear space; secondly, TQBF is reducible to $f^{\mathtt{ws}}$ in *quasilinear time*; thirdly, $f^{\mathtt{ws}}$ is *downward self-reducible* in time $2^{n/\text{polylog}(n)}$; and lastly, $f^{\mathtt{ws}}$ is sample-aided worst-case to δ-average-case reducible, for $\delta(n) = 2^{-n/\text{polylog}(n)}$. The last property, which is implicit in many works and was recently made explicit by Goldreich and G. Rothblum [GR17], asserts the following: There exists a uniform algorithm T that gets as input a circuit $C\colon \{0,1\}^n \to \{0,1\}^*$ that agrees with $f_n^{\mathtt{ws}}$ on at least $\delta(n)$ of the inputs, and *labeled examples* $(x, f^{\mathtt{ws}}(x))$ where $x \in \{0,1\}^n$ is uniformly-chosen, runs in time $2^{n/\text{poly}\log(n)}$ and with high probability outputs a circuit $C'\colon \{0,1\}^n \to \{0,1\}^*$ that computes $f_n^{\mathtt{ws}}$ on all inputs (see Definition 4.3.4).

4.3.2.2 Instantiating the [IW98] proof framework with the function $f^{\mathtt{ws}}$

Given this construction of $f^{\mathtt{ws}}$, we now use a variant of the [IW98] proof framework, as follows. (For simplicity, we show how to "fool" polynomial-time distinguishers that do not use advice.) Let ECC be the Goldreich-Levin [GL89] (i.e., Hadamard) encoding $\mathtt{ECC}(f^{\mathtt{ws}})(x,r) = \oplus_i f^{\mathtt{ws}}(x)_i \cdot r_i$. The argument of [IW98] (following [NW94]) shows that if for input length n there exists a uniform $\text{poly}(n)$-time distinguisher A for the

Nisan-Wigderson PRG (instantiated with $\mathtt{ECC}(f^{\mathtt{ws}})$) that succeeds with advantage $1/n$, then for input length $\ell = \widetilde{O}(\log(n))$ (corresponding to the set-size in the underlying combinatorial design) there is a *weak learner* for $\mathtt{ECC}(f^{\mathtt{ws}})$: That is, there exists an algorithm that gets oracle access to $\mathtt{ECC}(f^{\mathtt{ws}})$, runs in time $\mathrm{poly}(n) \approx 2^{\ell/\mathrm{polylog}(\ell)}$, and outputs a small circuit that agrees with $\mathtt{ECC}(f^{\mathtt{ws}})$ on approximately $1/2 + 1/n^2 \approx 1/2 + \delta_0(\ell)$ of the ℓ-bit inputs, where $\delta_0(\ell) = 2^{-\ell/\mathrm{polylog}(\ell)}$.

Assuming that there exists a distinguisher for the PRG as above for every $n \in \mathbb{N}$, we deduce that a weak learner exists for every $\ell \in \mathbb{N}$. Following [IW98], for each input length $i = 1, ..., \ell$ we construct a circuit of size $2^{i/\mathrm{polylog}(i)}$ for $f_i^{\mathtt{ws}}$. Specifically, in iteration i we run the learner for $\mathtt{ECC}(f^{\mathtt{ws}})$ on input length $2i$, and answer its oracle queries using the downward self-reducibility of $f^{\mathtt{ws}}$, the circuit that we have for $f_{i-1}^{\mathtt{ws}}$, and the fact that $\mathtt{ECC}(f^{\mathtt{ws}})_{2i}$ is easily computable given access to $f_i^{\mathtt{ws}}$. The learner outputs a circuit of size $2^{2i/\mathrm{polylog}(2i)}$ that agrees with $\mathtt{ECC}(f^{\mathtt{ws}})$ on approximately $1/2 + \delta_0(2i)$ of the $2i$-bit inputs, and the argument of [GL89] allows to efficiently transform this circuit to a circuit of similar size that computes $f^{\mathtt{ws}}$ on a approximately $\delta(i) = \mathrm{poly}(\delta_0(2i))$ of the i-bit inputs. Our goal now is to transform this circuit to a circuit of similar size that computes $f^{\mathtt{ws}}$ on all i-bit inputs. Recall that in general, performing such transformations by a *uniform* algorithm is challenging (intuitively, if $f^{\mathtt{ws}}$ is a codeword in an error-correcting code, this corresponds to uniform list-decoding of a "very corrupt" version of $f^{\mathtt{ws}}$). However, in our specific setting we can produce random *labeled samples* for $f^{\mathtt{ws}}$, using its downward self-reducibility and the circuit that we have for $f_{i-1}^{\mathtt{ws}}$. Relying on the *sample-aided worst-case to average-case reducibility* of $f^{\mathtt{ws}}$, we can transform our circuit to a circuit of similar size that computes $f_i^{\mathtt{ws}}$ on all inputs.

Finally, since $\mathtt{TQBF}$ is reducible with quasilinear overhead to $f^{\mathtt{ws}}$, if we can compute $f^{\mathtt{ws}}$ in time $2^{n/\mathrm{polylog}(n)}$ then we can compute $\mathtt{TQBF}$ in such time. Moreover, since $f^{\mathtt{ws}}$ is computable in space $O(\ell) = \tilde{O}(\log(n))$ (and thus in time $n^{\mathrm{polyloglog}(n)}$), the pseudorandom generator is computable in time $n^{\mathrm{polyloglog}(n)}$.

4.3.2.3 The "almost-always" version: Proof of Theorem 4.3.2

We now explain how we can adapt the proof above in order to get an "almost-always" PRG with near-exponential stretch. For starters, we will use a stronger property of $f^{\mathtt{ws}}$, namely that it is downward self-reducible in a polylogarithmic number of steps; this means that for every input length ℓ there exists an input length $\ell_0 \geq \ell - \mathrm{polylog}(\ell)$ such that $f^{\mathtt{ws}}$ is efficiently-computable at input length ℓ_0 (i.e., $f_{\ell_0}^{\mathtt{ws}}$ is computable in time $2^{\ell_0/\mathrm{polylog}(\ell_0)}$ without a "downward" oracle; see Section 4.3.3.1 for intuition and details).

Now, observe that the transformation of a probabilistic distinguisher A for the PRG to a probabilistic algorithm F that computes $f^{\mathtt{ws}}$ actually gives a "point-wise" guarantee: For every input length $n \in \mathbb{N}$, if A distinguishes the PRG on a corresponding set of input lengths S_n, then F computes $f^{\mathtt{ws}}$ correctly at input length $\ell = \ell(n) = \widetilde{O}(\log(n))$; specifically, S_n is the set of input lengths at which we need a distinguisher for G, in order to obtain a weak learner for $\mathtt{ECC}(f^{\mathtt{ws}})$ at smaller input lengths, and use the

downward self-reducibility argument for $f^{\mathtt{ws}}$ at input lengths $\ell, \ell-1, ..., \ell_0$. Moreover, since $f^{\mathtt{ws}}$ is downward self-reducible in polylog steps, we will only need weak learners at inputs $\ell, \ell-1, ..., \ell_0 = \ell - \text{polylog}(\ell)$; hence, we can show that S_n is a set of $\text{polylog}(\ell) = \text{polyloglog}(n)$ input lengths in the interval $[n, n^2]$ (see Lemma 4.3.11 for the precise calculation). Taking the contrapositive, if $f^{\mathtt{ws}}$ cannot be computed by F on almost all ℓ's, then for every $n \in \mathbb{N}$ there exists an input length $m \in S_n \subset [n, n^2]$ such that G fools A at input length m.[22]

Our derandomization algorithm gets input 1^n and also gets the "good" input length $m \in S_n$ *as non-uniform advice*; it then simulates $G(1^m)$ (i.e., the PRG at input length m) and truncates the output to n bits. (We can indeed show that truncating the output of our PRG preserves its pseudorandomness in a uniform setting; see Proposition 4.3.14 for details.) The crucial point is that since $|S_n| = \text{polyloglog}(n)$, the advice length is $O(\text{logloglog}(n))$. Note, however, that for every potential distinguisher A there exists a *different* input length $m \in S_n$ such that G is pseudorandom for A on m. Hence, our derandomization algorithm (or, more accurately, its advice) depends on the distinguisher that it wants to "fool". Thus, for every $L \in \mathcal{BPP}$ and every efficiently-samplable distribution $\mathcal{X}$ of inputs, there exists a corresponding "almost-always" derandomization algorithm $D_{\mathcal{X}}$ (see Proposition 4.3.14).

4.3.3 Construction of a well-structured function

In Section 4.3.3.1 we present the required properties of well-structured functions and define such functions. Then, in Section 4.3.3.2 we present a high-level overview of our construction of such functions. Finally, in Section 4.3.3.3 we present the construction itself in detail.

4.3.3.1 Well-structured function: Definition

Loosely speaking, we will say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is well-structured if it satisfies three properties. The *first property*, which is not crucial for our proofs but simplifies them a bit, is that f is length-preserving; that is, for every $x \in \{0,1\}^*$ it holds that $|f(x)| = |x|$.

The *second property* is a strengthening of the notion of downwards self-reducibility. Recall that a function $f : \{0,1\}^* \to \{0,1\}^*$ is downwards self-reducible if f_n can be computed by an efficient algorithm that has oracle access to f_{n-1}. First, we quantify the notion of "efficient", in order to also allow for very large running time (e.g., running time $2^{n/\text{polylog}(n)}$). Secondly, we also require that for any $n \in \mathbb{N}$ there exists an input length m that is not much smaller than n such that f_m is efficiently computable *without*

[22] Actually, since $f^{\mathtt{ws}}$ is downward self-reducible in polylog steps, it can be computed relatively-efficiently on infinitely-many input lengths, and thus cannot be "hard" for almost all ℓ's. However, since $\mathtt{TQBF}$ can be reduced to $f^{\mathtt{ws}}$ with quasilinear overhead, if $\mathtt{TQBF}$ is "hard" almost-always then for every $\ell(n)$ there exists $\ell' \leq \tilde{O}(\ell(n))$ such that $f^{\mathtt{ws}}$ is "hard" on ℓ', which allows our argument to follow through, with a similar set $\overline{S_n} \subset [n, n^{\text{polyloglog}(n)}]$ (see Proposition 4.3.13 for details). For simplicity, we ignore this issue in the overview.

any "downward" oracle. That is, intuitively, if we try to compute f on input length n by "iterating downwards" using downward self-reducibility, our "base case" in which the function is efficiently-computable is not input length $O(1)$, but a large input length m that is not much smaller than n. More formally:

Definition 4.3.3 (downward self-reducibility in few steps)**.** *For $t, s : \mathbb{N} \to \mathbb{N}$, we say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is* downward self-reducible in time t and s steps *if there exists a probabilistic oracle machine A that for any sufficiently large $n \in \mathbb{N}$ satisfies the following.*

1. *When A is given input $x \in \{0,1\}^n$ and oracle access to f_{n-1}, it runs in time at most $t(n)$ and satisfies $\Pr_r[A^{f_{n-1}}(x,r) = f(x)] \geq 2/3$.*
2. *There exists an input length $m \in [n - s(n), n]$ such that A computes f_m in time $t(m)$ without using randomness or oracle queries..*

In the special case that $s(n) = n$, we simply say that f is downward self-reducible in time t.

The third property that we need is a refinement of the notion of random self-reducibility, which is called sample-aided worst-case to average-case reducibility. This notion was recently made explicit by Goldreich and G. Rothblum [GR17], and is implicit in many previous results (see, e.g., the references in [GR17]).

To explain the notion, recall that if a function f is randomly self-reducible, then a circuit $\widetilde{C}$ that computes f on *most* of the inputs can be efficiently transformed to a (probabilistic) circuit C that computes f on *every* input (whp). We want to relax this notion, by allowing the efficient algorithm that transforms $\widetilde{C}$ into C to obtain *random labeled samples for* f (i.e., inputs of the form $(r, f(r))$ where r is chosen uniformly at random). The main advantage in this relaxation is that we will not need to assume that $\widetilde{C}$ computes f on *most* of the inputs, but will be satisfied with the weaker assumption that $\widetilde{C}$ computes f on a *tiny fraction* of the inputs. Specifically:[23]

Definition 4.3.4 (sample-aided reductions; see [GR17, Def 4.1])**.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a length-preserving function, and let $s : \mathbb{N} \to \mathbb{N}$ and $\delta_0 : \mathbb{N} \to [0,1)$. Let M be a probabilistic oracle machine that gets input 1^n and a sequence of $s(n)$ pairs of the form $(r,v) \in \{0,1\}^n \times \{0,1\}^n$ and oracle access to a function $\tilde{f}_n : \{0,1\}^n \to \{0,1\}^n$, and outputs a circuit $C : \{0,1\}^n \to \{0,1\}^n$ with oracle gates. We say that M is a* sample-aided reduction of computing f in the worst-case to computing f on δ_0 of the inputs using a sample of size s *if for every $\tilde{f}_n : \{0,1\}^n \to \{0,1\}^n$ satisfying $\Pr_{x \in \{0,1\}^n}[\tilde{f}_n(x) = f_n(x)] \geq \delta_0(n)$ the following holds: With probability at least $1 - \delta_0(n)$ over choice of $\bar{r} = r_1, ..., r_{s(n)} \in \{0,1\}^n$ and over the internal coin tosses of M, we have that $M^{\tilde{f}_n}(1^n, (r_i, f_n(r_i))_{i \in [s(n)]})$ outputs a circuit C such that $\Pr[C^{\tilde{f}_n}(x) = f_n(x)] \geq 2/3$ for every $x \in \{0,1\}^n$.*

[23] Definition 4.3.4 is actually a slightly modified version of the definition in [GR17]. First, we consider reductions of computing f in the worst-case to computing f in "rare-case", whereas [GR17] both reduce the computation of f to the computation of a possibly different function f', and parametrize the success probability of computing both f and f'. Secondly, we separately account for the success probability of the transformation M and of the final circuit C. And lastly, we also require f to be length-preserving.

Definition 4.3.5 (sample-aided worst-case to average-case reducibility)**.** *For $\delta_0 : \mathbb{N} \to (0,1)$, we say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is* sample-aided worst-case to δ_0-average-case reducible *if there exists a sample-aided reduction M of computing f in worst-case to computing f on δ_0 of the inputs such that M runs in time $\mathrm{poly}(n, 1/\delta_0(n))$ and uses $\mathrm{poly}(1/\delta_0(n))$ samples.*

For high-level intuition of why labeled samples can be helpful for worst-case to average-case reductions, and for a proof that if f is a low-degree multivariate polynomial then it is sample-aided worst-case to average-case reducible, see Appendix 4.3.6.2.

We are now ready to define well-structured functions. Fixing a parameter $\delta > 0$, a function $f^{\mathtt{ws}}$ is δ-well-structured if it is length-preserving, downward self-reducible in time $\mathrm{poly}(1/\delta)$, and sample-aided worst-case to δ-average case reducible. That is:

Definition 4.3.6 (well-structured function)**.** *For $\delta : \mathbb{N} \to (0,1)$ and $s : \mathbb{N} \to \mathbb{N}$, we say that a function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ is (δ, s)-*well-structured *if $f^{\mathtt{ws}}$ is length-preserving, downward self-reducible in time $\mathrm{poly}(1/\delta)$ and s steps, and sample-aided worst-case to δ-average-case reducible. Also, when $s(n) = n$ (i.e., $f^{\mathtt{ws}}$ is simply downward self-reducible in time $\mathrm{poly}(1/\delta)$), we say that $f^{\mathtt{ws}}$ is δ-*well-structured*.*

In the following definition, we consider reductions from a decision problem $L \subseteq \{0,1\}^*$ to a well-structured function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$. To formalize this we consider both a reduction R, which transforms any input x for L to an input $R(x)$ for $f^{\mathtt{ws}}$, and a "decision algorithm" D, which translates the non-Boolean result $f^{\mathtt{ws}}(R(x))$ into a decision of whether or not $x \in L$.

Definition 4.3.7 (reductions to multi-output functions)**.** *Let $L \subseteq \{0,1\}^*$ and $f : \{0,1\}^* \to \{0,1\}^*$. For $t, b : \mathbb{N} \to \mathbb{N}$, we say that L* reduces to f in time t with blow-up b *if there exist two deterministic time-t algorithms R and D such that for every $x \in \{0,1\}^*$ it holds that $|R(x)| \leq b(|x|)$ and that $x \in L$ if and only if $D(f(R(x))) = 1$.*

4.3.3.2 Overview of our construction

For $\delta = 2^{-n/\mathrm{polylog}(n)}$ and $s = \mathrm{polylog}(n)$, our goal is to construct a (δ, s)-well-structured function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ such that TQBF reduces to $f^{\mathtt{ws}}$ in *quasi-linear time* (and thus with quasilinear blow-up). Throughout the section, assume that an n-bit input to TQBF is simply a 3-SAT formula φ on n variables, and it is assumed that all variables are quantified in-order, with alternating quantifiers (e.g., $\forall w_1 \exists w_2 \forall w_3 ... \varphi(w_1, ..., w_n)$; see Definition 4.3.8).

Our starting point is the well-known construction of Trevisan and Vadhan [TV07], which (loosely speaking) transforms the protocol underlying the $\mathcal{IP} = \mathcal{PSPACE}$ proof into a computational problem $L^{\mathtt{TV}} : \{0,1\}^* \to \{0,1\}^*$.[24] They required that $L^{\mathtt{TV}}$

[24] Actually, in [TV07] they define a *Boolean function*, which treats a suffix of its input as an index of an output bit in the non-Boolean version that we describe, and outputs the corresponding bit. To streamline our exposition we ignore this issue.

will meet the weaker requirements (compared to our requirements) of being downward self-reducible and randomly self-reducible, where the latter means reducible from being worst-case computabile to being computable on, say, .99 of the inputs.

Before describing our new construction, let us first review the original construction of $L^{\mathtt{TV}}$. For every $n \in \mathbb{N}$, fix a corresponding interval $I_n = [N_0, N_1]$ of $r(n) = \mathrm{poly}(n)$ input lengths. The input to $L^{\mathtt{TV}}$ at any input length in I_n (disregarding necessary padding) is a pair $(\varphi, w) \in \mathbb{F}^{2n}$, where $\mathbb{F}$ is a sufficiently-large field. If $(\varphi, w) \in \{0,1\}^{2n}$ then we think of φ as representing a 3-SAT formula and of w as representing an assignment. At input length N_0 we define $L^{\mathtt{TV}}(\varphi, w) = P(\varphi, w)$, where $P(\varphi, x)$ is a low-degree arithmetized version of the Boolean function $(\varphi, w) \mapsto \varphi(w)$.

Now, recall that the $\mathcal{IP} = \mathcal{PSPACE}$ protocol defines three arithmetic operators on polynomials (two quantification operators and a linearization operator). Then, at input length $N_0 + i$, the problem $L^{\mathtt{TV}}$ is recursively defined by applying one of the three arithmetic operators on the polynomial from the previous input length $N_0 + i - 1$.[25] Observe that computing $L^{\mathtt{TV}}$ at input length $N_0 + i$ corresponds to the residual computational problem that the verifier faces at the $(r-i)^{th}$ round of the $\mathcal{IP} = \mathcal{PSPACE}$ protocol, when instantiated for formula φ and with $r = r(n)$ rounds. Indeed, at the largest input length $N_1 = N_0 + r(n)$ the polynomial $L^{\mathtt{TV}}$ is simply a low-degree arithmetized version of the function that decides whether or not $\varphi \in \mathtt{TQBF}$ (regardless of w); thus, TQBF can be reduced to $L^{\mathtt{TV}}$ by mapping $\varphi \in \{0,1\}^n$ to $(\varphi, 1^n) \in \mathbb{F}^{2n}$ and adding padding to get the input to be of length $N_1 = \mathrm{poly}(n)$. Note that $L^{\mathtt{TV}}$ is indeed both downward self-reducible (since for each operator O and polynomial P, we can compute $O(P)(\varphi, w)$ in polynomial-time with two oracle queries to P), and randomly self-reducible (since the polynomials have low degree.)

Let us now define our $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$, which replaces their $L^{\mathtt{TV}}$, and highlight what is different in our setting. Recall that our main goal is to construct the well-structured function $f^{\mathtt{ws}}$ such that TQBF is reducible to $f^{\mathtt{ws}}$ with *only quasilinear overhead* in the input length (i.e., we need to avoid polynomial overheads), while keeping the running time of all operations (i.e., of the algorithms for downward self-reducibility and for sample-aided worst-case to rare-case reducibility) to be *at most* $2^{n/\mathrm{polylog}(n)}$.

The first issue, which is relatively easy to handle, is the number of bits that we use to represent an (arithmetized) input (φ, w) for $f^{\mathtt{ws}}$. Recall that we want $f^{\mathtt{ws}}$ to be worst-case to δ-average-case reducible for a tiny $\delta = 2^{-n/\mathrm{polylog}(n)}$; thus, $f^{\mathtt{ws}}$ will involve computing polynomials over a field of large size $|\mathbb{F}| \geq \mathrm{poly}(1/\delta)$. Using the approach of [TV07], we would need $2n \cdot \log(|\mathbb{F}|) = \tilde{\Omega}(n^2)$ bits to represent (φ, w), and thus the reduction from TQBF to $f^{\mathtt{ws}}$ would incur a polynomial overhead. This is easily solvable by considering a "low-degree extension" instead of their "multilinear extension": To

[25] In more detail, we define three arithmetic operators on functions $\mathbb{F}^{2n} \to \mathbb{F}$, each indexed by a variable $j \in [n]$, and denote these operators by $\{\mathcal{O}^j_k\}_{k \in [3], j \in [n]}$. In each recursive step $i \in [r(n)]$, the polynomial corresponding to input length $N_0 + i$ is obtained by applying operator $\mathcal{O}^{j(i)}_{k(i)}$, where $j, k : \mathbb{N} \to [3]$ are polynomial-time computable functions, to the polynomial corresponding to input length $N_0 + i - 1$. Thus, at input length $N_0 + i$, we compute $L^{\mathtt{TV}}(\varphi, w)$ by applying i operators on the polynomial P and evaluating the resulting polynomial at (φ, w).

represent an input $(\varphi, w) \in \{0,1\}^{2n}$ to $f^{\mathtt{ws}}$ we will use *few elements in a very large field*. Specifically, we will use $\ell = \text{polylog}(n)$ variables (i.e., the polynomial will be $\mathbb{F}^{2^\ell} \to \mathbb{F}$) such that each variable "provides" $O(n/\text{polylog}(n))$ bits of information.

A second problem is constructing a low-degree arithmetization $P(\varphi, w)$ of the Boolean function that evaluates φ at w. In [TV07] they solve this by first reducing TQBF to an intermediate problem TQBF′ that is amenable to such low-degree arithmetization; however, their reduction incurs a quadratic blow-up in the input length, which we cannot afford in our setting. To overcome this we reduce TQBF to another intermediate problem, denoted $\mathtt{TQBF}^{\mathtt{loc}}$, which is amenable to low-degree arithmetization, such that the reduction incurs only a quasilinear blow-up in the input length. (Loosely speaking, we define $\mathtt{TQBF}^{\mathtt{loc}}$ by applying a very efficient Cook-Levin reduction to the Turing machine that gets input (φ, w) and outputs $\varphi(w)$; see Claim 4.3.9.1 for precise details.) We then carefully arithmetize $\mathtt{TQBF}^{\mathtt{loc}}$, while "paying" for this efficient arithmetization by the fact that computing the corresponding polynomial now takes time $\exp(n/\ell) = \text{poly}(1/\delta)$, instead of $\text{poly}(n)$ time as in [TV07] (see Claim 4.3.9.2).

Thirdly, the number of polynomials in the construction of $L^{\mathtt{TV}}$ (i.e., the size of the interval I_n) is $r(n) = \text{poly}(n)$, corresponding to the number of rounds in the $\mathcal{IP} = \mathcal{PSPACE}$ protocol. This poses a problem for us since the reduction from TQBF maps an input of length n is to an input of length $N_1 \geq \text{poly}(n)$. We solve this problem by "shrinking" the number of polynomials to be polylogarithmic, using an approach similar to an $\mathcal{IP} = \mathcal{PSPACE}$ protocol with only $\text{polylog}(n)$ rounds and a verifier that runs in time $2^{n/\text{polylog}(n)}$: Intuitively, at each input length, we define $f^{\mathtt{ws}}$ by simultaneously applying $O(\log(1/\delta))$ operators (rather than a single operator) to the polynomial that corresponds to the previous input length. Indeed, as one might expect, this increases the running-time of the downward self-reducibility algorithm to $\text{poly}(1/\delta)$, but we can afford this. Implementing this approach requires some care, since multiple operators will be applied to a single variable (which represents many bits of information), and since the linearization operator needs to be replaced by a "degree-lowering operation" (that will reduce the individual degree of a variable to be $\text{poly}(1/\delta)$); see Claim 4.3.9.3 for details.

Lastly, we also want our function to be downward self-reducible in $\text{polylog}(n)$ steps (i.e., after $\text{polylog}(n)$ "downward" steps, the function at the now-smaller input length is computable in time $\text{poly}(1/\delta)$ without an oracle). This follows by noting that the length of each interval I_n is now polylogarithmic, and that at the "bottom" input length the function $f^{\mathtt{ws}}$ simply computes the arithmetized version of $\mathtt{TQBF}^{\mathtt{loc}}$, which (as mentioned above) is computable in time $\text{poly}(1/\delta)$.

4.3.3.3 The construction itself

We consider the standard "totally quantified" variant of the Quantified Boolean Formula (QBF) problem, called Totally Quantified Boolean Formula (TQBF). In this version the quantifiers do not appear as part of the input, and we assume that all the variables are quantified, and that the quantifiers alternate according to the index of the variable

(i.e., x_i is quantified by $\exists$ if i is odd, and otherwise quantified by $\forall$).

Definition 4.3.8 (TQBF). *A string $\varphi \in \{0,1\}^*$ of length $n = |\varphi|$ is in the set* TQBF $\subseteq \{0,1\}^*$ *if φ is a representation of a* 3-SAT *formula in variables indexed by $[n]$ such that, denoting the variables by $w_1, ..., w_n$, it holds that $\exists w_1 \forall w_2 \exists w_3 \forall w_4 ... \varphi(w_1, ..., w_n)$. In other words, $\varphi \in$* TQBF *if the quantified expression that is obtained by quantifying all n variables, in order of their indices and with alternating quantifiers (starting with $\exists$), evaluates to true.*

Recall that QBF, in which the quantifiers are part of the input, is reducible in linear time to TQBF from Definition 4.3.8 (by renaming variables and adding dummy variables).

The main result in this section is a construction of a well-structured function $f^{\mathtt{ws}}$ such that TQBF can be reduced to $f^{\mathtt{ws}}$ with only quasilinear blow-up. This construction is detailed in the following lemma:

Lemma 4.3.9 (a well-structured set that is hard for TQBF under quasilinear reductions)**.** *There exists a universal constant $r \in \mathbb{N}$ such that for every constant $c \in \mathbb{N}$ the following holds. For $\ell(n) = \log(n)^{3c}$ and $\delta(n) = 2^{-n/\ell(n)}$, there exists a $(\delta, O(\ell^2))$-well-structured function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ such that $f^{\mathtt{ws}}$ is computable in linear space, and* TQBF *deterministically reduces to $f^{\mathtt{ws}}$ in time $n \cdot \log^{2c+r}(n)$.*

Proof. In high-level, we first reduce TQBF to a problem $\mathtt{TQBF}^{\mathtt{loc}}$ that will have a property useful for arithmetization, and then reduce $\mathtt{TQBF}^{\mathtt{loc}}$ to a function $f^{\mathtt{ws}}$ that we will construct as follows. We will first carefully arithmetize a suitable witness-relation that underlies $\mathtt{TQBF}^{\mathtt{loc}}$; then transform the corresponding arithmetic version of $\mathtt{TQBF}^{\mathtt{loc}}$ to a collection of low-degree polynomials that also satisfy a property akin to downward self-reducibility (loosely speaking, these polynomials arise from the protocol underlying the proof of $\mathcal{IP} = \mathcal{PSPACE}$ [LFK+92; Sha92]); and finally "combine" these polynomials to a Boolean function $f^{\mathtt{ws}}$ that will "inherit" the useful properties of the low-degree polynomials, and will thus be well-structured.

A variant of TQBF that is amenable to arithmetization. We will need a non-standard variant of TQBF, which we denote by $\mathtt{TQBF}^{\mathtt{loc}}$, such that TQBF is reducible to $\mathtt{TQBF}^{\mathtt{loc}}$ with quasilinear blow-up, and $\mathtt{TQBF}^{\mathtt{loc}}$ has an additional useful property. To explain this property, recall that the verification procedure of a "witness" $w = w_1, ..., w_n$ in TQBF is local, in the following sense: For every fixed φ it holds that $\varphi \in$ TQBF iff $\exists w_1 \forall w_2 ... \; 3SAT(\varphi, w)$, where $3SAT(\varphi, w) = \varphi(w)$ is a relation that can be decided by a conjunction of local conditions on the "witness" w. We want the stronger property that the relation that underlies $\mathtt{TQBF}^{\mathtt{loc}}$ can be tested by a conjunction of conditions that are local *both in the input and in the witness*. That is, denoting the underlying relation by $\mathtt{R\text{-}TQBF}^{\mathtt{loc}}$, we will have that $x \in \mathtt{TQBF}^{\mathtt{loc}}$ iff $\exists w_1 \forall w_2 ... \; \mathtt{R\text{-}TQBF}^{\mathtt{loc}}(x, w)$, where $\mathtt{R\text{-}TQBF}^{\mathtt{loc}}$ is a conjunction of local conditions on (x, w). In more detail:

Claim 4.3.9.1 (a variant of TQBF with verification that is local in both input and witness)**.** *There exists a set $\mathtt{TQBF}^{\mathtt{loc}} \in \mathcal{SPACE}[O(n)]$ and a relation $\mathtt{R\text{-}TQBF}^{\mathtt{loc}} \subseteq (\{0,1\}^* \times$*

$\{0,1\}^*$) *such that* $\texttt{TQBF}^{\texttt{loc}} = \{x : \exists w_1 \forall w_2 \exists w_3 \forall w_4 ... (x,w) \in \texttt{R-TQBF}^{\texttt{loc}}\}$, *and the following holds.*

1. *(Length-preserving witnesses.) For any* $(x,w) \in \texttt{R-TQBF}^{\texttt{loc}}$ *it holds that* $|w| = |x|$.
2. *(Verification that is local in both input and witness.) For every* $n \in \mathbb{N}$ *there exist* n *functions* $\{f_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\}_{i \in [n]}$ *such that the mapping* $(x,w,i) \mapsto f_i(x,w)$ *is computable in quasilinear time and linear space, and each* f_i *depends on only three variables, and* $(x,w) \in \texttt{R-TQBF}^{\texttt{loc}}$ *if and only if for all* $i \in [n]$ *it holds that* $f_i(x,w) = 1$.
3. *(Efficient reduction with quasilinear blow-up.) There exists a deterministic linear-space and quasilinear-time algorithm* A *that gets as input* $\varphi \in \{0,1\}^n$ *and outputs* $x = A(\varphi)$ *such that* $\varphi \in \texttt{TQBF}$ *if and only if* $x \in \texttt{TQBF}^{\texttt{loc}}$.

Proof. Consider a 3-SAT formula $\varphi \in \{0,1\}^n$ as an input to TQBF, and for simplicity assume that n is even (this assumption is insignificant for the proof and only simplifies the notation). By definition, we have that $\varphi \in \texttt{TQBF}$ if and only if

$$\exists w_1 \forall w_2 \exists w_3 .... \exists w_n \ \varphi(w_1, ..., w_n) = 1 \ .$$

Now, let M be a linear-space and quasilinear-time machine that gets as input (φ, w) and outputs $\varphi(w)$. We use an efficient Cook-Levin transformation of the computation of the machine M on inputs of length $2n$ to a 3-SAT formula, and deduce the following:[26] There exists a linear-space and quasilinear-time algorithm that, on input 1^n, constructs a 3-SAT formula $\Phi_n : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^{\texttt{ql}(n)} \to \{0,1\}$ of size $\texttt{ql}(n) = \tilde{O}(n)$ such that for any $(\varphi, w) \in \{0,1\}^n \times \{0,1\}^n$ it holds that $\varphi(w) = 1$ if and only if there exists a unique $w' \in \{0,1\}^{\texttt{ql}(n)}$ satisfying $\Phi_n(x,w,w') = 1$.

Now, using the formula Φ_n, note that $\varphi \in \{0,1\}^n$ is in TQBF if and only if

$$\exists w_1 \forall w_2 \exists w_3 ... \exists w_n \ \exists w'_1 \exists w'_2 ... \exists w'_{\texttt{ql}(n)} \ \Phi_n(\varphi, w, w') = 1 \ . \tag{4.3.1}$$

We slightly modify Φ_n in order to make the suffix of existential quantifiers in Eq. (4.3.1) alternate with universal quantifiers that are applied to dummy variables. (Specifically, for each $i \in [\texttt{ql}(n)]$, we rename w'_i to w'_{2i}, which effectively introduces a dummy variable before w'_i.) Denoting the modified formula by Φ'_n, we have that $\varphi \in \texttt{TQBF}$ if and only if

$$\exists w_1 \forall w_2 \exists w_3 ... \exists w_n \forall w'_1 \exists w'_2 \forall w'_3 ... \exists w'_{2\texttt{ql}(n)} \ \Phi'_n(\varphi, w, w') = 1 \ .$$

We define the relation $\texttt{R-TQBF}^{\texttt{loc}}$ to consist of all pairs (x,w) such that $x = (\varphi, 1^{2\texttt{ql}(|\varphi|)})$ and $w = (w^{(0)}, w^{(1)}) \in \{0,1\}^{|\varphi|} \times \{0,1\}^{2\texttt{ql}(|\varphi|)}$ and $\Phi'_{|\varphi|}(\varphi, w^{(0)}, w^{(1)}) = 1$. Indeed, in this case the corresponding set $\texttt{TQBF}^{\texttt{loc}}$ is defined by

$$\texttt{TQBF}^{\texttt{loc}} = \left\{ (\varphi, 1^{2\texttt{ql}(|\varphi|)}) : \exists w_1^{(0)} \forall w_2^{(0)} ... \exists w_{|\varphi|}^{(0)} \forall w_1^{(1)} \exists w_2^{(1)} ... \exists w_{2\texttt{ql}(|\varphi|)}^{(1)} \ \Phi'_{|\varphi|}(\varphi, w^{(0)}, w^{(1)}) = 1 \right\} \ .$$

[26] The algorithm transforms M into an oblivious machine [PF79; GS89], and then applies an efficient Cook-Levin transformation of the oblivious machine to a 3-SAT formula (see, e.g., [AB09, Sec 2.3.4]).

Note that, by definition, for every $(x,w) \in \text{R-TQBF}^{\text{loc}}$ we have that $|w| = |x|$. To see that $\text{R-TQBF}^{\text{loc}}$ can be tested by a conjunction of efficiently-computable local conditions, note that an n-bit input to TQBF^{loc} is of the form $(\varphi, 1^{2\text{ql}(|\varphi|)}) \in \{0,1\}^m \times \{1\}^{2\text{ql}(m)}$, and recall that Φ'_m is a 3-SAT formula of size $\text{ql}(m) < n$ that can be produced in linear space and quasilinear time from input 1^m. Also, TQBF^{loc} is computable in linear space, since on input $(\varphi, 1^{2\text{ql}(|\varphi|)})$ the number of variables that are quantified is $|\varphi| + 2\text{ql}(|\varphi|)$, and since $\Phi'_{|\varphi|}$ can be evaluated in space $O(|\varphi|)$. Lastly, TQBF trivially reduces to TQBF^{loc} by adding padding $\varphi \mapsto (\varphi, 1^{2\text{ql}(|\varphi|)})$. □

Arithmetic setting. For any $n \in \mathbb{N}$, let $\ell_0 = \ell_0(n) = \lfloor (\log n)^c \rfloor$, let $n' = \lceil n/\ell_0 \rceil$, let $\delta_0(n) = 2^{-n'}$, and let $\mathbb{F}$ be the field with $2^{5n'} = 1/\text{poly}(\delta_0(n))$ elements. Recall that a representation of such a field (i.e., an irreducible polynomial of degree $5n'$ over $\mathbb{F}_2$) can be found deterministically either in linear space (by a brute-force algorithm) or in time $\text{poly}(n') = \text{poly}(n)$ (by Shoup's [Sho90] algorithm).

Fix a bijection π between $\{0,1\}^{5n'}$ and $\mathbb{F}$ (i.e., π maps any string in $\{0,1\}^{5n'}$ to the bit-representation of the corresponding element in $\mathbb{F}$) such that both π and π^{-1} can be computed in polynomial time and linear space. Let $H \subset \mathbb{F}$ be the set of $2^{n'}$ elements that are represented (via π) by bit-strings with a prefix of n' arbitrary bits and a suffix of $4n'$ zeroes (i.e., $H = \left\{ \pi(z) : z = x0^{4n'}, x \in \{0,1\}^{n'} \right\} \subset \mathbb{F}$ such that $|H| = 2^{n'}$).[27]

We will consider polynomials $\mathbb{F}^{2\ell_0} \to \mathbb{F}$, and we think of the inputs to each such polynomial as of the form $(x,w) \in \mathbb{F}^{\ell_0} \times \mathbb{F}^{\ell_0}$. Note that, intuitively, x and w each represent about $5n$ bits of information. When x and w are elements in the subset $H^{\ell_0} \subset \mathbb{F}^{\ell_0}$, we think of them as a pair of n-bit strings that might belong to $\text{R-TQBF}^{\text{loc}}$.

Arithmetization of $\text{R-TQBF}^{\text{loc}}$. Our first step is to carefully arithmetize the relation $\text{R-TQBF}^{\text{loc}}$ within the arithmetic setting detailed above. We will mainly rely on the property that there is a "doubly-local" verification procedure for $\text{R-TQBF}^{\text{loc}}$.

Claim 4.3.9.2 (low-degree arithmetization)**.** *There exists a polynomial $P^{\text{TQBF}^{\text{loc}}} : \mathbb{F}^{2\ell_0} \to \mathbb{F}$ such that the following holds:*

1. *(Low-degree.) The degree of $P^{\text{TQBF}^{\text{loc}}}$ is at most $O(n \cdot 2^{n'})$.*
2. *(Arithmetizes $\text{R-TQBF}^{\text{loc}}$.) For every $(x,w) \in H^{\ell_0} \times H^{\ell_0}$ it holds that $P^{\text{TQBF}^{\text{loc}}}(x,w) = 1$ if $(x,w) \in \text{R-TQBF}^{\text{loc}}$, and $P^{\text{TQBF}^{\text{loc}}}(x,w) = 0$ otherwise.*
3. *(Efficiently-computable.) There exists a deterministic algorithm that gets as input $(x,w) \in \mathbb{F}^{2\ell_0}$, runs in time $\text{poly}(|\mathbb{F}|)$, and outputs $P^{\text{TQBF}^{\text{loc}}}(x,w) \in \mathbb{F}$. There also exists a deterministic linear-space algorithm with the same functionality.*

Proof. We first show a polynomial-time and linear-space algorithm that, given input 1^n, constructs a low-degree polynomial $P_0^{\text{TQBF}^{\text{loc}}} : \mathbb{F}^{2n' \cdot \ell_0} \to \mathbb{F}$ that satisfies the following:

[27] The specific choice of H as the image of $H_0 = \{x0^{4n'} : x \in \{0,1\}^{n'}\}$ under π is immaterial for our argument, as long as we can efficiently decide H_0 and enumerate over H_0.

For every $(x,w) \in \mathbb{F}_2^{2n' \cdot \ell_0}$ (i.e., when the input is a string of $2n' \cdot \ell_0 \geq 2n$ *bits*, and we interpret it as a pair $(x,w) \in \{0,1\}^{2n}$) it holds that $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}(x,w) = 1$ if $(x,w) \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}(x,w)$, and $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}(x,w) = 0$ otherwise.

To do so, recall that by Claim 4.3.9.1 we can construct in polynomial time and linear space a collection of n polynomials $\left\{f_i : \mathbb{F}_2^{2n' \cdot \ell_0} \to \mathbb{F}_2\right\}_{i \in [n]}$ such that for each $i \in [n]$ the polynomial f_i depends only on three variables in the input (x,w), and such that $(x,w) \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}$ if and only if for all $i \in [n]$ it holds that $f_i(x,w) = 1$. For each $i \in [n]$, let $p_i : \mathbb{F}^{2n' \cdot \ell_0} \to \mathbb{F}$ be the multilinear extension of f_i, which can be evaluated in time $\mathrm{poly}(n)$ and in linear space (since f_i depends only on three variables, and using Lagrange's interpolation formula and the fact that π is efficiently-computable). Then, the polynomial $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}$ is simply the multiplication of all the p_i's; that is, $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}(x,w) = \Pi_{i \in [n]} p_i(x,w)$. Note that $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}$ can indeed be evaluated in time $\mathrm{poly}(n)$ and in linear space, and that the degree of $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}$ is $O(n)$ (since each p_i is a multilinear polynomial in $O(1)$ variables).

Now, let $\pi_1^{(H)}, ..., \pi_{n'}^{(H)} : H \to \{0,1\}$ be the "projection" functions such that $\pi_i^{(H)}$ outputs the i^{th} bit in the bit-representation of its input according to π. Abusing notation, we let $\pi_1^{(H)}, ..., \pi_{n'}^{(H)} : \mathbb{F} \to \mathbb{F}$ be the low-degree extensions of the $\pi_i^{(H)}$'s, which are of degree at most $|H| - 1 < 2^{n'}$. Also, for every $\sigma \in \mathbb{F}$, we denote by $\pi^{(H)}(\sigma)$ the string $\pi_1^{(H)}(\sigma), ..., \pi_{n'}^{(H)}(\sigma) \in \mathbb{F}^{n'}$. Note that the mapping of $\sigma \in \mathbb{F}$ to $\pi^{(H)}(\sigma) \in \mathbb{F}^{n'}$ can be computed in time $\mathrm{poly}(|H|) = \mathrm{poly}(|\mathbb{F}|)$ and in linear space (again just using Lagrange's interpolation formula and the fact that π is efficiently-computable).

Finally, we define the polynomial $P^{\mathtt{TQBF}^{\mathtt{loc}}} : \mathbb{F}^{2\ell_0} \to \mathbb{F}$. Intuitively, for $(x,w) \in H^{\ell_0} \times H^{\ell_0}$, the polynomial $P^{\mathtt{TQBF}^{\mathtt{loc}}}$ first uses the $\pi_i^{(H)}$'s to compute the bit-projections of x and w, which are each of length $n' \cdot \ell_0$, and then evaluates the polynomial $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}$ on these $2n' \cdot \ell_0$ bit-projections. More formally, for every $(x,w) \in \mathbb{F}^{2\ell_0}$ we define

$$P^{\mathtt{TQBF}^{\mathtt{loc}}}(x,w) = P_0^{\mathtt{TQBF}^{\mathtt{loc}}}\left(\pi^{(H)}(x_1), ..., \pi^{(H)}(x_{\ell_0}), \pi^{(H)}(w_1), ..., \pi^{(H)}(w_{\ell_0})\right).$$

The first item in the claim follows since for every $i \in [n']$ the degree of $\pi_i^{(H)}$ is less than $2^{n'}$, and since $\deg(P_0^{\mathtt{TQBF}^{\mathtt{loc}}}) = O(n)$. The second item in the claim follows immediately from the definition of $P^{\mathtt{TQBF}^{\mathtt{loc}}}$. And the third item in the claim follows since $\pi^{(H)}$ can be computed in time $\mathrm{poly}(|\mathbb{F}|)$ and in linear space, and since $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}$ can be constructed and evaluated in polynomial time and in linear space. (The two different algorithms are since we need to find an irreducible polynomial, which can be done either in linear space or in time $\mathrm{poly}(n) < \mathrm{poly}(|\mathbb{F}|)$.) □

Constructing a "downward self-reducible" collection of low-degree polynomials. Our goal now is to define a collection of $O(\ell_0^2)$ polynomials $\left\{P_{n,i} : \mathbb{F}^{2\ell_0} \to \mathbb{F}\right\}_{i \in [O(\ell_0^2)]}$ such that the polynomials are of low degree, and $P_{n,1}$ essentially computes $\mathtt{TQBF}^{\mathtt{loc}}$,

and computing $P_{n,i}$ can be reduced in time $\text{poly}(1/\delta_0(n))$ to computing $P_{n,i+1}$. The collection and its properties are detailed in the following claim:

Claim 4.3.9.3. *There exists a collection of $\bar{\ell_0} = \ell_0(2\ell_0+1)+1$ polynomials, denoted $\left\{P_{n,i} : \mathbb{F}^{2\ell_0} \to \mathbb{F}\right\}_{i\in[\bar{\ell_0}]}$, that satisfies the following:*

1. (Low degree:) *For every $i \in [\bar{\ell_0}]$, the degree of $P_{n,i}$ is at most $O(n \cdot \ell_0 \cdot 2^{2n'})$.*
2. ($P_{n,1}$ computes $\texttt{TQBF}^{\texttt{loc}}$ on H-inputs:) *For any $(x,w) \in H^{\ell_0} \times H^{\ell_0}$ it holds that $P_{n,1}(x,w) = 1$ if $x \in \texttt{TQBF}^{\texttt{loc}}$, and $P_{n,1}(x,w) = 0$ if $x \notin \texttt{TQBF}^{\texttt{loc}}$. (Regardless of w.)*
3. ("Forward" self-reducible:) *For every $i \in [\bar{\ell_0}]$ it holds that $P_{n,i}$ can be computed in time $\text{poly}(2^{n'})$ when given oracle access to $P_{n,i+1}$.*
4. (Efficiently-computable:) *The polynomial $P_{n,\bar{\ell_0}}$ can be computed in time $\text{poly}(2^{n'})$. Moreover, for every $i \in [\bar{\ell_0}]$ it holds that $P_{n,i}$ can be computed in space $O(n \cdot \bar{\ell_0})$.*

Proof. For simplicity of notation, assume throughout the proof that n' is even. Towards defining the collection of polynomials, we first define two operators on functions $p : \mathbb{F}^{2\ell_0} \to \mathbb{F}$. Loosely speaking, the first operator corresponds to n' alternating quantification steps in the $\mathcal{IP} = \mathcal{PSPACE}$ proof (i.e., n' steps of alternately quantifying the next variable either by $\exists$ or by $\forall$), and the second operator roughly corresponds to a linearization step that is simultaneously applied to n' variables. In both cases, the n' variables that we consider are the bits in the representation of a single element in the second input to p.

Quantifications operator: Let $i \in [\ell_0]$. Loosely speaking, $\texttt{Quant}^{(i)}(p)$ causes p to ignore the i^{th} variable of its second input, and instead consider alternating quantification steps applied to the bits that represent this variable. To do this, we define a sequence of functions such that the first function replaces the i^{th} variable in the second input for p by a dummy variable in H, and each subsequent function corresponds to a quantification step applied to a single bit in the representation of this dummy variable.

Formally, we recursively define $n'+1$ functions $\texttt{Quant}^{(i,0)}, ..., \texttt{Quant}^{(i,n')} = \texttt{Quant}^{(i)}(p)$ such that for $j \in \{0, ..., n'\}$ it holds that $\texttt{Quant}^{(i,j)}(p)$ is a function $\mathbb{F}^{2\ell_0} \times \{0,1\}^{n'-j} \to \mathbb{F}$. The function $\texttt{Quant}^{(i,0)}(p)$ gets as input $(x,w) \in \mathbb{F}^{2\ell_0}$ and $\sigma \in \{0,1\}^{n'}$, ignores the i^{th} element of w, and outputs $\texttt{Quant}^{(i,0)}(x,w,\sigma) = p(x, w_1...w_{i-1}\pi(\sigma 0^{4n'}))$. Then, for $j \in [n']$, if j is odd then we define

$$\texttt{Quant}^{(i,j)}(p)(x,w,\sigma_1...\sigma_{n'-j}) = 1 - \left(\prod_{z\in\{0,1\}} \left(1 - \texttt{Quant}^{(i,j-1)}(p)(x,w,\sigma_1,...,\sigma_{n'-j}z)\right)\right) ,$$

and if j is even then we define

$$\texttt{Quant}^{(i,j)}(p)(x,\sigma_1,...,\sigma_{n'-j}) = \prod_{z\in\{0,1\}} \texttt{Quant}^{(i,j-1)}(p)(x,w,\sigma_1...\sigma_{n'-j}z) .$$

Note that the function $\texttt{Quant}^{(i)}(p)$ can be evaluated at any input in linear space with oracle access to p (since each $\texttt{Quant}^{(i,j)}(p)$ can be evaluated in linear space with

oracle access to $\texttt{Quant}^{(i,j-1)}(p)$). Also observe the following property of $\texttt{Quant}^{(i)}(p)$, which follows immediately from the definition:

Fact 4.3.9.3.1. *If for some $x \in H^{\ell_0}$ and any $w \in H^{\ell_0}$ it holds that $p(x,w) \in \{0,1\}$, then for the same x and any $w \in H^{\ell_0}$ it holds that $\texttt{Quant}^{(i)}(p)(x,w) = 1$ if $\exists\sigma_1\forall\sigma_2\exists\sigma_3...\forall\sigma_{n'}$ such that $p(x,w_1...w_{i-1}\pi(\sigma_1...\sigma_{n'}0^{4n'})w_{i+1}...w_{\ell_0}) = 1$, and $\texttt{Quant}^{(i)}(p)(x,w) = 0$ otherwise.*

Degree-reduction operator: For every fixed $z \in H$, let $I_z : H \to \{0,1\}$ be the indicator function of whether the input equals z, and let $\bar{I}_z : \mathbb{F} \to \mathbb{F}$ be the low-degree extension of I_z, which is of degree at most $|H|-1$ (i.e., $\bar{I}_z(x) = \prod_{h\in H\setminus\{z\}} \frac{x-h}{z-h}$). Then, for any $i \in [\ell_0]$, we define

$$\texttt{DegRed}^{(i)}(p)(x,w) = \sum_{z\in H} \bar{I}_z(x_i)\cdot p(x_1...x_{i-1}zx_{i+1}...x_{\ell_0},w)\ ,$$

and similarly for $i \in [2\ell_0]$ we denote $i' = i-\ell_0$ and define

$$\texttt{DegRed}^{(i)}(p)(x,w) = \sum_{z\in H} \bar{I}_z(w_{i'})\cdot p(x,w_1...w_{i'-1}zw_{i'+1}...w_{\ell_0})\ .$$

Similarly to the operator $\texttt{Quant}^{(i)}$, note that the function $\texttt{DegRed}^{(i)}(p)$ can be evaluated at any input in linear space with oracle access to p. Also, the definition of the operator $\texttt{DegRed}^{(i)}$ implies that:

Fact 4.3.9.3.2. *For $i \in [2\ell_0]$, let v be the variable whose degree $\texttt{DegRed}^{(i)}$ reduces (i.e., $v = x_i$ if $i \in [\ell_0]$ and $v = w_{i'} = w_{i-\ell_0}$ if $i \in [2\ell_0]$). Then, the individual degree of v in $\texttt{DegRed}^{(i)}(p)$ is $|H|-1$, and the individual degree of any other input variable to $\texttt{DegRed}^{(i)}(p)$ remains the same as in p. Moreover, for every $(x,w) \in \mathbb{F}^{\ell_0}\times\mathbb{F}^{\ell_0}$, if the input (x,w) assigns the variable v to a value in H, then $\texttt{DegRed}^{(i)}(p)(x,w) = p(x,w)$.*

Composing the operators: We will be particularly interested in what happens when we first apply the quantifications operator to some variable $i \in [\ell_0]$, and then apply the degree-reduction operator to all variables, sequentially. A useful property of this operation is detailed in the following claim:

Claim 4.3.9.3.3. *Let $p : \mathbb{F}^{2\ell_0} \to \mathbb{F}$ and $x \in H^{\ell_0}$ such that for any $w \in H^{\ell_0}$ it holds that $p(x,w) \in \{0,1\}$. For $i \in [\ell_0]$, let $p' : \mathbb{F}^{2\ell_0} \to \mathbb{F}$ be the function that is obtained by first applying $\texttt{Quant}^{(i)}$ to p, then applying $\texttt{DegRed}^{(j)}$ for each $j = 1,...,2\ell_0$. Then, for any $w' \in H^{\ell_0}$ we have that $p'(x,w') = 1$ if $\exists\sigma_1\forall\sigma_2\exists\sigma_3...\forall\sigma_{n'} : p(x,w'_1...w'_{i-1}\pi(\sigma_1...\sigma_{n'})w'_{i+1}...w'_{\ell_0}) = 1$, and $p'(x,w') = 0$ otherwise.*

Proof. Fix any $w' \in H^{\ell_0}$. By Fact 4.3.9.3.1, and relying on the hypothesis that for any $w \in H^{\ell_0}$ we have that $p(x,w) \in \{0,1\}$, it follows that $\texttt{Quant}^{(i)}(p)(x,w') = 1$ if $\exists\sigma_1\forall\sigma_2\exists\sigma_3...\forall\sigma_{n'} : p(x,w'_1...w'_{i-1}\pi(\sigma_1...\sigma_{n'})w'_{i+1}...w'_{\ell_0}) = 1$ and that $\texttt{Quant}^{(i)}(p)(x,w') = 0$ otherwise. Now, let $p^{(0)} = \texttt{Quant}^{(i)}(p)$, and for every $j \in [2\ell_0]$ recursively define $p^{(j)} = \texttt{DegRed}^{(j)}(p^{(j-1)})$. By the "moreover" part of Fact 4.3.9.3.2, and since $(x,w') \in H^{\ell_0}\times H^{\ell_0}$, for every $j \in [2\ell_0]$ we have that $p^{(j)}(x,w') = p^{(j-1)}(x,w')$, and hence $p'(x,w') = \texttt{Quant}^{(i)}(x,w')$. □

Defining the collection of polynomials: Let us now define the collection of $\bar{\ell_0} = \ell_0(2\ell_0+1)+1$ polynomials. We first define $P_{n,\ell_0(2\ell_0+1)+1}(x,w) = P^{\mathtt{TQBF}^{\mathtt{loc}}}(x,w)$. Then, we recursively construct the collection in ℓ_0 blocks such that each block consists of $2\ell_0+1$ polynomials. The base case will be block $i=\ell_0$, and we will decrease i down to 1. Loosely speaking, in each block $i \in [\ell_0]$, starting from the last polynomial in the previous block, we first apply a quantification operator to the i^{th} variable of the second input w, and then apply $2\ell_0$ linearization operators, one for each variable in the inputs (x,w). Specifically, for the i^{th} block, we define the first polynomial by $P_{n,i(2\ell_0+1)}(x,w) = \mathtt{Quant}^{(i)}(P_{n,i(2\ell_0+1)+1})(x,w)$; and for each $j=1,...,2\ell_0$, we define $P_{n,i(2\ell_0+1)-j}(x,w) = \mathtt{DegRed}^{(j)}(P_{n,i(2\ell_0+1)-j+1})(x,w)$.

Note that the claimed Property (3) of the collection holds immediately from our definition. To see that Property (4) also holds, note that the first part (regarding $P_{n,\bar{\ell_0}}$) holds by Claim 4.3.9.2; and for the "moreover" part, recall (by the properties of the operators $\mathtt{Quant}^{(i)}$ and $\mathtt{DegRed}^{(i)}$ that were mentioned above) that each polynomial $P_{n,k}$ in the collection can be computed in linear space when given access to the "previous" polynomial $P_{n,k-1}$, and also that we can compute the "first" polynomial $P_{n,\ell_0(2\ell_0+1)+1}$ in linear space (since this polynomial is just $P^{\mathtt{TQBF}^{\mathtt{loc}}}$, and relying on Claim 4.3.9.2). Using a suitable composition lemma for space-bounded computation (see, e.g., [Gol08, Lem. 5.2]), we can compute any polynomial in the collection in space $O(n \cdot \bar{\ell_0})$.

We now prove Property (1), which asserts that all the polynomials in the collection are of degree at most $O(n \cdot \ell_0 \cdot 2^{2n'})$. We prove this by induction on the blocks, going from $i=\ell_0$ down to $i=1$, while maintaining the invariant that the "last" polynomial in the previous block $i+1$ (i.e., the polynomial $P_{n,i(2\ell_0+1)+1}$) is of degree at most $O(n \cdot 2^{n'})$. For the base case $i=\ell_0$ the invariant holds by our definition that $P_{n,\ell_0(2\ell_0+1)+1} = P^{\mathtt{TQBF}^{\mathtt{loc}}}$ and by Claim 4.3.9.2. Now, for every $i=\ell_0,...,1$, note that the first polynomial $P_{n,i(2\ell_0+1)}$ in the block is of degree at most $2^{n'} \cdot \deg(P_{n,i(\ell_0+1)+1}) = O(n \cdot 2^{2n'})$ (i.e., the quantifications operator induces a degree blow-up of $2^{n'}$), and in particular the individual degrees of all variables of $P_{n,i(2\ell_0+1)}$ are upper-bounded by this expression. Then, in the subsequent $2\ell_0$ polynomials in the block, we reduce the individual degrees of the variables (sequentially) until all individual degrees are at most $|H|-1 < 2^{n'}$ (this relies on Fact 4.3.9.3.2). Thus, the degree of the last polynomial in the block (i.e., of $P_{n,(i-1)(2\ell_0+1)+1}$) is at most $2\ell_0 \cdot 2^{n'} < n \cdot 2^{n'}$, and the invariant is indeed maintained.

Finally, to see that Property (2) holds, fix any $(x,w) \in H^{\ell_0} \times H^{\ell_0}$. Our goal is to show that $P_{n,1}(x,w)=1$ if $x \in \mathtt{TQBF}^{\mathtt{loc}}$ and $P_{n,1}(x,w)=0$ otherwise (regardless of w). To do so, recall that $P_{n,\bar{\ell_0}} = P^{\mathtt{TQBF}^{\mathtt{loc}}}$, and hence for any $w' \in H^{\ell_0}$ it holds that $P_{n,\bar{\ell_0}}(x,w')=1$ if $(x,w') \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}$ and $P_{n,\bar{\ell_0}}(x,w')=0$ otherwise. Note that the last polynomial in block $i=\ell_0$ (i.e., the polynomial $P_{n,\ell_0(2\ell_0+1)-2\ell_0}$) is obtained by applying $\mathtt{Quant}^{(\ell_0)}$ to $P_{n,\bar{\ell_0}}$ and then applying $\mathtt{DegRed}^{(j)}$ for each $j=1,...,2\ell_0$. Using Claim 4.3.9.3.3, for any $w' \in H^{\ell_0}$, when this polynomial is given input (x,w'), it outputs the value 1 if $\exists\sigma_1\forall\sigma_2\exists\sigma_3...\forall\sigma_{n'}(x,w'_1...w'_{\ell_0-1}\pi(\sigma_1...\sigma_{n'})) \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}$, and outputs 0 otherwise. By repeatedly using Claim 4.3.9.3.3 for the last polynomial in each

block $i = \ell_0 - 1, ..., 1$, we have that $P_{n,1}(x, w) = 1$ if $\exists \sigma_1^{(1)} \forall \sigma_2^{(1)} ... \forall \sigma_{n'}^{(1)} ... \exists \sigma_1^{(\ell_0)} ... \forall \sigma_{n'}^{(\ell_0)}$: $(x, w') \in \text{R-TQBF}^{\text{loc}}$, where $w' = (\pi(\sigma_1^{(1)} ... \sigma_{n'}^{(1)}), ..., \pi(\sigma_1^{(\ell_0)} ... \sigma_{n'}^{(\ell_0)}))$; and $P_{n,1}(x, w) = 0$ otherwise. In other words, we have that $P_{n,1}(x, w) = 1$ if $x \in \text{TQBF}^{\text{loc}}$ and $P_{n,1}(x, w) = 0$ otherwise, as we wanted. □

Combining the polynomials into a Boolean function. Intuitively, the polynomials in our collection are already downward self-reducible (where "downward" here means that $P_{n,i}$ is reducible to $P_{n,i+1}$) and sample-aided worst-case to average-case reducible (since the polynomials have low degree, and relying on Proposition 4.3.19). Our goal now is simply to "combine" these polynomials into a single Boolean function $f^{\text{ws}} : \{0,1\}^* \to \{0,1\}^*$ that will be δ-well-structured.

For every $n \in \mathbb{N}$, we define a corresponding interval of input lengths $I_n = [N, N + \bar{\ell}_0 - 1]$, where $N = 10n' \cdot \ell_0 + 11n \cdot \bar{\ell}_0 = O(n \cdot \bar{\ell}_0)$. Then, for every $i \in \{0, ..., \bar{\ell}_0 - 1\}$, we define f^{ws} on input length $N + i$ such that it computes (a Boolean version of) $P_{n,\bar{\ell}_0 - i}$. Specifically, $f^{\text{ws}} : \{0,1\}^{N+i} \to \{0,1\}^{N+i}$ considers only the first $10n' \cdot \ell_0 = 2\ell_0 \cdot \log(|\mathbb{F}|) = O(n)$ bits of its input, maps these bits to $(x, w) \in \mathbb{F}^{2\ell_0}$ using π, computes $P_{n,\bar{\ell}_0 - i}(x, w)$, and outputs the bit-representation of $P_{n,\bar{\ell}_0 - i}(x, w)$ (using π^{-1}), padded to the appropriate length $N + i$. On input lengths that do not belong to any interval I_n for $n \in \mathbb{N}$, we define f^{ws} in some fixed trivial way (e.g., as the identity function).

A straightforward calculation shows that the intervals $\{I_n\}_{n \in \mathbb{N}}$ are disjoint, and thus f^{ws} is well-defined.[28] In addition, since the input length to f^{ws} is $N = O(n \cdot \bar{\ell}_0)$ and each polynomial in the collection is computable in space $O(n \cdot \bar{\ell}_0)$, it follows that f^{ws} is computable in linear space. To see that TQBF reduces to f^{ws}, recall that by Claim 4.3.9.1 we can reduce TQBF to TQBF^{loc} in time $n \cdot (\log n)^r$ (for some universal constant $r \in \mathbb{N}$); and note that we can then further reduce TQBF^{loc} to f^{ws} by mapping any $x \in \{0,1\}^n$ to an $(N + \bar{\ell}_0 - 1)$-bit input of the form (x, w, p), where w is an arbitrary string and p is padding. (This is since f^{ws} on inputs of length $N + \bar{\ell}_0 - 1$ essentially computes $P_{n,1}$.) This reduction is computable in deterministic time $n \cdot \log(n)^{r+2c+1}$.

We now want to show that f^{ws} is downward self-reducible in time $\text{poly}(1/\delta)$ and in $O((\log N)^{2c})$ steps, where $\delta(N) = 2^{N/(\log N)^{3c}}$ and N denotes the input length. To see this, first note that given input length $N \in \mathbb{N}$ we can find in polynomial time an input length n such that $N \in I_n$, if such n exists. If such n does not exist, then the function is defined trivially on input length n and can be computed in polynomial time. Otherwise, let $N_0 \leq N$ be the smallest input length in I_n (i.e., $N_0 = 10 \lceil n/\ell_0(n) \rceil \cdot \ell_0(n) + 11n \cdot \bar{\ell}_0(n)$), and denote $N = N_0 + i$, for some $i \in \{0, ..., \bar{\ell}_0(n) - 1\}$. Note that f_N^{ws} corresponds to the polynomial $P_{n,\bar{\ell}_0(n)-i}$, and f_{N-1}^{ws} corresponds to the polynomial $P_{n,\bar{\ell}_0(n)-(i-1)}$. By Claim 4.3.9.3, the former can be computed in time $\text{poly}(2^{n'}) = \text{poly}(2^{n/(\log n)^c}) = \text{poly}(2^{N/(\log N)^{3c}})$ with oracle access to the latter. Lastly, recall that

[28] This is the case since the largest input length in I_n is $10 \lceil n/\ell_0(n) \rceil \cdot \ell_0(n) + 11n \cdot \bar{\ell}_0(n) + (\bar{\ell}_0(n) - 1) < 10n + 10\ell_0(n) + (11n + 1) \cdot \bar{\ell}_0(n) - 1 < 10n + 11(n + 1) \cdot \bar{\ell}_0(n) - 1$, whereas the smallest input length in I_{n+1} is $10 \lceil (n+1)/\ell_0(n+1) \rceil \cdot \ell_0(n+1) + 11(n+1) \cdot \bar{\ell}_0(n+1) \geq 10n + 11(n+1)\bar{\ell}_0(n+1) + 10$.

$|I_n| = \bar{\ell}_0(n) < O(\log N)^{2c}$ and that $f^{\mathtt{ws}}_{N_0}$ corresponds to $P_{n,\ell_0(n)}$, which can be computed in time $\mathrm{poly}(2^{n'})$; hence, there exists an input length $N_0 \geq N - O((\log N)^{2c})$ such that $f^{\mathtt{ws}}_{N_0}$ can be computed in time $\mathrm{poly}(2^{n'}) < \mathrm{poly}(1/\delta(N_0))$.

To see that $f^{\mathtt{ws}}$ is sample-aided worst-case to δ-average-case reducible, first note that computing $f^{\mathtt{ws}}$ on any input length N on which it is not trivially defined is equivalent (up to a polynomial factor in the runtime) to computing a polynomial $\mathbb{F}^{2\ell_0(n)} \to \mathbb{F}$ of degree $d = O(\mathrm{poly}(n) \cdot 2^{2n'})$ in a field of size $q = |\mathbb{F}| = 2^{5n'}$, where $n < N/(\log N)^{2c}$ and $n' = \lceil n/\ell_0(n) \rceil$. [29] We use Proposition 4.3.19 with parameter $\rho(\log(|\mathbb{F}^{2\ell_0(n)}|)) = \delta_0(n) < \delta(N)$, and note that its hypothesis $\delta_0(n) \geq 10 \cdot \sqrt{d/|\mathbb{F}|}$ is satisfied since we chose $|\mathbb{F}| = \mathrm{poly}(1/\delta_0(n))$ to be sufficiently large. ■

4.3.4 PRGs for uniform circuits with almost-exponential stretch

Let $\delta(n) = 2^{-n/\mathrm{polylog}(n)}$. The following proposition asserts that if there exists a function that is both δ-well-structured and "hard" for probabilistic algorithms that run in time $2^{n/\mathrm{polylog}(n)}$, then there exists an i.o.-PRG for uniform circuits with almost-exponential stretch. That is:

Proposition 4.3.10 (almost-exponential hardness of a well-structured function ⇒ PRG for uniform circuits with almost-exponential stretch)**.** *Assume that for some constant $c \in \mathbb{N}$ and for $\delta(n) = 2^{-n/\log(n)^{c+1}}$ there exists a δ-well-structured function that can be computed in linear space but cannot be computed by probabilistic algorithms that run in time $2^{n/\log(n)^c}$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\mathrm{loglog}(n)^k}$ there exists a $(1/t)$-i.o.-PRG for $(t, \log(t))$-uniform circuits that has seed length $\tilde{O}(\log(n))$ and is computable in time $n^{\mathrm{polyloglog}(n)}$.*

Proposition 4.3.10 follows as an immediate corollary of the following lemma. Loosely speaking, the lemma asserts that for any δ-well-structured function $f^{\mathtt{ws}}$, there exists a corresponding PRG with almost-exponential stretch such that a uniform algorithm that distinguishes the output of the PRG from uniform yields a uniform probabilistic algorithm that computes $f^{\mathtt{ws}}$. Moreover, the lemma provides a "point-wise" statement: For any $n \in \mathbb{N}$, a distinguisher on a small number (i.e., $\mathrm{polyloglog}(n)$) of input lengths in a small interval around n yields a uniform algorithm for $f^{\mathtt{ws}}$ on input length $\tilde{O}(\log(n)$. We will later use this "point-wise" property of the lemma to extend Proposition 4.3.10 to "almost everywhere" versions (see Propositions 4.3.13 and 4.3.14).

In the following statement we consider three algorithms: The pseudorandom generator G; a potential distinguisher for the PRG, denoted A; and an algorithm F for the "hard" function $f^{\mathtt{ws}}$. Loosely speaking, the lemma asserts that for any $n \in \mathbb{N}$, if G is *not* pseudorandom for A on a every input length in a small set of input lengths

[29] The only potential issue here is that the Boolean function is actually a "padded" version of the function that corresponds to polynomial: It is not immediate that if there exists an algorithm that computes the Boolean function correctly on $\epsilon > 0$ of the n-bit inputs, then there exists an algorithm that computes the polynomial correctly on the same fraction $\epsilon > 0$ of the $m = \log(|\mathbb{F}^{2\ell_0}|)$-bit inputs. However, the latter assertion holds in our case since we are interested in *probabilistic* algorithms.

surrounding n, then F computes $f^{\mathtt{ws}}$ on input length $\ell(n) = \tilde{O}(\log(n))$. We will first fix a constant c that determines the target running time of F (i.e., running time $t_F(\ell) = 2^{\ell/\log(\ell)^c}$), and the other parameters (e.g., the parameters of the well-structured function, and the seed length of the PRG) will depend on c. Specifically:

Lemma 4.3.11 (distinguishing a PRG based on $f^{\mathtt{ws}} \Rightarrow$ computing $f^{\mathtt{ws}}$)**.** *Let $c \in \mathbb{N}$ be an arbitrary constant, let $\delta(n) = 2^{-n/\log(n)^{c+1}}$, and let $s : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function such that $s(n) \leq n/2$ for all $n \in \mathbb{N}$. Let $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ be a (δ, s)-well-structured function that is computable in linear space, let $t(n) = n^{\mathrm{loglog}(n)^k}$ for some constant $k \in \mathbb{N}$, and let $\ell(n) = \lceil \log(n) \cdot (\mathrm{loglog} n)^b \rceil$ for a sufficiently large constant $b \in \mathbb{N}$. Then, there exist two objects that satisfy the property detailed below:*

1. *(Pseudorandom generator). An algorithm G_0 that gets as input 1^n and a random seed of length $\ell_G(n) = \tilde{O}(\ell(n))$, runs in time $n^{\mathrm{polyloglog}(n)}$, and outputs a string of length n.*
2. *(Mapping of any input length to a small set of surrounding input lengths). A polynomial-time computable mapping of any unary string 1^n to a set $S_n \subset [n, n^2]$ of size $|S_n| = s(\tilde{O}(\log(n)))$, where $a \in \mathbb{N}$ is a sufficiently large constant that depends on k.*

The property that the foregoing objects satisfy is the following. For every probabilistic time-t algorithm A that uses $\log(t)$ bits of non-uniform advice there exists a corresponding probabilistic algorithm F that runs in time $t_F(\ell) = 2^{\ell/\log(\ell)^c}$ such that for any $n \in \mathbb{N}$ we have that: If for every $m \in S_n$ it holds that $G_0(1^m, \mathbf{u}_{\ell_{G_0}(m)})$ is not *$(1/t(m))$-pseudorandom for A, then F computes $f^{\mathtt{ws}}$ on strings of length $\ell(n)$.*

Moreover, for any function $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ such that $\mathtt{str}(n) \leq n$, the above property holds if we replace G_0 by the algorithm G that computes G_0 and truncates the output to length $\mathtt{str}(n)$ (i.e., $G(1^n, z) = G_0(1^n, z)_1, ..., G_0(1^n, z)_{\mathtt{str}(n)}$).

Observe that Proposition 4.3.10 indeed follows as a contra-positive of Lemma 4.3.11 (with $\mathtt{str}$ being the identity function, which means that $G = G_0$): If every probabilistic algorithm F that gets an ℓ-bit input and runs in time $2^{\ell/\log(\ell)^c}$ fails to compute $f^{\mathtt{ws}}$ infinitely-often, then for every corresponding time-t algorithm A there exists an infinite set of inputs on which G is pseudorandom for A.

Proof of Lemma 4.3.11. For any p, s, δ, k, t, and $f^{\mathtt{ws}}$ that satisfy our hypothesis, let $f^{\mathtt{GL}(\mathtt{ws})} : \{0,1\}^* \to \{0,1\}$ be defined as follows: For any $(x, r) \in \{0,1\}^n \times \{0,1\}^n$ we let $f^{\mathtt{GL}(\mathtt{ws})}(x, r) = \sum_{i \in [n]} f^{\mathtt{ws}}(x)_i \cdot r_i$, where the arithmetic is over $\mathbb{F}_2$.[30] (We use the notation $f^{\mathtt{GL}(\mathtt{ws})}$ since we will use the algorithm of Goldreich and Levin [GL89] to transform a circuit that agrees with $f^{\mathtt{GL}(\mathtt{ws})}$ on $1/2 + \epsilon$ of the inputs into a circuit that computes $f^{\mathtt{ws}}$ on $\mathrm{poly}(\epsilon)$ of the inputs.)

The algorithm G_0 is the Nisan-Wigderson generator, instantiated with $f^{\mathtt{GL}(\mathtt{ws})}$ as the hard function and with combinatorial designs such that the output length is n, the sets in the design are of size $\ell(n) = \lceil \log(n) \cdot (\mathrm{loglog} n)^b \rceil$ (where b is a sufficiently large

[30] On odd input lengths the function $f^{\mathtt{GL}(\mathtt{ws})}$ is defined by ignoring the last input bit; that is, $f^{\mathtt{GL}(\mathtt{ws})}(x, r\sigma) = f^{\mathtt{GL}(\mathtt{ws})}(x, r)$, where $|x| = |r|$ and $|\sigma| = 1$.

constant that depends on k), the seed length is $\ell_G(n) = \tilde{O}(\ell(n)) = \tilde{O}(\log(n))$, and the size of the intersection between any two sets in the design is $\gamma \cdot \log(n)$ where $\gamma > 0$ is a sufficiently small constant (see, e.g., [Vad12, Prob 3.2] for a suitable construction). Since $f^{\mathtt{ws}}$ is computable in linear space, the function $f^{\mathtt{GL}(\mathtt{ws})}(x, r)$ is computable in time $n^{\mathrm{polyloglog}(n)}$, and hence G_0 is computable in time $n^{\mathrm{polyloglog}(n)}$.

Fix a mapping of any 1^n to a corresponding set S_n that will be defined in a moment (and depends only on the parameters up to this point). Now, let $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ be any polynomial-time computable function satisfying $\mathtt{str}(n) \leq n$, and let G be such that $G(1^n, s) = G_0(1^n, s)_{1,\ldots,\mathtt{str}(n)}$. For $t(n) = n^{\mathrm{loglog}(n)^k}$, let A be a probabilistic algorithm that gets input 1^n and $\log(t(n))$ bits of non-uniform advice and runs in time $t(n)$. For any sufficiently large $n \in \mathbb{N}$, we assume that for every $m \in S_n$, when A is given input $1^{\mathtt{str}(m)}$ and corresponding "good" advice, with probability at least $1/t(m)$ it outputs a circuit $D_{\mathtt{str}(m)} : \{0,1\}^{\mathtt{str}(m)} \to \{0,1\}$ that $(1/t(m))$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform. Under this assumption, we will construct a probabilistic algorithm that gets input $1^{\ell(n)}$, runs in time $\mathrm{poly}(1/\delta(\ell(n)) = 2^{O(\ell(n)/\log(\ell(n))^{c+1})}$, and with high probability outputs a circuit $\{0,1\}^{\ell(n)} \to \{0,1\}$ that correctly computes $f^{\mathtt{ws}}$ on $\ell(n)$-bit inputs. This implies that a probabilistic algorithm can decide $f^{\mathtt{ws}}$ on $\{0,1\}^{\ell(n)}$ in time at most $2^{\ell(n)/\log(\ell(n))^c}$.

Towards presenting the construction, denote $\ell'(n) = \ell(n)/\log(\ell(n))^{c+1}$, and fix a sufficiently small universal constant $\epsilon > 0$ (which depends only on universal constants from arguments in [NW94; IW98]). We assume that $\ell(n)$ is sufficiently large such that $t(n) = n^{\mathrm{loglog}(n)^k} \leq 2^{\epsilon \cdot \ell'(n)}$. Recall that, since $f^{\mathtt{ws}}$ is downward self-reducible in s steps, there exists an input length $\ell_0(n) \geq \ell(n) - s(\ell(n))$ such that $f^{\mathtt{ws}}_{\ell_0(n)}$ is computable in time $\mathrm{poly}(1/\delta(\ell_0(n)))$. For $L_n = \{\ell_0(n), \ldots, \ell(n)\}$, we define $S_n = \{\ell^{-1}(2i) : i \in L_n\}$. Note that indeed $|S_n| \leq s(\ell(n)) = s(\tilde{O}(\log(n)))$; and relying on the fact that $s(\ell(n)) \leq \ell(n)/2$, we have that $S_n \subset [n_0, n_1]$ where $n_0 = \ell^{-1}(2\ell_0) \geq \ell^{-1}(\ell(n)) = n$ and $n_1 = \ell^{-1}(2\ell(n)) < n^2$. Lastly, note that S_n does not depend on the function $\mathtt{str}$ or on the algorithm A.

Our first step is to show that (loosely speaking) under our assumption about A, for any $m \in S_n$ we can efficiently construct (using only a small amount of non-uniform advice) a circuit that computes $f^{\mathtt{GL}(\mathtt{ws})}$ on noticeably more than half of the inputs of length $\ell(m)$. The proof of this claim is a variation on the standard efficient transformation of distinguishers for the Nisan-Wigderson PRG to approximating circuits for the "hard" function, from [IW98] (following [NW94]).

Claim 4.3.11.1. *There exists a probabilistic algorithm such that for any $m \in S_n$, when the algorithm is given input $1^{\ell(m)}$, and oracle access to $f^{\mathtt{GL}(\mathtt{ws})}$ on $\ell(m)$-bit inputs, and $2\epsilon \cdot \ell'(m)$ bits of non-uniform advice, the algorithm runs in time $2^{\ell'(m)}$ and with probability more than $2^{-\ell'(m)}$ outputs a circuit $\{0,1\}^{\ell(m)} \to \{0,1\}$ that computes $f^{\mathtt{GL}(\mathtt{ws})}$ correctly on more than $1/2 + 2^{-\ell'(m)}$ of the inputs.*

Proof. Let $\ell = \ell(m)$, let $\ell' = \ell'(m)$, and let $m' = \mathtt{str}(m) \leq m$. Let us first assume that $m' = m$ (i.e., $G_0 = G$ and $\mathtt{str}$ is the identity function). In this case, a standard argument

(based on [NW94] and first noted in [IW98]) shows that there exists a probabilistic polynomial time algorithm A_{NW} that satisfies the following: When given as input a circuit $D_m : \{0,1\}^m \to \{0,1\}$ that $(1/m^{\log\log(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and also given oracle access to $f^{\mathtt{GL}(\mathtt{ws})}$ on ℓ-bit inputs, with probability at least $1/O(m)$ the algorithm A_{NW} outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{x\in\{0,1\}^\ell}[C_\ell(x) = f^{\mathtt{GL}(\mathtt{ws})}(x)] \geq 1/2 + 1/O(m^{\log\log(m)^k})$.

Towards extending this claim to the setting of $\mathtt{str}(m) < m$, let us quickly recap the original construction of A_{NW}: The algorithm randomly chooses an index $i \in [m]$ (for a hybrid argument) and values for all the bits in the seed of the NW generator outside the i^{th} set (in the underlying design); then uses its oracle to query $\mathrm{poly}(m)$ values for $f^{\mathtt{GL}(\mathtt{ws})}$ (these are potential values for the output indices whose sets in the seed intersect with the i^{th} set), and "hard-wires" them into a circuit C_ℓ that gets input $x \in \{0,1\}^\ell$, simulates the corresponding m-bit output of the PRG, and uses the distinguisher to decide if $x \in f^{\mathtt{GL}(\mathtt{ws})}$. Now, note that if the output of the PRG is truncated to length $m' = \mathtt{str}(m) < m$, the construction above works essentially the same if we choose an initial index $i \in [m']$ instead of $i \in [m]$, and if C_ℓ completes x to an m'-bit output of the PRG instead of an m-bit output. Indeed, referring to the underlying analysis, these changes only improve the guarantee on the algorithm's probability of success (we do not use the fact that the guarantee is better). Thus, for any $m' = \mathtt{str}(m) \leq m$, there is an algorithm A_{NW} that gets as input a circuit $D_{m'} : \{0,1\}^{m'} \to \{0,1\}$ that $(1/m^{\log\log(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and oracle access to $f_\ell^{\mathtt{GL}(\mathtt{ws})}$, and with probability at least $1/O(m)$ outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{x\in\{0,1\}^\ell}[C_\ell(x) = f^{\mathtt{GL}(\mathtt{ws})}(x)] \geq 1/2 + 1/O(m^{\log\log(m)^k})$.

Now, for $\ell \in \mathbb{N}$, let $m = m(\ell)$ be such that ℓ is the seed length of G on m-bit inputs, and let $m' = \mathtt{str}(m)$. Our probabilistic algorithm is given as input 1^ℓ and non-uniform advice (a, m') such that $|a| = \log(t(m)) = \log(m) \cdot \mathrm{loglog}(m)^k = \epsilon \cdot \ell'$; note that, since $m' \leq m$, the total length of the advice is at most $\epsilon \cdot \ell' + \log(m) < 2\epsilon \cdot \ell'$. Our probabilistic algorithm simulates the algorithm A on input $1^{m'}$ with the advice a, and feeds the output of A as input for A_{NW}. This algorithm runs in time $m^{O(\log\log(m)^k)} = 2^{\ell'}$. Note that with probability more than $(1/m^{\log\log(m)^k})$, the algorithm A outputs $D_{m'} : \{0,1\}^{m'} \to \{0,1\}$ that $(1/m^{\log\log(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and conditioned on this event, with probability at least $1/O(m)$ the combined algorithm outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ that correctly computes $f^{\mathtt{GL}(\mathtt{ws})}$ on $1/2 + 1/O(m^{\log\log(m)^k}) > 1/2 + 2^{-\ell'}$ of the ℓ-bit inputs. □

We will call the algorithm in the statement of Claim 4.3.11.1 a weak learner for $f^{\mathtt{GL}(\mathtt{ws})}$ on input length $\ell(m)$. Then, Claim 4.3.11.1 implies that there exists a weak learner for $f^{\mathtt{GL}(\mathtt{ws})}$ on any input length in $2L_n = \{2i : i \in L_n\}$. See Figure 4.1 for a pictorial description of the sets L_n, $2L_n$, and S_n, and for a reminder about our assumptions at this point.

Given as input $1^{\ell(n)}$, we construct in time $\mathrm{poly}(1/\delta(\ell(n))) = 2^{O(\ell(n)/\log(\ell(n))^{c+1})} = 2^{O(\ell'(n))}$ a circuit for $f^{\mathtt{ws}}_{\ell(n)}$, by inductively constructing circuits for $f_i^{\mathtt{ws}}$, for increasing

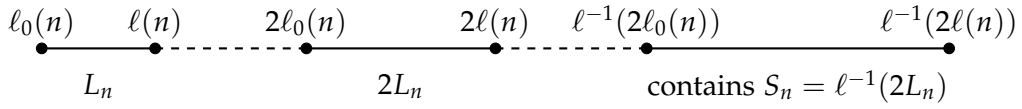

Figure 4.1: We want to compute $f^{\mathtt{ws}}$ on inputs of length $\ell(n)$. We define a corresponding interval $L_n = \{\ell_0(n), ..., \ell(n)\}$ of input lengths, where $\ell_0(n) \geq \ell(n) - s(\ell(n))$, in which we will use the downward self-reducibility of $f^{\mathtt{ws}}$. We assume that there is a uniform distinguisher A for the PRG on all input lengths in $S_n = \ell^{-1}(2L_n)$, and deduced that there exists a weak learner for $f^{\mathtt{GL}(\mathtt{ws})}$ on all input lengths in $2L_n$.

values of $i \in L_n = \{\ell_0(n), ..., \ell(n)\}$, where for each i we will construct the corresponding circuit in time $2^{O(i/\log(i)^{c+!})}$. Indeed, the construction for the base case $i = \ell_0(n)$ is trivial, since $f^{\mathtt{ws}}_{\ell_0(n)}$ is computable in time $\mathrm{poly}(1/\delta(\ell_0(n))) \leq 2^{O(\ell_0(n)/\log(\ell_0(n))^{c+1})}$, where the inequality is due to our hypothesis that δ is sufficiently large (the precise requirement from δ will be specified below). Therefore we just need to prove the inductive step. This will be done as follows:

Claim 4.3.11.2. *There exists an algorithm that gets as input $i \in L_n \setminus \{\ell_0(n)\}$ and a circuit $C_{i-1} : \{0,1\}^{i-1} \to \{0,1\}$ that computes $f^{\mathtt{ws}}_{i-1}$, runs in time $2^{O(i/\log(i)^{c+1})} \cdot \mathrm{poly}(|C_{i-1}|)$, and with probability at least $1 - \exp(i/\log(i)^{c+1})$ outputs a circuit $C_i : \{0,1\}^i \to \{0,1\}$ of size $2^{O(i/\log(i)^{c+1})}$ that computes $f^{\mathtt{ws}}_i$. (Note that the size of the output circuit C_i does not depend on the size of the input circuit C_{i-1}.)*

Proof. Let $i' = 2i/\log(2i)^{c+1}$, and let $S = |C_{i-1}|$. First note that the algorithm can compute $f^{\mathtt{ws}}_i$ in time $\mathrm{poly}(1/\delta(i), S)$ (using the downward self-reducibility of $f^{\mathtt{ws}}$ and the circuit C_{i-1}) and also compute $f^{\mathtt{GL}(\mathtt{ws})}_{2i}$ in time $\mathrm{poly}(1/\delta(i), S)$ (using the fact that $f^{\mathtt{GL}(\mathtt{ws})}(x, r) = \sum_{j \in [i]} f^{\mathtt{ws}}_i(x)_j \cdot r_j$). We will construct C_i in four steps:

1. Simulating the learner for $f^{\mathtt{GL}(\mathtt{ws})}_{2i}$. We use the weak learner for $f^{\mathtt{GL}(\mathtt{ws})}_{2i}$ to construct a list of $2^{O(i')}$ circuits $\{0,1\}^{2i} \to \{0,1\}$ of size $2^{i'}$ such that at least one circuit in the list correctly decides $f^{\mathtt{GL}(\mathtt{ws})}_{2i}$ on $1/2 + 2^{-i'}$ of the $(2i)$-bit inputs.

To do so, we enumerate over all $2^{2\epsilon \cdot i'}$ possible advice strings for the weak learner for $f^{\mathtt{GL}(\mathtt{ws})}_{2i}$. For each fixed advice string $a \in \{0,1\}^{2\epsilon \cdot i'}$, we simulate the weak learner with advice a for $2^{O(i')}$ times (using independent randomness in each simulation), while answering its queries to $f^{\mathtt{GL}(\mathtt{ws})}_{2i}$ using C_{i-1}. Note that when a is the "good" advice, each simulation of the learner is successful with probability at least $2^{-i'}$. Thus, with probability at least $1 - \exp(-i')$ our list contains at least one circuit that correctly computes $f^{\mathtt{GL}(\mathtt{ws})}_{2i}$ on at least $1/2 + 2^{-i'}$ of its inputs.

2. Weeding the list to find a circuit for $f_{2i}^{\mathtt{GL}(\mathtt{ws})}$. We now test each of the $2^{O(i')}$ circuits in order to find a single circuit $C_i' : \{0,1\}^{2i} \to \{0,1\}$ that computes $f_{2i}^{\mathtt{GL}(\mathtt{ws})}$ on $1/2 + 2^{-2i'}$ of the inputs.

To test each circuit we randomly sample $2^{O(i')}$ inputs, compute $f_{2i}^{\mathtt{GL}(\mathtt{ws})}$ at each of these inputs using C_{i-1}, and compare the value of $f_{2i}^{\mathtt{GL}(\mathtt{ws})}$ to the output of the candidate circuit. For each circuit, with probability at least $1 - 2^{-O(i')}$ over the sampled inputs, we correctly estimate its agreement with $f_{2i}^{\mathtt{GL}(\mathtt{ws})}$ up to error $2^{-2i'-1}$. Union-bounding over the $2^{O(i')}$ circuits, with probability at least $1 - 2^{-O(i')}$, the circuit that we find in this step has agreement at least $1/2 + 2^{-2i'}$ with $f^{\mathtt{ws}}$.

3. Conversion to a circuit that computes $f_i^{\mathtt{ws}}$ on average. We now convert the circuit C_i' for $f_{2i}^{\mathtt{GL}(\mathtt{ws})}$ to a circuit $\{0,1\}^i \to \{0,1\}^i$ of size $2^{O(i')}$ that computes $f_i^{\mathtt{ws}}$ correctly on $\delta(i) = 2^{-O(i')}$ of its i-bit inputs.[31]

To do so, we first use the algorithm of Goldreich and Levin [GL89] to convert the deterministic circuit C_i' into a probabilistic circuit C_i'' of size $2^{O(i')}$ such that $\Pr[C_i''(x) = f_i^{\mathtt{ws}}(x)] \geq 2^{-O(i')}$, where the probability is taken both over a random choice of $x \in \{0,1\}^i$ and over the internal randomness of C_i''. Specifically, the circuit $C_i'' : \{0,1\}^i \to \{0,1\}$ gets input $x \in \{0,1\}^i$, and simulates the algorithm from [Gol08, Thm 7.8] with parameter $\delta_0 = 2^{-2i'}$, while resolving the oracle queries of the algorithm using the circuit C_i'; then, the circuit C_i'' outputs a random element from the list that is produced by the algorithm from [Gol08]. Since $\mathbb{E}_x[\Pr_r[C_i'(x,r) = f_{2i}^{\mathtt{GL}(\mathtt{ws})}(x,r)]] \geq 1/2 + \delta_0$, it follows that for at least $\delta_0/2$ of the inputs $x \in \{0,1\}^i$ it holds that $\Pr_r[C_i'(x,r) = f_{2i}^{\mathtt{GL}(\mathtt{ws})}(x,r)] \geq 1/2 + \delta_0/2$. For each such input, with probability at least $1/2$ the algorithm of [GL89] outputs a list of size $\mathrm{poly}(1/\delta_0)$ that contains $f^{\mathtt{ws}}(x)$, and thus the circuit C_i'' outputs $f^{\mathtt{ws}}(x)$ with probability $\mathrm{poly}(\delta_0)$.

To conclude we now choose randomness for C_i' and "hard-wire" it into the circuit. With probability at least $1 - \exp(i')$, we obtain a circuit C_i''' of size $2^{O(i')}$ that computes $f_i^{\mathtt{ws}}$ correctly on $\delta = \mathrm{poly}(\delta_0)$ of the inputs.

4. Worst-case to δ-average-case reduction for $f_i^{\mathtt{ws}}$. Our final step is to convert C_i'' (which computes $f_i^{\mathtt{ws}}$ correctly on $\delta(i)$ of the i-bit inputs) into a circuit C_i of size $2^{O(i')}$ that correctly computes $f_i^{\mathtt{ws}}$ on all inputs.

To do so we will use the fact that $f^{\mathtt{ws}}$ is sample-aided worst-case to δ-average-case reducible, and the fact that we can generate random labeled samples $(r, f_i^{\mathtt{ws}}(r))$ by using the circuit C_{i-1} to compute $f_i^{\mathtt{ws}}(r)$. With probability at least $1 - \delta(i)$, the uniform reduction outputs a probabilistic circuit C_i''' of size $2^{O(i')}$ such that for every $x \in \{0,1\}^i$

[31] Recall that in our hypothesis we required a δ-well-structured function where $\delta(n) = 2^{-n/\mathrm{polylog}(n)}$ for a sufficiently large polylogarithmic function. At this point we can specify our precise requirement, which is that $\delta(n) = 2^{-O(n/\log(n)^{c+1})}$, where the universal constant hidden inside the O-notation depends only on universal constants from [GL89] as explained in the argument that we now present.

it holds that $\Pr_r[C_i'''(x,r) = f^{\mathtt{ws}}(x)] \geq 2/3$. [32] Using naive error-reduction we obtain a circuit of size $2^{O(i')}$ that correctly computes $f^{\mathtt{ws}}$ at any input with probability $1 - 2^{-O(i)}$. Then we uniformly choose randomness of this circuit and "hard-wire" the randomness into it, such that with probability at least $1 - 2^{-i}$ we obtain a deterministic circuit $C_i : \{0,1\}^i \to \{0,1\}$ that computes $f_i^{\mathtt{ws}}$. □

Repeating the algorithm from Claim 4.3.11.2 for $i = \ell_0(n) + 1, ..., \ell(n)$, we obtain an algorithm that runs in time $2^{O(\ell')}$, and outputs a circuit for $f_{\ell(n)}^{\mathtt{ws}}$ with probability at least $1 - \sum_{i=\ell'}^{\ell} \exp(i/\log(i)^{c+1}) \geq 2/3$, assuming that ℓ is sufficiently large. ■

In the last part of the proof of Lemma 4.3.11, after we converted a distinguisher for $f^{\mathtt{GL}(\mathtt{ws})}$ into a weak learner for $f^{\mathtt{GL}(\mathtt{ws})}$ (i.e., after Claim 4.3.11.1), we used the existence of the weak learner for $f^{\mathtt{GL}(\mathtt{ws})}$ on $2L_n$ to obtain a circuit that computes $f^{\mathtt{ws}}$ on L_n. This part of the proof immediately implies the following, weaker corollary. (The corollary is weaker since it does not have any "point-wise" property, i.e. does not convert a learner on specific input lengths to a circuit for $f^{\mathtt{ws}}$ on a corresponding input length.)

Corollary 4.3.12 (learning $f^{\mathtt{GL}(\mathtt{ws})} \Longrightarrow$ computing $f^{\mathtt{ws}}$)**.** *Let $c \in \mathbb{N}$ be an arbitrary constant, let $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ be a δ-well-structured function for $\delta(n) = 2^{-n/\log(n)^{c+1}}$, and let $f^{\mathtt{GL}(\mathtt{ws})}$ be defined as in the proof of Lemma 4.3.11. Assume that for every $\ell \in \mathbb{N}$ there exists a* weak learner *for $f^{\mathtt{GL}(\mathtt{ws})}$; that is, an algorithm that gets input 1^ℓ and oracle access to $f_\ell^{\mathtt{GL}(\mathtt{ws})}$, runs in time $\delta^{-1}(\ell)$, and with probability more than $\delta(\ell)$ outputs a circuit over ℓ bits that computes $f^{\mathtt{GL}(\mathtt{ws})}$ correctly on more than $1/2 + \delta(\ell)$ of the inputs. Then, there exists an algorithm that for every ℓ, when given input 1^ℓ, runs in time $2^{\ell/\log(\ell)^c}$ and outputs an ℓ-bit circuit that computes $f^{\mathtt{ws}}$.*

We now use the "point-wise" property of Lemma 4.3.11 to deduce two "almost-always" versions of Proposition 4.3.10. Recall that in our construction of a well-structured function $f^{\mathtt{ws}}$, on some input lengths $f^{\mathtt{ws}}$ is defined trivially, and thus it cannot be that $f^{\mathtt{ws}}$ is "hard" almost-almost.[33] However, since $\mathtt{TQBF}$ can be reduced to $f^{\mathtt{ws}}$ with a quasilinear blow-up $b : \mathbb{N} \to \mathbb{N}$, we can still deduce the following: If $\mathtt{TQBF}$ is "hard" almost-always, then for every $n \in \mathbb{N}$ there exists $n' \leq b(n)$ such that $f^{\mathtt{ws}}$ is "hard" on input length n' (i.e., this holds for the smallest $n' \geq n$ of the form $b(n_0)$ for $n_0 \in \mathbb{N}$).

In our first "almost-always" result, the hypothesis is that a well-structured function is "hard" on a dense set of input lengths as above, and the conclusion is that there exists an "almost-everywhere" HSG for uniform circuits.

Proposition 4.3.13 ("almost everywhere" hardness of $f^{\mathtt{ws}} \Rightarrow$ "almost everywhere" derandomization of $\mathcal{RP}$ "on average")**.** *Assume that for some constant $c \in \mathbb{N}$ and for $\delta(n) =$*

[32] In Definition 4.3.5 the output circuit has oracle gates to a function that agrees with the target function on a δ fraction of the inputs. Indeed, we replace these oracle gates with copies of the circuit C_i''.

[33] Moreover, in every small interval of input lengths, there is an input length on which $f^{\mathtt{ws}}$ can be solved in time $\mathrm{poly}(1/\delta)$ (without using an oracle).

$2^{-n/\log(n)^{c+1}}$ *there exists a* $(\delta, \text{polylog}(n))$*-well-structured function and* $b(n) = \tilde{O}(n)$ *such that for every probabilistic algorithm that runs in time* $2^{n/\log(n)^c}$*, and every sufficiently large* $n \in \mathbb{N}$*, the algorithm fails to compute* $f^{\mathtt{ws}}$ *on input length* $\overline{n} = \min\{b(n_0) \geq n : n_0 \in \mathbb{N}\}$*. Then, for every* $k \in \mathbb{N}$ *and for* $t(n) = n^{\log\log(n)^k}$ *there exists a* $(1/t)$*-HSG for* $(t, \log(t))$*-uniform circuits that is computable in time* $n^{\text{polyloglog}(n)}$ *and has seed length* $\tilde{O}(\log(n))$*.*

Proof. We instantiate Lemma 4.3.11 with the constant c, the function $f^{\mathtt{ws}}$, the parameter $2k$ instead of k (i.e., the parameter t in Lemma 4.3.11 is $t(n) = n^{\log\log(n)^{2k}}$) and with $\mathtt{str}(n) = n$ (i.e., $\mathtt{str}$ is the identity function). Let $\ell(n) = \lceil \tilde{O}(\log(n)) \rceil$ be the quasilogarithmic function given by Lemma 4.3.11, let $G = G_0$ be the corresponding PRG, and let $\ell_G(n) = \tilde{O}(\log(n))$ be the seed length of G. From our hypothesis regarding the hardness of $f^{\mathtt{ws}}$, we can deduce the following:

Corollary 4.3.13.1. *For every* $n \in \mathbb{N}$ *there is a polynomial-time-enumerable set* $\overline{S_n} = S_{n^{\text{polyloglog}(n)}} \subset [n, n^{\text{polyloglog}(n)}]$ *of size* $\text{polyloglog}(n)$ *such that for every probabilistic algorithm* A' *that runs in time* t^2 *and uses* $2\log(t)$ *bits of advice, if* $n \in \mathbb{N}$ *is sufficiently large then there exists* $m \in \overline{S_n}$ *such that* $G(1^m, \mathbf{u}_{\ell_G(m)})$ *is* $(1/t^2(m))$*-pseudorandom for* A'*.*

Proof. For every $n \in \mathbb{N}$, let $\overline{\ell}(n) = \min\{b(\ell_0) \geq \ell(n) : \ell_0 \in \mathbb{N}\}$, and let $\overline{n} = \ell^{-1}(\overline{\ell}(n)) \in [n, n^{\text{polyloglog}(n)}]$. We define $\overline{S_n} = S_{\overline{n}}$, where $S_{\overline{n}}$ is the set from Item (2) of Lemma 4.3.11 that corresponds to $\overline{n}$. Note that $\overline{S_n} \subset [n, n^{\text{polyloglog}(n)}]$ and that $|\overline{S_n}| \leq \text{polyloglog}(n)$.

Now, let A' be a probabilistic algorithm as in our hypothesis, let F' be the corresponding probabilistic algorithm from Lemma 4.3.11 that runs in time $t_{F'}(i) = 2^{i/\log(i)^c}$, and let $n \in \mathbb{N}$ be sufficiently large. By Lemma 4.3.11, if there is no $m \in \overline{S_n}$ such that $G(1^m, \mathbf{u}_{\ell_G(m)})$ is $(1/t(m))$-pseudorandom for A', then F' correctly computes $f^{\mathtt{ws}}$ on input length $\overline{\ell}(\overline{n}) = \overline{\ell}(n)$, which contradicts our hypothesis. □

The HSG, denoted H, gets input 1^n, uniformly chooses $m \in \overline{S_n}$, computes $G(1^m, s)$ for a random $s \in \{0,1\}^{\ell_G(m)}$, and outputs the n-bit prefix of $G(1^m, s)$. Note that the seed length that H requires is $\tilde{O}(\log(n^{\text{polyloglog}(n)})) + \log(|\overline{S_n}|) = \tilde{O}(\log(n))$, and that H is computable in time at most $n^{\text{polyloglog}(n)}$.

To prove that H is a $(1/t)$-HSG for $(t, \log(t))$-uniform circuits, let A be a probabilistic algorithm that runs in time t and uses $\log(t)$ bits of advice. Assume towards a contradiction that there exists an infinite set $B_A \subseteq \mathbb{N}$ such that for every $n \in B_A$, with probability more than $1/t(n)$ the algorithm A outputs a circuit $D_n : \{0,1\}^n \to \{0,1\}$ satisfying $\Pr_s[D_n(H(1^n, s)) = 0] = 1$ and $\Pr_{x \in \{0,1\}^n}[D_n(x) = 1] > 1/t(n)$. We will construct an algorithm A' that runs in time less than t^2, uses $\log(t) + \log(n) < 2\log(t)$ bits of advice, and for infinitely-many sets of the form $\overline{S_n}$, for every $m \in \overline{S_n}$ it holds that $G(1^m, \mathbf{u}_{\ell_G(m)})$ is not $(1/t(m))$-pseudorandom for A'. This contradicts Corollary 4.3.13.1.

The algorithm A' gets input 1^m, and as advice it gets an integer of size at most m. Specifically, if m is in a set $\overline{S_n}$ for some $n \in B_A$, then the advice will be set to n; and otherwise the advice is zero (which signals to A' that it can fail on input length

m). For any $m \in \mathbb{N}$ such that the first case holds, we know that $A(1^n)$ outputs, with probability more than $1/t(n)$, a circuit $D_n : \{0,1\}^n \to \{0,1\}$ satisfying both $\Pr_{s \in \{0,1\}^{\tilde{O}(\log(n))}}[D_n(H(1^n, s)) = 0] = 1$ and $\Pr_{x \in \{0,1\}^n}[D_n(x) = 1] > 1/t(n)$. The algorithm A' simulates A on input length n, and outputs a circuit $D_m : \{0,1\}^m \to \{0,1\}$ such that D_m computes D_n on the n-bit prefix of its input. By our hypothesis regarding D_n, when fixing the first part of the seed of H to be the integer m, we have that $\Pr_{s'}[D_n(H(1^n, m \circ s')) = 0] = \Pr_{s'}[D_m(G(1^m, s')) = 0] = 1$, whereas $\Pr_{x \in \{0,1\}^m}[D_m(x) = 1] > 1/t(n)$. It follows that D_m distinguishes the m-bit output of G from uniform with advantage $1/t(n) \geq 1/t(m)$. ■

We also prove another "almost-everywhere" version of Proposition 4.3.10. Loosely speaking, under the same hypothesis as in Proposition 4.3.13, we show that $\mathcal{BPP}$ can be derandomized "on average" using only a small (triple-logarithmic) amount of advice. In contrast to the conclusion of Proposition 4.3.13, in the following proposition we do *not* construct a PRG or HSG, but rather simulate every $\mathcal{BPP}$ algorithm by a corresponding deterministic algorithm that uses a small amount of non-uniform advice.

Proposition 4.3.14 ("almost everywhere" hardness of $f^{\mathtt{ws}} \Rightarrow$ "almost everywhere" derandomization of $\mathcal{BPP}$ "on average" with short advice)**.** *Assume that for some constant $c \in \mathbb{N}$ and for $\delta(n) = 2^{-n/\log(n)^{c+1}}$ there exists a $(\delta, \mathrm{polylog}(n))$-well-structured function and $b(n) = \tilde{O}(n)$ such that for every probabilistic algorithm that runs in time $2^{n/\log(n)^c}$, and every sufficiently large $n \in \mathbb{N}$, the algorithm fails to compute $f^{\mathtt{ws}}$ on input length $\overline{n} = \min\{b(n_0) \geq n : n_0 \in \mathbb{N}\}$.*

For $k \in \mathbb{N}$ and $t(n) = n^{\mathrm{loglog}(n)^k}$, let $L \in \mathcal{BPTIME}[t]$ and let F be a probabilistic t-time algorithm. Then, there exists a deterministic machine D that runs in time $n^{\mathrm{polyloglog}(n)}$ and gets $O(\mathrm{logloglog}(n))$ bits of non-uniform advice such that for all sufficiently large $n \in \mathbb{N}$, the probability (over coin tosses of F) that $F(1^n)$ is an input $x \in \{0,1\}^n$ for which $D(x) \neq L(x)$ is at most $1/t(n)$.

Proof. Let us first prove the claim assuming that $L \in \mathcal{BPTIME}[t]$ can be decided using only a number of random coins that equals the input length; later on we show how to remove this assumption (by a padding argument). For t as in our hypothesis for L as above, let M be a probabilistic t-time algorithm that decides L and that for every input $x \in \{0,1\}^*$ uses $|x|$ random coins, and let F be a probabilistic t-time algorithm. Consider the algorithm A that, on input 1^n, simulates F on input 1^n to obtain $x \in \{0,1\}^n$, and outputs a circuit $C_x : \{0,1\}^n \to \{0,1\}$ that computes the decision of M at input x as a function of the random coins of M.

We instantiate Lemma 4.3.11 with the constant c, the function $f^{\mathtt{ws}}$, and the parameter k. Let $\ell = \tilde{O}(\log(n))$ be the quasilogarithmic function given by the lemma, let G_0 be the PRG, and let $\ell_G = \tilde{O}(\log(n))$ be the seed length of G_0. We first need a claim similar to Corollary 4.3.13.1, but this time also quantifying over the function $\mathtt{str}$:

Corollary 4.3.14.1. *For every $n \in \mathbb{N}$ there is a polynomial-time-enumerable set $\overline{S_n} = S_{n^{\mathrm{polyloglog}(n)}} \subset [n, n^{\mathrm{polyloglog}(n)}]$ of size $\mathrm{polyloglog}(n)$ that satisfies the following. For every $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ satisfying $\mathtt{str}(n) \leq n$, let $G_{\mathtt{str}}$ be the algorithm that on input 1^n uses a random seed of length $\tilde{O}(\log(n))$, computes G_0, which outputs an n-bit string, and truncates the output to length $\mathtt{str}(n)$. Then, for every probabilistic algorithm A' that runs in time t and uses $\log(t)$ bits of advice, if $n \in \mathbb{N}$ is sufficiently large then there exists $m \in \overline{S_n}$ such that $G_{\mathtt{str}}(1^m, \mathbf{u}_{\ell_G(m)})$ is $(1/t(m))$-pseudorandom for A'.*

Proof. For any $n \in \mathbb{N}$ we define $\overline{\ell}(n)$ and $\overline{S_n}$ as in the proof of Corollary 4.3.13.1. For any $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ satisfying $\mathtt{str}(n) \leq n$, let $G_{\mathtt{str}}$ be the corresponding function. Now, let A' be any probabilistic algorithm as in our hypothesis, let F' be the corresponding probabilistic algorithm from Lemma 4.3.11 that runs in time $t_{F'}(i) = 2^{i/\log(i)^c}$, and let $n \in \mathbb{N}$ be sufficiently large. By Lemma 4.3.11, if there is no $m \in \overline{S_n}$ such that $G_{\mathtt{str}}(1^m, \mathbf{u}_{\ell_G(m)})$ is $(1/t(m))$-pseudorandom for A', then F' correctly computes $f^{\mathtt{ws}}$ on input length $\overline{\ell}(n)$. This contradicts our hypothesis regarding $f^{\mathtt{ws}}$. □

The machine D gets input $x \in \{0,1\}^n$ and advice of length $O(\mathrm{logloglog}(n))$, which is interpreted as an index of an element m in the set $\overline{S_n}$. Then, for each $s \in \{0,1\}^{\ell_G(m)}$ the algorithm computes the n-bit prefix of $G_0(1^m, s)$, denoted $w_s = G_0(1^m, s)_{1,\ldots,n}$, and outputs the majority value of $\{M(x, w_s) : s \in \{0,1\}^{\ell_G(m)}\}$. Note that the machine D indeed runs in time $m^{\mathrm{polyloglog}(m)} = n^{\mathrm{polyloglog}(n)}$.

Our goal now is to prove that for every sufficiently large $n \in \mathbb{N}$ there exists advice $m \in \overline{S_n}$ such that with probability at least $1 - 1/t(n)$ over the coin tosses of F (which determine $x \in \{0,1\}^n$ and $C_x \colon \{0,1\}^n \to \{0,1\}$) it holds that

$$\left| \Pr_{r \in \{0,1\}^n}[C_x(r) = 1] - \Pr_s[C_x(G_0(1^m, s)_{1,\ldots,n}) = 1] \right| < 1/t(n) \,, \tag{4.3.2}$$

which is equivalent (for a fixed $x \in \{0,1\}^n$) to the following statement:

$$\left| \Pr_{r \in \{0,1\}^n}[M(x, r) = 1] - \Pr_s[M(x, w_s) = 1] \right| < 1/t(n) \,. \tag{4.3.3}$$

Indeed, proving this would suffice to prove our claim, since for every $x \in \{0,1\}^n$ such that Eq. (4.3.3) holds we have that $D(x) = L(x)$.

To prove the claim above, assume towards a contradiction that there exists an infinite set of input lengths $B_A \subseteq \mathbb{N}$ such that for every $n \in B_A$ and every advice $m \in \overline{S_n}$, with probability more than $1/t(n)$ over $x \leftarrow F(1^n)$ it holds that $C_x : \{0,1\}^n \to \{0,1\}$ violates Eq. (4.3.2). Let $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ be defined by $\mathtt{str}(m) = n$ if $m \in \overline{S_n}$ for some $n \in B_A$, and $\mathtt{str}(m) = m$ otherwise.[34] Then, our assumption implies that for infinitely-many input lengths $n \in B_A$, for every $m \in \overline{S_n}$ it holds that $G_{\mathtt{str}}(1^m, \mathbf{u}_{\ell_G(m)})$ is not $(1/t(n))$-pseudorandom for A. This contradicts Corollary 4.3.14.1.

[34] Note that $\mathtt{str}$ is well-defined, since we can assume without loss of generality that $\overline{S_n} \cap \overline{S_{n'}} = \emptyset$ for distinct $n, n' \in B_A$ (i.e., we can assume without loss of generality that n and n' are sufficiently far apart).

Finally, let us remove the assumption that L can be decided using a linear number of coins, by a padding argument. For any $L \in \mathcal{BPTIME}[t]$, consider a padded version $L^{\mathtt{pad}} = \{(x, 1^{t(|x|)}) : x \in L\}$, and note that $L^{\mathtt{pad}}$ can be decided in linear time using $|z|$ coins on any input z. By the argument above, for every probabilistic t-time algorithm $F^{\mathtt{pad}}$ there exists an algorithm $D^{\mathtt{pad}}$ that runs in time $t_{D^{\mathtt{pad}}}(m) = m^{\mathrm{polyloglog}(m)}$ such that for all sufficiently large $m \in \mathbb{N}$ it holds that $\Pr_{z \leftarrow F^{\mathtt{pad}}(1^m)}[D^{\mathtt{pad}}(z) \neq L^{\mathtt{pad}}(z)] \leq 1/t(m)$.

We define the algorithm D in the natural way, i.e. $D(x) = D^{\mathtt{pad}}(x, 1^{t(|x|)})$, and note that this algorithm runs in time $n^{\mathrm{polyloglog}(n)}$. Assume towards a contradiction that there exists a t-time algorithm F and an infinite set of input lengths $B_F \subseteq \mathbb{N}$ such that for every $n \in B_F$, with probability more than $1/t(n)$ it holds that $D(x) \neq L(x)$. Consider the algorithm $F^{\mathtt{pad}}$ that on input of the form $1^{n+t(n)}$ runs $F(1^n)$ to obtain $x \in \{0,1\}^n$, and outputs $(x, 1^n)$ (on inputs of another form $F^{\mathtt{pad}}$ fails and halts), and let $B_{F^{\mathtt{pad}}} = \{n + t(n) : n \in B_F\}$. For any $m \in B_{F^{\mathtt{pad}}}$ we have that

$$\Pr_{z \leftarrow F^{\mathtt{pad}}(1^m)}[D^{\mathtt{pad}}(z) \neq L^{\mathtt{pad}}(z)] = \Pr_{x \leftarrow F(1^n)}[D(x) \neq L(x)] > 1/t(n) > 1/t(m) \,,$$

which yields a contradiction. ■

An aside: Derandomization using quasilogarithmic space. The PRG constructed in Lemma 4.3.11 actually works in *quasilogarithmic space* (since $f^{\mathtt{ws}}$ is computable in linear space), except for one crucial part: The construction of combinatorial designs. Combinatorial designs with parameters as in our proof actually *can* be constructed in logarithmic space, but only for values of ℓ that are of a specific form (since the constructions are algebraic).[35] However, in our downward self-reducibility argument we need such designs for *every* integer ℓ (such that we can assume the existence of distinguishers on the set $S_n = \ell^{-1}(2L_n)$, and hence of learners for $f^{\mathtt{GL}(\mathtt{ws})}$ on $2L_n$).

4.3.5 Proofs of Theorems 4.3.1 and 4.3.2

Let us now formally state Theorem 4.3.1 and prove it. The theorem follows immediately as a corollary of Lemma 4.3.9 and Proposition 4.3.10.

Theorem 4.3.15 (rETH $\Rightarrow$ i.o.-PRG for uniform circuits)**.** *Assume that there exists $i \geq 1$ such that $\mathtt{TQBF} \notin \mathcal{BPTIME}[2^{n/\log(n)^i}]$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\mathrm{loglog}(n)^k}$ there exists a $(1/t)$-i.o.-PRG for $(t, \log(t))$-uniform circuits that has seed length $\tilde{O}(\log(n))$ and is computable in time $n^{\mathrm{polyloglog}(n)}$.*

Proof. Let $\delta(n) = 2^{n/\log(n)^{3c}}$ for a sufficiently large constant $c \in \mathbb{N}$. By Lemma 4.3.9, there exists $(\delta, O(\ell^2))$-well-structured function $f^{\mathtt{ws}}$ that is computable in linear space, and such that $\mathtt{TQBF}$ reduces to $f^{\mathtt{ws}}$ in time $\mathtt{ql}(n) = n \cdot \log(n)^{2c+r}$, where $r \in \mathbb{N}$ is a

[35] This can be done using an idea from [HR03, Lemma 5.5] (attributed to Salil Vadhan), essentially "composing" Reed-Solomon codes over $GF(n)$ of degree $n/\mathrm{polylog}(n)$ with standard designs (a-la Nisan and Wigderson [NW94]; see [HR03, Lemma 2.2]) with set-size $\ell = \mathrm{polylog}(n)$.

universal constant. Using our hypothesis, we deduce that $f^{\mathtt{ws}}$ cannot be computed in probabilistic time $2^{n/\log(n)^{3c-1}}$; this is the case since otherwise, TQBF could have been computed in probabilistic time

$$2^{\mathtt{ql}(n)/\log(\mathtt{ql}(n))^{3c-1}} = 2^{n\cdot\log(n)^{2c+r}/\log(\mathtt{ql}(n))^{3c-1}} < 2^{n/\log(n)^{c-r-1}}, \tag{4.3.4}$$

which is a contradiction if $c \geq i + r + 1$. Our conclusion now follows from Proposition 4.3.10. ∎

We also formally state Theorem 4.3.2 and prove it, as a corollary of Lemma 4.3.9 and of Propositions 4.3.13 and 4.3.14.

Theorem 4.3.16 (a.a.-rETH ⇒ almost-always HSG for uniform circuits and alm0st-always "average-case" derandomization of $\mathcal{BPP}$)**.** *Assume that there exists $i \geq 1$ such that* TQBF $\notin$ *i.o.-*$\mathcal{BPTIME}[2^{n/\log(n)^i}]$*. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\mathrm{loglog}(n)^k}$:*

1. *There exists a $(1/t)$-HSG for $(t, \log(t))$-uniform circuits that is computable in time $n^{\mathrm{polyloglog}(n)}$ and has seed length $\tilde{O}(\log(n))$.*
2. *For every $L \in \mathcal{BPTIME}[t]$ and probabilistic t-time algorithm F there exists a deterministic machine D that runs in time $n^{\mathrm{polyloglog}(n)}$ and gets $O(\mathrm{logloglog}(n))$ bits of non-uniform advice such that for all sufficiently large $n \in \mathbb{N}$ the probability (over coin tosses of F) that $F(1^n)$ is an input $x \in \{0,1\}^n$ for which $D(x) \neq L(x)$ is at most $1/t(n)$.*

Proof. Note that both Proposition 4.3.13 and Proposition 4.3.14 rely on the same hypothesis, and that their respective conclusions correspond to Items (1) and (2) in our claim. Thus, it suffices to prove that their hypothesis holds.

To see this, as in the proof of Theorem 4.3.15, let $\delta(n) = 2^{n/\log(n)^{3c}}$ for a sufficiently large constant $c \in \mathbb{N}$, and let $f^{\mathtt{ws}}$ be the $(\delta, \mathrm{polylog}(n))$-well-structured function that is obtained from Lemma 4.3.9 with parameter δ. Let $r \in \mathbb{N}$ be the universal constant from Lemma 4.3.9, and let $\mathtt{ql}(n) = n \cdot \log(n)^{2c+r}$. Note that for every algorithm that runs in time $2^{n/\log(n)^{3c-1}}$ and every sufficiently large $n_0 \in \mathbb{N}$, the algorithm fails to compute $f^{\mathtt{ws}}$ on input length $n = \mathtt{ql}(n_0)$; this is because otherwise we could have computed TQBF on infinitely-often n_0's in time $2^{n/\log(n)^{c-r-1}} \leq 2^{n_0/\log(n_0)^k}$, where the calculation is as in Eq. (4.3.4). This implies the hypothesis of Propositions 4.3.13 and 4.3.14. ∎

4.3.6 Appendices for Section 4.3

4.3.6.1 On implications of MAETH

Consider the hypothesis MAETH, which asserts that *co*-3SAT cannot be solved by Merlin-Arthur protocols running in time $2^{\epsilon \cdot n}$, for some $\epsilon > 0$. Recall that the "strong" version of this hypothesis is false (since Williams [Wil16] showed that #CircuitSAT

can be solved by a Merlin-Arthur protocol in time $\tilde{O}(2^{n/2})$), but there is currently no evidence against the "non-strong" version.

As mentioned in Section 4.3.1, the assumption MAETH can be easily shown to imply strong circuit lower bounds and derandomization of $prBPP$ (and thus also of $prMA$). Specifically, the following more general (i.e., parametrized) result relies on a standard Karp-Lipton-style argument, which originates in [BFN+93]. We note in advance that after the proof of this result we prove another result, which shows a very different tradeoff between MA lower bounds (specifically, lower bounds for fixed-polynomial-time verifiers) and derandomization.

Theorem 4.3.17 (lower bounds for MA algorithms imply non-uniform circuit lower bounds)**.** *There exists $L \in \mathcal{E}$ and a constant $k > 1$ such that for any time-constructible function $S : \mathbb{N} \to \mathbb{N}$ such that $S(n) \geq n$ the following holds. Assume that $DTIME[2^n] \not\subseteq MATIME[S']$, where $S'(n) = S(k \cdot n)^k$. Then, $L \notin SIZE[S]$.*

Note that, using Corollary 2.4.8, under the hypothesis of Theorem 4.3.17 we have that $\mathtt{CAPP} \in \mathtt{i.o.}prDTIME[T]$, where $T(n) = 2^{O(S^{-1}(n^{O(1)}))}$. In particular, under MAETH (which refers to $S(n) = 2^{\Omega(n/\log(n))}$) we have that $prBPP \subseteq \mathtt{i.o.}prDTIME[n^{O(\log\log(n))}]$.

Proof of Theorem 4.3.17. Let L be the problem from Proposition 4.4.8. Assuming towards a contradiction that $L \in SIZE[S]$, we show that $DTIME[2^n] \subseteq MATIME[S']$.

Let $L_0 \in DTIME[2^n]$. We construct a probabilistic verifier that gets input $x_0 \in \{0,1\}^{n_0}$, and if $x_0 \in L_0$ then for some non-deterministic choices the verifier accepts with probability one, and if $x_0 \notin L_0$ then for all non-deterministic choices the verifier rejects, with high probability. The verifier first reduces L_0 to L, by computing $x \in \{0,1\}^n$ of length $n = O(n_0)$ such that $x_0 \in L_0$ if and only if $x \in L$.

Let $n' = \ell(n) = O(n) = O(n_0)$. By our hypothesis, there exists a circuit over n' input bits of size $S(n')$ that decides $L_{n'}$. The verifier guesses a circuit $C_L : \{0,1\}^{n'} \to \{0,1\}$ of size $S(n')$, and simulates the machine M from Proposition 4.4.8 on input x, while resolving its oracle queries of using C_L. The verifier accepts if and only if M accepts. Note that if $x_0 \in L_0$ and the verifier's guess was correct (i.e., C_L decides $L_{n'}$), then the verifier accepts with probability one. On the other hand, if $x_0 \notin L_0$, then for every guess of C_L (i.e., every oracle for M) the verifier rejects, with high probability. The running time of the verifier is $\mathrm{poly}(n) \cdot \mathrm{poly}(S(n')) = S(O(n))^{O(1)}$. ∎

In the following result, instead of assuming strong (e.g., super-polynomial) lower bounds for $MATIME$ against $\mathcal{E}$, we assume fixed polynomial lower bounds for $MATIME$ against $\mathcal{P}$, and deduce both a sub-exponential derandomization of BPP, and a polynomial-time derandomization of BPP with n^ϵ advice, for an arbitrarly small constant $\epsilon > 0$.[36]

[36]Recall that, by Adleman's theorem [Adl78; BG81], we can derandomize $prBPP$ with $\mathrm{poly}(n)$ bits of non-uniform advice (and even with $O(n)$ bits, using Theorem 2.5.7). However, an unconditional derandomization of $prBPP$ with $o(n)$ bits of non-uniform advice is not known.

Theorem 4.3.18 (fixed-polynomial-size lower bounds for $\mathcal{MA} \implies$ derandomization and circuit lower bounds)**.** *Assume that for every $k \in \mathbb{N}$ it holds that $\mathcal{P} \not\subseteq \mathtt{i.o.}\mathcal{MATIME}[n^k]$. Then, for every $\epsilon > 0$ it holds that $pr\mathcal{BPP} \subseteq (pr\mathcal{P}/n^\epsilon \cap pr\mathcal{DTIME}[2^{n^\epsilon}])$.*

Proof. In high-level, we want to use our hypothesis to deduce that there exists a polynomial-time algorithm that outputs the truth-table of a "hard" function, and then use that "hard" function for derandomization. Loosely speaking, the following claim, whose proof is a refinement of on an argument from [CMM+19], asserts that if the output string of every polynomial-time algorithm has circuit complexity at most n^k, then all of $\mathcal{P}$ can be decided by $\mathcal{MA}$ verifiers running in time $n^{O(k)}$.

Claim 4.3.18.1. *Assume that there exists $k \in \mathbb{N}$ such that for every deterministic polynomial-time machine M there exists an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ the following holds: For every $x \in \{0,1\}^n$, when the output string $M(x)$ is viewed as a truth-table of a function, this function has circuit complexity at most n^k. Then, $\mathcal{P} \subseteq \mathtt{i.o.}\mathcal{MATIME}[n^{O(k)}]$.*

Proof. Let $L \in \mathcal{P}$, and let M be a polynomial-time machine that decides L. Our goal is to decide L in $\mathcal{MATIME}[n^k]$ on infinitely-many input lengths.

For every $x \in \{0,1\}^n$, let $T_x : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}$ be a polynomial-sized circuit that gets as input a string Π, and accepts if and only if Π is the computational history of $M(x)$ and $M(x) = 1$. Note that the mapping of $x \mapsto T_x$ can be computed in polynomial time (since M runs in polynomial time). Also, fix a PCP system for CircuitSAT with the following properties: The verifier runs in polynomial time and uses $O(\log(n))$ randomness and $O(1)$ queries; the verifier has perfect completeness and soundness error $1/3$; and there is a polynomial-time algorithm W that maps any circuit C and a satisfying assignment for C (i.e., $y \in C^{-1}(1)$) to a PCP proof that the verifier accepts. For every $x \in \{0,1\}^n$ and every input $\Pi \in \{0,1\}^{\mathrm{poly}(n)}$ for T_x, let $W(T_x, \Pi)$ be the corresponding PCP proof that W produces.

Observe that there is a polynomial-time algorithm A that gets as input $x \in \{0,1\}^n$, produces the computational history of $M(x)$, which we denote by $H_{M(x)}$, produces the circuit T_x, and finally prints the PCP witness $W(T_x, H_{M(x)})$. Thus, by our hypothesis, there exists an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ and every $x \in \{0,1\}^n$ there exists a circuit $C_x : \{0,1\}^{O(\log(n))} \to \{0,1\}$ of size n^k whose truth-table is $W(T_x, H_{M(x)})$.

The $\mathcal{MA}$ verifier V gets input x, and expects to get as proof a circuit $C : \{0,1\}^{O(\log(n))} \to \{0,1\}$ bits. The verifier V now simulates the PCP verifier, while resolving its queries to the PCP using the circuit C. Note that for every $n \in S$ and every $x \in \{0,1\}^n$ the following holds: If $M(x) = 1$ then there exists a proof (i.e., a circuit C_x) such that the verifier accepts with probability one; on the other hand, if $M(x) = 0$, then T_x rejects all of its inputs, which implies that for every proof, with probability at least $2/3$ the $\mathcal{MA}$ verifier rejects. □

Using our hypothesis that for every $k \in \mathbb{N}$ it holds that $\mathcal{P} \not\subseteq \mathtt{i.o.}\mathcal{MATIME}[n^k]$, and taking the counter-positive of Claim 4.3.18.1, we deduce that:

Corollary 4.3.18.2. *For every $k \in \mathbb{N}$ there exists a polynomial-time machine M such that for every sufficiently large $n \in \mathbb{N}$ there exists an input $x \in \{0,1\}^n$ such that $M(x)$ is the truth-table of a function with circuit complexity more than n^k.*

Now, fix $\epsilon > 0$, let $L \in prBPP$, and let R be a probabilistic polynomial-time machine that decides L. Given input $x \in \{0,1\}^n$, we decide whether $x \in L$ in polynomial-time and with n^ϵ advice, as follows. Consider the circuit R_x that computes the decision of R at x as a function of the random coins of R, and let $c > 1$ such that the size of R_x is at most n^c. We instantiate Corollary 4.3.18.2 with $k = c'/\epsilon$, where $c' > c$ is a sufficiently large constant. We expect as advice an input y of length n^ϵ to the machine M such that $M(y)$ has circuit complexity $n^{c'}$. We then use $M(y)$ to instantiate Theorem 2.4.7 with seed length $O(\log(n))$ and error $1/10$ and for circuits of size n^c (such that the PRG "fools" the circuit R_x), and enumerate its seeds to approximate the acceptance probability of R_x (and hence decide whether or not $x \in L$).

We now also show that $L \in prDTIME[2^{n^{2\epsilon}}]$. To do so, consider the foregoing algorithm, and assume that it gets no advice. Instead, it enumerates over all 2^{n^ϵ} possible advice strings to obtain 2^{n^ϵ} truth-tables, each of size $\text{poly}(n)$. We know that at least one of these truth-tables has circuit complexity $n^{c'}$. Now the algorithm constructs the truth-table of a function f over $n^\epsilon + O(\log(n))$ bits, which uses the first n^ϵ bits to "choose" one of the 2^{n^ϵ} truth-tables, and uses the $O(\log(n))$ bits as an index to an entry in that truth-table (i.e., for $i \in \{0,1\}^{n^\epsilon}$ and $z \in O(\log(n))$ it holds that $f(i,z) = g_i(z)$, where g_i is the function that is obtained from the i^{th} advice string). Note that, since at least one of the 2^{n^ϵ} functions had circuit complexity $n^{c'}$, it follows that f also has circuit complexity $n^{c'}$. Thus, this algorithm can use f to instantiate Theorem 2.4.7 with seed length $n^\epsilon + O(\log(n))$ and for circuits of size n^c to "fool" the circuit R_x. ■

4.3.6.2 Polynomials are sample-aided worst-case to average-case reducible

Recall that in Section 4.3.3 we defined the notion of sample-aided worst-case to δ-average-case-reducible function (see Definitions 4.3.4 and 4.3.5), following [GR17]. In this appendix we explain why labeled samples can be helpful for uniform worst-case to "rare-case" reductions, and show that low-degree polynomials are indeed sample-aided worst-case to average-case-reducible.

Consider a function f whose truth-table is a codeword of a locally list-decodable code, and also assume that f is randomly self-reducible (i.e., computing f in the worst-case is reducible to computing f on, say, .99 of the inputs). Then, for every circuit $\tilde{C}$ that agrees with f on a tiny fraction of inputs (i.e., $\tilde{C}$ computes a "corrupt" version of f), we can efficiently produce a small list of circuits with oracle gates to $\tilde{C}$ such that one of these circuits correctly computes f on all inputs. The main trouble is that we don't know which candidate circuit in this list to use. This is where the labeled samples come in: We can iterate over the candidates in the list, use the labeled samples to *test* each candidate circuit for agreement with f, and with high probability find a circuit that agrees with f on (say) .99 of the inputs. Then, using the random self-reducibility of f, we obtain a circuit that correctly computes f on each input, with high probability.

The crucial property that we need from the code in order to make the foregoing algorithmic approach work is that the local list-decoding algorithm will *efficiently* produce a relatively *short* list. Specifically, recall that by our definition, a sample-aided worst-case to δ-average-case reduction needs to run in time $\text{poly}(1/\delta)$. Hence, we need a list-decoding algorithm that runs in time $\text{poly}(1/\delta)$ (and indeed produces a list of such size). A suitable local list-decoding algorithm indeed exists in the case that the code is the Reed-Muller code, which leads us to the following result:

Proposition 4.3.19 (low-degree polynomials are uniformly worst-case to average-case reducible with a self-oracle)**.** *Let $q : \mathbb{N} \to \mathbb{N}$ be a field-size function, let $\ell : \mathbb{N} \to \mathbb{N}$ such that $n \geq \ell \cdot \log(q)$, and let $d, \rho : \mathbb{N} \to \mathbb{N}$ such that $10\sqrt{d(n)/q(n)} \leq \rho(n) \leq (q(n))^{-\Omega(1)} = o(1)$. Let $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ be a sequence of functions such that f_n computes a polynomial $\mathbb{F}_n^{\ell(n)} \to \mathbb{F}_n$ of degree $d(n)$ where $|\mathbb{F}_n| = q(n)$. Then f is sample-aided worst-case to ρ-average-case reducible.*

Proof. We construct a probabilistic machine M that gets input 1^n, and oracle access to a function $\widetilde{f_n}$ that agrees with f_n on $\rho(n)$ of the inputs, and also $\text{poly}(1/\rho(n))$ labeled samples for f_n, and with probability $1 - \rho(n)$ outputs a circuit $C : \mathbb{F}^\ell \to \mathbb{F}$ such that for every $x \in \mathbb{F}^\ell$ it holds that $\Pr_r[C^{\widetilde{f_n}}(x, r) = f_n(x)] \geq 2/3$.

The first step of the machine M is to invoke the local list-decoding algorithm of [STV01, Thm 29], instantiated with degree parameter $d = d(n)$ and agreement parameter $\rho = \rho(n)$. The algorithm runs in time $\text{poly}(\ell(n), d, \log(q(n)), 1/\rho) = \text{poly}(n, 1/\rho)$ and outputs a list of $O(1/\rho)$ probabilistic oracle circuits $C_1, ..., C_{O(1/\rho)} : \{0,1\}^n \to \{0,1\}^n$ such that with probability at least $2/3$ there exists $i \in [O(1/\rho)]$ satisfying $\Pr[C_i^{\widetilde{f_n}}(x) = f_n(x)] \geq 2/3$ for all $x \in \{0,1\}^n$. We call any circuit that satisfies the latter condition `good`. By invoking the algorithm of [STV01] for $\text{poly}(1/\rho)$ times, we obtain a list of $t = \text{poly}(1/\rho)$ circuits $C_1, ..., C_t$ such that with probability at least $1 - \text{poly}(\rho)$ there exists $i \in [t]$ such that C_i is good.

The second step of the machine is to transform the probabilistic circuits into deterministic circuits such that, with high probability, the deterministic circuit corresponding to the "good" circuit C_i will correctly compute f_n on .99 of the inputs (when given oracle access to $\widetilde{f_n}$). Specifically, by implementing naive error-reduction in all circuits, we can assume that for every $x \in \mathbb{F}^\ell$ it holds that $\Pr_r[C_i^{\widetilde{f_n}}(x, r) = f_n(x)] \geq .995$. Now the machine M creates $O(\log(1/\rho))$ copies of each circuit in the list, and for each copy M "hard-wires" a randomly-chosen fixed value for the circuit's randomness. The result is a list of $t' = \text{poly}(1/\rho)$ *deterministic* circuits $D_1, ..., D_{t'}$ such that with probability $1 - \text{poly}(\rho)$ there exists a circuit D_i satisfying $\Pr_x[D_i^{\widetilde{f_n}}(x) = f_n(x)] \geq .99$.

The third step of the machine M is to "weed" the list in order to find a single circuit D_i that (when given access to $\widetilde{f_n}$) correctly computes f on .95 of the inputs. To do so M iterates over the list, and for each circuit D_j estimates the agreement of $D_j^{\widetilde{f_n}}$ with f_n with error .01 and confidence $1 - \text{poly}(\rho)$, using the random samples.

The final step of the machine M is to use the standard random self-reducibility of the Reed-Muller code to transform the circuit D_i into a probabilistic circuit that

correctly computes f at each input with probability at least $2/3$. Specifically, the probabilistic circuit implements the standard random self-reducibility algorithm for the (q, ℓ, d) Reed-Muller code (see, e.g., [AB09, Thm 19.19]), while resolving its oracle queries using the circuit D_i. The standard algorithm runs in time $\text{poly}(q, \ell, d)$, and works whenever D_i agrees with f_n on at least $1 - \frac{1-d/q}{6} < .95 + d/q$ of the inputs, which holds in our case since $d/q < \delta = o(1)$. ■

4.4 Towards an equivalence between derandomization and circuit lower bounds

4.4.1 The main results

Recall that the Non-Deterministic Exponential-Time Hypothesis (NETH) conjectures that *co*-3SAT (with n variables and $O(n)$ clauses) cannot be solved by non-deterministic machines running in time $2^{\epsilon \cdot n}$ for some $\epsilon > 0$. The motivating observation for our results in this section is that NETH has an unexpected consequence to the long-standing question of whether *worst-case derandomization of $prBPP$ is equivalent to circuit lower bounds against $\mathcal{E}$*. Specifically, recall that two-way implications between derandomization and circuit lower bounds have been gradually developing since the early '90s (for surveys see, e.g., [Oli13; Wil14a]), and that it is a long-standing question whether the foregoing implications can be strengthened to show a *complete equivalence* between the two. One well-known implication of such an equivalence would be that any (worst-case) derandomization of *$prBPP$ necessitates* the construction of PRGs that "fool" non-uniform circuits.[37] Then, being more concrete, the motivating observation for our results in this section is that NETH *implies an affirmative answer* to the foregoing question (and this is not difficult to show; see Section 4.4.2).

Our main contribution is in showing that, loosely speaking, even a *very weak form of* NETH suffices to answer the question of equivalence in the affirmative, and that this weak form of NETH is in some sense *inherent* (see details below). Towards presenting this very weak form, let us define $\mathcal{NTIME}$-uniform circuits:

Definition 4.4.1 ($\mathcal{NTIME}[T]$-uniform circuits)**.** *For $S, T : \mathbb{N} \to \mathbb{N}$, we say that a set $L \subseteq \{0,1\}^*$ can be decided by $\mathcal{NTIME}[T]$-uniform circuits of size S if there exists a non-deterministic machine M that gets input 1^n, runs in time $T(n)$, and satisfies the following:*

[37] The question of equivalence is mostly "folklore", but was mentioned several times in writing. It was asked in [IKW02, Remark 33], who proved an analogous equivalence between non-deterministic derandomization with short advice and circuit lower bounds against non-deterministic classes (i.e., against $\mathcal{NTIME}$; see also [CR20]). It was also mentioned as a hypothetical possibility in [TV07] (referred to there as a "super-Karp-Lipton theorem"). Following the results of [MW18], the question was recently raised again as a conjecture in [Tel19b] (see Conjecture 4.2.4). We note that in the context of uniform "hardness-to-randomness", equivalences between average-case derandomization, lower bounds for uniform classes, and PRGs for uniform circuits have long been known (see [IW98; Gol11]), but these equivalences do not involve circuit lower bounds or standard PRGs.

1. *For every $n \in \mathbb{N}$ there exist non-deterministic choices such that $M(1^n)$ outputs a circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size at most $S(n)$ that decides L_n.*

2. *For every $n \in \mathbb{N}$ and non-deterministic choices, $M(1^n)$ either outputs a circuit $C\colon \{0,1\}^n \to \{0,1\}$ that decides L_n, or outputs $\perp$.*

When we simply say that L can be decided by $\mathcal{NTIME}[T]$-uniform circuits (without specifying a size bound S), we consider the trivial size bound $S(n) = T(n)$.

The hypotheses that will suffice to show an equivalence between derandomization and circuit lower bounds are of the form "$\mathcal{E}$ does not have $\mathcal{NTIME}[T]$-uniform circuits of size $S(n) \ll T(n)$", for values of T and S that will be specified below. In words, this hypothesis rules out a world in which every $L \in \mathcal{E}$ can be computed by *small circuits* that can be *efficiently produced by a uniform* (non-deterministic) *machine*. Indeed, this hypothesis is weaker than the NETH-style hypothesis $\mathcal{E} \not\subseteq \mathcal{NTIME}[T]$, and even than the hypothesis $\mathcal{E} \not\subseteq (\mathcal{NTIME}[T] \cap \mathcal{SIZE}[T])$. We stress that our hypothesis refers to lower bounds for *uniform* models of computation, for which strong lower bounds (compared to those for non-uniform circuits) are already known. (For example, $\mathcal{NP}$ is hard for $\mathcal{NP}$-uniform circuits of size n^k for every fixed $k \in \mathbb{N}$ (see [SW13]), whereas we do not even know if $\mathcal{E}^{\mathcal{NP}}$ is hard for non-uniform circuits of arbitrarily large *linear* size.) The fact that such a weak hypothesis suffices to deduce that derandomization and circuit lower bounds are equivalent can be seen as appealing evidence that the equivalence indeed holds.

Our first result is that if $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of polynomial size (for some $\delta > 0$), then derandomization of $pr\mathcal{BPP}$ in sub-exponential time is equivalent to lower bounds for polynomial-sized circuits against $\mathcal{EXP}$.

Theorem 4.4.2 (NETH $\Rightarrow$ circuit lower bounds are equivalent to derandomization; "low-end" setting)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of arbitrary polynomial size, even infinitely-often. Then,*

$$pr\mathcal{BPP} \subseteq \mathtt{i.o.}pr\mathcal{SUBEXP} \iff \mathcal{EXP} \not\subset \mathcal{P}/\mathrm{poly}\ .$$

Theorem 4.4.2 also scales-up to "high-end" parameter settings, albeit not smoothly, and using different proof techniques (see Section 4.4.4 for details). Nevertheless, an analogous result holds for the extreme "high-end" setting: Under the stronger hypothesis that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\Omega(n)}]$-uniform circuits, we show that $pr\mathcal{BPP} = pr\mathcal{P}$ is equivalent to lower bounds for exponential-sized circuits against $\mathcal{E}$; that is:

Theorem 4.4.3 (NETH $\Rightarrow$ circuit lower bounds are equivalent to derandomization; "high-end" setting)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\delta \cdot n}]$-uniform circuits, even infinitely-often. Then:*

$$pr\mathcal{BPP} = pr\mathcal{P} \iff \exists \epsilon > 0 : \mathcal{DTIME}[2^n] \not\subset \mathtt{i.o.}\mathcal{SIZE}[2^{\epsilon \cdot n}]\ .$$

Remarkably, as mentioned above, hypotheses such as the ones in Theorems 4.4.2 and 4.4.3 actually yield a stronger conclusion, and are also *necessary* for that stronger conclusion. Specifically, the stronger conclusion is that even *non-deterministic derandomization of* $prBPP$ (such as $prBPP \subseteq prNSUBEXP$) yields circuit lower bounds against $\mathcal{E}$, which in turn yield PRGs for non-uniform circuits.

Theorem 4.4.4 ($NTIME$-uniform circuits for $\mathcal{E}$, non-deterministic derandomization, and circuit lower bounds)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ cannot be decided by $NTIME[2^{n^\delta}]$-uniform circuits of arbitrary polynomial size. Then,*

$$prBPP \subseteq prNSUBEXP \implies EXP \not\subset P/\text{poly} \,. \tag{4.4.1}$$

In the other direction, if Eq. (4.4.1) holds, then $\mathcal{E}$ cannot be decided by NP-uniform circuits.

Note that in Theorem 4.4.4 there is a gap between the hypothesis that implies Eq. (4.4.1) and the conclusion from Eq. (4.4.1). Specifically, the hypothesis refers to $NTIME[2^{n^\delta}]$-uniform circuits of polynomial size, whereas the conclusion refers to NP-uniform circuits. By optimizing the parameters, this gap between sub-exponential and polynomial can be considerably narrowed (see Theorem 4.4.17).

4.4.2 Proof overviews

As mentioned in Section 4.4.1, the motivating observation is that NETH implies an equivalence between derandomization and circuit lower bounds; let us start by proving this statement:

Proposition 4.4.5 ("warm-up": a weaker version of Theorem 4.4.2)**.** *Assume that $EXP \not\subset \text{i.o.}NSUBEXP$. Then, $prBPP \subseteq prSUBEXP \iff EXP \not\subset \text{i.o.}P/\text{poly}$.*

Proof. The "$\impliedby$" direction follows (without any assumption) from [BFN+93]. For the "$\implies$" direction, assume that $prBPP \subseteq prSUBEXP$, and assume towards a contradiction that $EXP \subset \text{i.o.}P/\text{poly}$. The latter hypothesis implies (using the Karp-Lipton style result of [BFN+93]) that $EXP \subset \text{i.o.}MA$. Combining this with the former hypothesis, we deduce that $EXP \subset \text{i.o.}NSUBEXP$, a contradiction. ∎

Our proofs of Theorems 4.4.2 and 4.4.3 will follow the same logical structure as the proof of Proposition 4.4.5, and our goal will be to relax the hypothesis $EXP \not\subset \text{i.o.}NSUBEXP$. We will do so by strengthening the Karp-Lipton style result that uses [BFN+93] and asserts that a joint "collapse" hypothesis and derandomization hypothesis implies that EXP can be decided in small non-deterministic time. We will show two different strengthenings, each referring to a different parameter setting: The first strengthening refers to a "low-end" setting, and asserts that if $EXP \subset P/\text{poly}$ and $prBPP \subseteq prSUBEXP$ then EXP has $NSUBEXP$-uniform circuits of polynomial size (see Item (1) of Proposition 4.4.12); and the second strengthening refers to a "high-end" setting, and asserts that if $\mathcal{E} \subset \text{i.o.}SIZE[2^{\epsilon \cdot n}]$ and $prBPP = prP$ then

$\mathcal{E}$ has $\mathcal{NTIME}[2^{O(\epsilon)\cdot n}]$-uniform circuits (see Proposition 4.4.13). The proofs of these two different strengthenings rely on different ideas; for high-level descriptions of the proofs see Sections 4.4.4.2 and 4.4.4.3, respectively.

For context, recall that (as noted by Fortnow, Santhanam, and Williams [FSW09]), the proof of [BFN+93] already supports the stronger result that $\mathcal{EXP} \subset \mathcal{P}/\text{poly} \iff \mathcal{EXP} = \mathcal{OMA}$;[38] and by adding a derandomization hypothesis (e.g., $pr\mathcal{BPP} = pr\mathcal{P}$) we can deduce that $\mathcal{EXP} = \mathcal{ONP}$. Nevertheless, our results above are stronger, because $\mathcal{NP}$-uniform circuits are an even weaker model than $\mathcal{ONP}$: This is since in the latter model the proof is verified on an input-by-input basis, whereas in the former model we only verify *once* that the proof is convincing for *all* inputs. We also stress that some lower bounds for this weaker model (i.e., for $\mathcal{NTIME}$-uniform circuits of small size) are already known: Santhanam and Williams [SW13] proved that for every $k \in \mathbb{N}$ there exists a function in $\mathcal{NP}$ that cannot be computed by $\mathcal{NP}$-uniform circuits of size n^k.

We also note that our proofs actually show that (conditioned on lower bounds for $\mathcal{NTIME}$-uniform circuits against $\mathcal{E}$) even a *relaxed derandomization hypothesis* is already equivalent to the corresponding circuit lower bounds. For example, in the "high-end" setting, to deduce that $\mathcal{E} \not\subset \mathcal{SIZE}[2^{\Omega(n)}]$ it suffices to assume that CAPP on v-bit circuits of size $n = 2^{\Omega(v)}$ can be solved in time $2^{\epsilon \cdot v}$, for a sufficiently small $\epsilon > 0$.[39] For more details, see Section 4.4.5.

Proof of Theorem 4.4.4. The first part of Theorem 4.4.4 asserts that if $\mathcal{E}$ does not have $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of polynomial size, then the conditional statement "$pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP} \implies \mathcal{EXP} \not\subset \mathcal{P}/\text{poly}$" holds. The proof of this statement again follows the logical structure from the proof of Proposition 4.4.5, and relies on a further strengthening of our "low-end" Karp-Lipton style result such that the result only uses the hypothesis that $pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP}$ rather than $pr\mathcal{BPP} \subseteq pr\mathcal{SUBEXP}$.[40]

The second part of Theorem 4.4.4 asserts that if the conditional statement "$pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP} \implies \mathcal{EXP} \not\subset \mathcal{P}/\text{poly}$" holds, then $\mathcal{E}$ does not have $\mathcal{NP}$-uniform circuits. We will in fact prove the stronger conclusion that $\mathcal{E} \not\subseteq (\mathcal{NP} \cap \mathcal{P}/\text{poly})$. (Recall that the class of problems decidable by $\mathcal{NP}$-uniform circuits is a subclass of $\mathcal{ONP} \subseteq \mathcal{NP} \cap \mathcal{P}/\text{poly}$.) The proof itself is very simple: Assume towards a contradiction that $\mathcal{E} \subseteq (\mathcal{NP} \cap \mathcal{P}/\text{poly})$; since $\mathcal{BPP} \subseteq \mathcal{EXP}$, it follows that $pr\mathcal{BPP} \subseteq pr\mathcal{NP}$

[38] The notation $\mathcal{OMA}$ stands for "oblivious" $\mathcal{MA}$. It denotes the class of problems that can be decided by an $\mathcal{MA}$ verifier such that for every input length there is a single "good" proof that convinces the verifier on all inputs in the set (rather than a separate proof for each input); see, e.g., [FSW09; GM15].

[39] Note that the problem of solving CAPP for v-bit circuits of size $n = 2^{\Omega(v)}$ can be trivially solved in time $2^{O(v)} = \text{poly}(n)$, and thus unconditionally lies in $pr\mathcal{P} \cap pr\mathcal{BPTIME}[\tilde{O}(n)]$. The derandomization problem described above simply calls for a *faster* deterministic algorithm for this problem.

[40] Intuitively, in the "low-end" Karp-Lipton result we only need to derandomize probabilistic decisions made by the non-deterministic machine that constructs the circuit, whereas the circuit itself is deterministic; thus, a non-deterministic derandomization hypothesis suffices for this result. See Section 4.4.4.2 for details.

(see the proof of Theorem 4.4.16); and by the hypothesized conditional statement, we deduce that $\mathcal{EXP} \not\subset \mathcal{P}/\mathrm{poly}$, a contradiction. Indeed, the parameter choices in the foregoing proof are far from tight, and (as mentioned after the statement of Theorem 4.4.4) the quantitative gap between the two parts of Theorem 4.4.4 can be considerably narrowed (see Theorem 4.4.17).

4.4.3 An $\mathcal{E}$-complete problem with useful properties

Our proofs will rely on the well-known existence of an $\mathcal{E}$-complete problem $L^{\mathtt{nice}}$ with the following useful properties: The problem $L^{\mathtt{nice}}$ is randomly self-reducible and that has an instance checker with linear-length queries such that both the instance checker and the random self-reducibility algorithm use a linear number of random bits. Let us properly define these notions:

Definition 4.4.6 (instance checkers)**.** *A probabilistic polynomial-time oracle machine* $\mathtt{IC}$ *is an* instance checker *for a set* $L \subseteq \{0,1\}^*$ *if for every* $x \in \{0,1\}^*$ *the following holds:*

1. *(Completeness.)* $\mathtt{IC}^L(x) = L(x)$*, with probability one.*
2. *(Soundness.) For every* $L' \subseteq \{0,1\}^*$ *we have that* $\Pr[\mathtt{IC}^{L'}(x) \notin \{L(x), \perp\}] \leq 1/6$*.* [41]

For $\ell : \mathbb{N} \to \mathbb{N}$*, if for every* $x \in \{0,1\}^*$*, all the oracle queries of* $\mathtt{IC}$ *on input* x *are of length* $\ell(|x|)$*, then we say that* $\mathtt{IC}$ *has* queries of length ℓ*. We will also measure the maximal* number of queries *that* $\mathtt{IC}$ *makes on inputs of any given length.*

Definition 4.4.7 (random self-reducible function)**.** *We say that* $f\colon \{0,1\}^* \to \{0,1\}^*$ *is* randomly self-reducible *if there exists a probabilistic oracle machine* $\mathtt{Dec}$ *that gets input* $x \in \{0,1\}^n$ *and access to an oracle* $g\colon \{0,1\}^n \to \{0,1\}^*$*, runs in time* $\mathrm{poly}(n)$*, makes oracle queries such that each query is uniformly distributed in* $\{0,1\}^n$*, and if for every oracle query* $q \in \{0,1\}^n$ *it holds that* $g(q) = f(q)$*, then* $\mathtt{Dec}^g(x) = f(x)$*.*

In high-level, the problem $L^{\mathtt{nice}}$ is the low-degree extension of an (arbitrary) $\mathcal{E}$-complete problem. The intuition is that since $L^{\mathtt{nice}}$ is a low-degree extension it is randomly self-reducible, and since $L^{\mathtt{nice}}$ is $\mathcal{E}$-complete we can construct an instance checker for it. (Specifically, the instance checker for $L^{\mathtt{nice}}$ simulates a PCP verifier for $L^{\mathtt{nice}}$, and the problem of answering the verifier's queries reduces to $L^{\mathtt{nice}}$, to the verifier's queries can be answered using an oracle to $L^{\mathtt{nice}}$.) Details follow.

Proposition 4.4.8 (an $\mathcal{E}$-complete problem that is random self-reducible and has a good instance checker)**.** *There exists* $L^{\mathtt{nice}} \in \mathcal{DTIME}[\tilde{O}(2^n)]$ *such that:*

1. *Any* $L \in \mathcal{DTIME}[2^n]$ *reduces to* $L^{\mathtt{nice}}$ *in polynomial time with a constant multiplicative blow-up in the input length; specifically, for every* n *there exists* $n' = O(n)$ *such that any* n*-bit input for* L *is mapped to an* n'*-bit input for* $L^{\mathtt{nice}}$*.*

[41] The standard definition of instance checkers fixes the error probability to $1/3$, but we can reduce the error to $1/6$ using standard error-reduction.

2. *The problem* $L^{\mathtt{nice}}$ *is randomly self-reducible by an algorithm* $\mathtt{Dec}$ *that on inputs of length* n *uses* $n + \mathrm{polylog}(n)$ *random bits.*

3. *There is an instance checker* $\mathtt{IC}$ *for* $L^{\mathtt{nice}}$ *that on inputs of length* n *uses* $n + O(\log(n))$ *random bits and makes* $O(1)$ *queries of length* $\ell(n) = O(n)$.

Proof. For a sufficiently small $\delta \leq \eta/7$, let $L^{\mathcal{E}} = \{(\langle M \rangle, x) : M \text{ accepts } x \text{ in } 2^{|x|} \text{ steps}\}$. Let $f_{L^{\mathcal{E}}} : \{0,1\}^* \to \{0,1\}^*$ be the low-degree extension of $L^{\mathcal{E}}$ such that inputs of length n_0 for $L^{\mathcal{E}}$ are mapped to inputs in $\mathbb{F}^m$, where $m = \delta \cdot \frac{n_0}{\lceil \log(n_0) \rceil}$ and $|\mathbb{F}| = 2^{(1/\delta+1)\cdot\lceil \log(n_0)\rceil}$, for a polynomial of individual degree $d = \lceil (n_0)^{1/\delta} \rceil$. Note that $(d+1)^m \geq 2^{n_0}$ (i.e., there is a unique extension of $L^{\mathcal{E}}$ with these parameters), and that $|\mathbb{F}| > m \cdot d$ (i.e., the polynomial is indeed of low degree). Finally, let $L^{\mathtt{nice}}$ be the set of pairs $(z,i) \in \{0,1\}^{m\cdot\log(|\mathbb{F}|)} \times \{0,1\}^{\lceil \mathrm{loglog}(|\mathbb{F}|)\rceil}$, such that $f_{L^{\mathcal{E}}}(z)_i = 1$ (i.e., the i^{th} bit in the binary representation of $f_{L^{\mathcal{E}}}(z) \in \mathbb{F}$ equals one).

Note that $L^{\mathcal{E}}$ is reducible in polynomial time to $f_{L^{\mathcal{E}}}$, which is in turn reducible in polynomial time to $L^{\mathtt{nice}}$; and that inputs of length $n_0 \in \mathbb{N}$ for $L^{\mathcal{E}}$ are mapped to inputs of length $n = m \cdot \log(|\mathbb{F}|) + \lceil \mathrm{loglog}(|\mathbb{F}|)\rceil + 1 < (1+2\delta)\cdot n_0$ for $L^{\mathtt{nice}}$. Thus any $L \in \mathcal{DTIME}[2^n]$ is reducible in polynomial time to $L^{\mathtt{nice}}$ with a multiplicative overhead of at most $1+3\delta$ in the input length. Also note that $L^{\mathtt{nice}} \in \mathcal{DTIME}[\tilde{O}(2^n)]$, since the polynomial $f_{L^{\mathcal{E}}}$ can be evaluated in such time.

Let us now prove that $L^{\mathtt{nice}}$ is randomly self-reducible with at most $(1+\delta)\cdot n$ random bits. Let $\mathtt{Dec}_0$ be the standard random self-reducibility algorithm for $f_{L^{\mathcal{E}}}$, which uses less than n random bits.[42] Given input $(z,i) \in \{0,1\}^{m\cdot\lceil\log(|\mathbb{F}|)\rceil + \lceil\mathrm{loglog}(|\mathbb{F}|)\rceil}$ and oracle access to some $L' \subseteq \{0,1\}^n$, we simulate $\mathtt{Dec}_0$ at input z and with oracle access to a function induced by L' (as detailed below), and then output the i^{th} bit of its answer. Specifically, we initially choose a random permutation π of $\{0,1\}^{\mathrm{loglog}(|\mathbb{F}|)}$, using $\mathrm{polylog}(n) < \delta \cdot n$ random coins, and whenever $\mathtt{Dec}_0$ makes a query $q_1 \in \mathbb{F}^m$, we query L' at all inputs $\{(q_1, q_2)\}_{q_2 \in \{0,1\}^{\lceil \mathrm{loglog}(|\mathbb{F}|)\rceil}}$, ordered according to π, and answer $\mathtt{Dec}_0$ accordingly. Note that each of our queries is uniformly distributed: This is since for every query (q_1, q_2) we have that q_1 is uniform (because $\mathtt{Dec}_0$'s queries are uniform) and that q_2 is uniform and independent from q_1 (because we chose a random π). Also note that if $L'(q_1,q_2) = L^{\mathtt{nice}}(q_1,q_2)$ for every query (q_1,q_2), then each query q_1 of $\mathtt{Dec}_0$ is answered by $f_{L^{\mathcal{E}}}(q_1)$, in which case we output $f_{L^{\mathcal{E}}}(z)_i = L^{\mathtt{nice}}(z,i)$.

Finally, to see that $L^{\mathtt{nice}}$ has an instance checker that uses $n + O(\log(n))$ random bits and issues $O(1)$ queries of length $(2+7\delta)\cdot n$, fix a PCP system for $\mathcal{DTIME}[T]$, where $T(n) = \tilde{O}(2^n)$, with the following specifications: The verifier V runs in polynomial time, uses $n + O(\log(n))$ bits of randomness, issues $O(1)$ queries, and has perfect completeness and soundness error $1/6$; and there is an algorithm P that gets an input $x \in \{0,1\}^n$ and outputs a proof for x in this PCP system (or $\perp$, if $x \notin L$) in deterministic time $\tilde{O}(2^n)$ (for a suitable PCP system, see [BSCG+13, Thm 1]). We will instantiate this PCP system for the set $L_1^{\mathtt{nice}} = \{(z,i,b) : L^{\mathtt{nice}}(z,i) = b\}$, which is in $\mathcal{DTIME}[\tilde{O}(2^n)]$.

[42] Recall that $\mathtt{Dec}_0$ chooses a random vector $\vec{u} \in \mathbb{F}^m$, which requires $m \cdot \log(|\mathbb{F}|) < n$ random bits, and queries its oracle on a set of points on the line corresponding to $\vec{u}$; see, e.g., [Gol08, Sec. 7.2.1.1].

The instance checker IC for $L^{\mathtt{nice}}$ gets input $(z,i) \in \{0,1\}^n$ and simulates the verifier V for $L_1^{\mathtt{nice}}$ on inputs $(z,i,0)$ and $(z,i,1)$. Whenever $V(z,i,b)$ queries its proof at location $j \in [\tilde{O}(2^n)]$, the instance checker IC uses its oracle to try and decide the problem Π at input (z,i,b,j), where $\Pi = \{((z,i,b),j) : P(z,i,b)_j = 1\}$. Specifically, since $\Pi \in \mathcal{DTIME}[\tilde{O}(2^{n/2})] \subseteq \mathcal{DTIME}[\tilde{O}(2^n)]$ it holds that Π reduces to $L^{\mathtt{nice}}$ in polynomial time and with multiplicative blow-up of $1+3\delta$ in the input length; hence, IC reduces $((z,i,b),j)$ to an input for $L^{\mathtt{nice}}$ of length $\ell(n) \leq (1+3\delta)\cdot(2n+1) < (2+7\delta)\cdot n$ and uses its oracle to try and obtain $\Pi((z,i,b),j)$. For $\sigma \in \{0,1\}$, the instance checker IC outputs σ if and only if $V(z,i,\sigma) = 1$ and $V(z,i,1-\sigma) = 0$, and otherwise outputs $\perp$. Note that $\mathtt{IC}^{L^{\mathtt{nice}}}(z,i) = L^{\mathtt{nice}}(z,i)$, with probability one; and that IC errs when given oracle $L' \neq L^{\mathtt{nice}}$ (i.e., $\mathtt{IC}^{L'}(z,i) = 1 - L^{\mathtt{nice}}(z,i)$) only when V accepts $(z,i,1-L^{\mathtt{nice}}(z,i)) \notin L_1^{\mathtt{nice}}$, which happens with probability at most $1/6$ for any L'. ■

4.4.4 Strengthened Karp-Lipton style results

Recall that Babai *et al.* [BFN+93] proved that if $\mathcal{EXP} \subset \mathcal{P}/\mathrm{poly}$ then $\mathcal{EXP} = \mathcal{MA}$; if we also use an additional hypothesis that $pr\mathcal{BPP} = pr\mathcal{P}$, then we can deduce the stronger conclusion $\mathcal{EXP} = \mathcal{NP}$. In the current section we will prove two strengthenings of this result, which further strengthen the foregoing conclusion: Instead of deducing that $\mathcal{EXP} = \mathcal{NP}$, we will deduce that $\mathcal{EXP}$ can be decided by $\mathcal{NTIME}[T]$-uniform circuits of size S, for small values of T, S.

Since we will be repeating some technical non-degeneracy conditions on functions throughout the section, let us define these conditions concisely at this point:

Definition 4.4.9 (size functions and time functions)**.** *We say that $S\colon \mathbb{N} \to \mathbb{N}$ is a* size function *if S is time-computable, increasing, satisfies $S(n) = o(2^n/n)$, and for every $n \in \mathbb{N}$ satisfies $S(n) > n$ and $S(n+1) \leq 2S(n)$. We say that $T\colon \mathbb{N} \to \mathbb{N}$ is a* time function *if T is time-computable, increasing, and for every $n \in \mathbb{N}$ satisfies $T(n) > n$.*

We first prove, in Section 4.4.4.1 a lemma that will be used in one of our proofs; we present this lemma and the underlying question in a separate section since they might be of independent interest. The two strengthened Karp-Lipton style results will be subsequently proved in Sections 4.4.4.2 and 4.4.4.3, respectively.

4.4.4.1 Solving $(1,1/3)$-CAPP using many untrusted CAPP algorithms

Recall that in the problem (α,β)-CAPP, we get as input a description of a circuit, and our goal is to distinguish between circuits with acceptance probability at least $\alpha > 0$ and circuits with acceptance probability at most $\beta > 0$; we also denote $\mathtt{CAPP} = (2/3,1/3)$-CAPP (see Definition 2.4.1). Assume that we want to solve CAPP on an input circuit C of description length n, and that we are guaranteed that an algorithm A solves CAPP on *some* input length (unknown to us) in the interval $[n, S(n)]$, for some function S. This problem arises, for example, if we assume that $pr\mathcal{BPP} \subset \mathtt{i.o.}pr\mathcal{NP}$ (which

implies that CAPP $\in$ i.o.$pr\mathcal{NP}$), and want to derandomize $\mathcal{MA}$ infinitely-often. (This is because when the $\mathcal{MA}$ verifier gets an input of length m, the derandomization of the verifier corresponds to a CAPP problem on some input length $n = m^k$, but we are not guaranteed that the CAPP algorithm works on input length n.)[43] How can we solve this problem?

If we invoke the algorithm A on each input length in the interval $[n, S(n)]$, while feeding it C as input each time (i.e., C is padded up to the appropriate length), then we obtain a variety of answers, and it is not clear a-priori how we can distinguish the correct answer from possibly-misleading ones. In this section we show a solution for this problem in the setting where we only need to solve CAPP with *one-sided error*, and when A solves a problem in $pr\mathcal{BPP}$ that slightly generalizes CAPP. Intuitively, since we only need to solve $(1, 1/3)$-CAPP, it will be possible to *prove* to us that C is not a YES instance (i.e., that C does not accept all of its inputs); and since A solves a problem that slightly generalizes CAPP, we will be able to modify it to an algorithm that is able to provide such a proof when C is not a YES instance. Details follow.

We first define the aforementioned variation of (α, β)-CAPP, denoted pCAPP (for "parametrized CAPP"), in which α and β are specified as part of the input.

Definition 4.4.10 (parametrized CAPP)**.** *In the promise problem* pCAPP$[S, \ell]$*, the input is a triplet* (C, α, β)*, where* C *is a Boolean circuit over* v *variables and of size* $S(v)$ *and* $1 > \alpha > \beta > 0$ *are rational numbers specified with* $\ell(v)$ *bits. The YES instances are such that* $\Pr_x[C(x) = 1] \geq \alpha$ *and the NO instances are such that* $\Pr_x[C(x) = 1] \leq \beta$*.*

Note that if $\ell(v) = O(\log(S(v)))$, then pCAPP$[S, \ell] \in pr\mathcal{BPP}$. (This is since we can uniformly sample ϵ^{-2} inputs for C, where $\epsilon = \beta - \alpha \geq 1/\text{poly}(S(v))$, and estimate $\Pr_x[C(x) = 1]$ with accuracy $(\alpha - \beta)/2$, with high probability). We now show that solving $(1, 1/3)$-CAPP for circuits of size $S(n)$ infinitely-often reduces to solving pCAPP infinitely-often (i.e., on an arbitrary infinite set of input lengths).

Lemma 4.4.11 (solving CAPP with one-sided error on a fixed input length reduces to solving pCAPP on an unknown "close" input length)**.** *For any two size functions* $S^{(n)}, S^{(v)} \colon \mathbb{N} \to \mathbb{N}$ *and time function* $T \colon \mathbb{N} \to \mathbb{N}$*, assume that* pCAPP$[S^{(v)}, \ell] \in$ i.o.$\mathcal{DTIME}[T]$*, where* $\ell(v) = 4 \cdot \log(v)$*. Then, there exists an algorithm* $M^{co\mathcal{RP}}$ *that for infinitely-many values of* $n \in \mathbb{N}$*, when given as input* $(1^n, C)$ *such that* C *a* v*-bit circuit of size at most* $\max\left\{S^{(n)}(n), S^{(v)}(v)\right\}$*, the algorithm* $M^{co\mathcal{RP}}$ *solves* $(1, 1/3)$-CAPP *on* C *in time* $\text{poly}(n) \cdot v \cdot \tilde{O}(S(n)) \cdot T(\tilde{O}(S^{(n)}(n)))$*.*

Proof. Let $\texttt{ql}(S) = \tilde{O}(S)$ such that circuits of size S can be described by strings of length $\texttt{ql}(S)$. For any $n \in \mathbb{N}$, we consider inputs of length $S^{(n)}(n)$ that describe v-bit circuits of size $S^{(v)}(v)$. Let $I_n = [2\texttt{ql}(S^{(n)}(n)), 2\texttt{ql}(S^{(n)}(n+1)) - 1]$, and note that any sufficiently large integer belongs to a unique interval I_n. Let M^{pCAPP} be a time-T algorithm that solves pCAPP$[S^{(v)}, \ell]$ infinitely-often. We will use M^{pCAPP} to construct the following search algorithm:

[43] Also, in this setting the function S represents "how far ahead" (beyond n) we are willing to look in our search for the "good" input length.

Claim 4.4.11.1 (search-to-decision reduction that preserves the input length)**.** *There exists an algorithm F that gets as input $(1^n, C, m)$, where C is a v-bit circuit of size at most $\max\left\{S^{(n)}(n), S^{(v)}(v)\right\}$ and $m \in I_n$, runs in time $\mathrm{poly}(n) \cdot v \cdot T(m)$, and if M^{pCAPP} correctly solves $\mathsf{pCAPP}[S^{(v)}, \ell]$ on input length m and $\Pr_x[C(x) = 1] \leq 1/3$ then $F(1^n, C, m) \in C^{-1}(0)$.*

Proof. In the following we will construct a set of m-bit inputs and run M^{pCAPP} on each of those inputs. Since all of our inputs will be of the form (C, α, β) where α and β can be specified with $4 \cdot \log(v)$ bits, each input will be of size less than $2\mathtt{ql}(S^{O(n)}(n)) \leq m$; we will therefore pad each input to be of length exactly m.

First we run M^{pCAPP} on input $(C, 1/2, 1/3)$, and if M^{pCAPP} accepts then we output 0^v. Otherwise, when M^{pCAPP} rejected, we have that $\Pr_x[C(x) = 1] \leq 1/2$; in this case our goal will be to construct a string in $C^{-1}(0)$, bit-by-bit. Let $\neg C$ be the circuit that computes C and negates the output, let σ_0 be the empty string, and for $i \in [v]$, in iteration i we act as follows:

1. We start with a prefix $\sigma_{i-1} \in \{0,1\}^{i-1}$, and with the guarantee that the circuit $\neg C_{\sigma_{i-1}}$, which is obtained by fixing the first $i-1$ input variables of $\neg C$ to σ_{i-1}, satisfies $\Pr_x[\neg C_{\sigma_{i-1}}(x) = 1] \geq 1/2 - (i-1) \cdot v^{-2}$.

2. We run M^{pCAPP} at input $(\neg C_{\sigma_{i-1}0}, 1/2 - (i-1) \cdot v^{-2}, 1/2 - i \cdot v^{-2})$. If M^{pCAPP} accepts then we define $\sigma_i = \sigma_{i-1}0$, and otherwise we define $\sigma_i = \sigma_{i-1}1$.

3. To see that the guarantee on $\neg C_{\sigma_i}$ is preserved for iteration $i+1$, note that if M^{pCAPP} accepted then $\Pr_x[\neg C_{\sigma_i}(x) = 1] > 1/2 - i \cdot v^{-2}$; and otherwise we have that $\Pr_x[\neg C_{\sigma_{i-1}1}(x) = 0] \leq 1/2 - (i-1) \cdot v^{-2}$, which implies (by the guarantee on $\neg C_{\sigma_{i-1}}$ from the beginning of the iteration) that $\Pr_x[\neg C_{\sigma_i}(x) = 1] \geq 1/2 - (i-1) \cdot v^{-2}$.

After the v iterations we have that $\Pr_x[\neg C_{\sigma_i}(x) = 1] > 0$, and therefore $\sigma_i \in (\neg C^{-1})(1) = C^{-1}(0)$ and we output σ_i. The running time of each iteration is $\mathrm{poly}(n) \cdot v \cdot T(m)$. □

Our algorithm $M^{co\mathcal{RP}}$ runs F at inputs $\{(1^n, C, k)\}_{k \in I_n}$, and evaluates C at the outputs of F; if for some $k \in I_n$ it holds that $C(F(C,k)) = 0$ then $M^{co\mathcal{RP}}$ rejects, and otherwise $M^{co\mathcal{RP}}$ accepts. The running time of $M^{co\mathcal{RP}}$ is $\mathrm{poly}(n) \cdot v \cdot T(2\mathtt{ql}(S^{(n)}(n+1))) \cdot |I_n| = \mathrm{poly}(n) \cdot \tilde{O}(S^{(n)}(n)) \cdot v \cdot T(\tilde{O}(S^{(n)}(n)))$.

Now, fix $n \in \mathbb{N}$ such that for some $m \in I_n$ it holds that M^{pCAPP} decides $\mathsf{pCAPP}[S^{(v)}, \ell]$ on inputs of length m. To see that $M^{co\mathcal{RP}}$ correctly solves $(1, 1/3)$-$\mathtt{CAPP}$ on an input circuit C over v bits of size at most $\max\left\{S^{(n)}(n), S^{(v)}(v)\right\}$, note that if C accepts all its inputs then $M^{co\mathcal{RP}}$ always accepts C; and if C accepts at most $1/3$ of its inputs then for the "good" $m \in I_n$ it holds that $F(1^n, C, m) \in C^{-1}(0)$, in which case $M^{co\mathcal{RP}}$ rejects. ■

4.4.4.2 A strengthened Karp-Lipton style result for the "low-end" setting

To prove our first strengthening of [BFN+93], let $L \in \mathcal{EXP}$, and note that by our assumption $L \in \mathcal{P}/\text{poly}$. Consider an $\mathcal{MA}$ verifier V that gets input 1^n, guesses a circuit $C_L\colon \{0,1\}^n \to \{0,1\}$, and tries to decide if C_L correctly computes $L_n = L \cap \{0,1\}^n$. The key observation is that since this decision problem (of deciding whether or not a given n-bit circuit computes L_n) is in $\mathcal{EXP}$, we can apply the original Karp-Lipton style result of [BFN+93] to it. The latter result implies that there exists an $\mathcal{MA}$ verifier M that decides whether or not C_L computes L_n correctly. Our verifier V guesses C_L and a witness for M, simulates M, and if M confirms that C_L computes L_n then V outputs C_L.

We will derandomize the foregoing $\mathcal{MA}$ verifier in one of two ways. The first relies on a hypothesis of the form $prBPP \subseteq prNSUBEXP$, which immediately implies that $\mathcal{MA} \subseteq NSUBEXP$. The second relies on a hypothesis of the form $prBPP \subset \texttt{i.o.}prSUBEXP$; in this case we derandomize the $\mathcal{MA}$ verifier infinitely-often, relying on the fact that the $\mathcal{MA}$ verifier can be assumed to have perfect completeness [FGM+89] and on Lemma 4.4.11 (which was presented in Section 4.4.4.1). Note that in both cases, the running time of the resulting non-deterministic machine is subexponential, but the size of the output circuit C_L is nevertheless still polynomial.

The following statement and proof generalize the above, using parametrized "collapse" and derandomization hypotheses. Specifically, if we assume that $\mathcal{E} \subset \mathcal{SIZE}[S]$ and that $prBPP$ can be derandomized in time T, we deduce that $\mathcal{E}$ has $\mathcal{NTIME}[T']$ uniform circuits of size $S(n)$, where $T'(n) \approx T(S(S(n)))$.

Proposition 4.4.12 (a strengthened "low-end" Karp-Lipton style result)**.** *There exist two constants $k, k' > 1$ such that for any size function $S\colon \mathbb{N} \to \mathbb{N}$ and time function $T\colon \mathbb{N} \to \mathbb{N}$ satisfying $T(n) \geq n^{k'}$ the following holds. Let $T'(n) = T(\bar{S}(n)))^{O(1)}$ where $\bar{S}(n) = \tilde{O}(S(\tilde{O}(S(n))))$.*

1. *If $\mathcal{DTIME}[2^n] \subset \mathcal{SIZE}[S]$ and $\mathsf{pCAPP}[v^k \cdot S(v), 4 \cdot \log(v)] \in \texttt{i.o.}pr\mathcal{DTIME}[T]$, then any $L \in \mathcal{DTIME}[2^n]$ can be decided on infinitely-many input lengths by $\mathcal{NTIME}[T']$-uniform circuits of size $S(n)$.*
2. *If $\mathcal{DTIME}[2^n] \subset \mathcal{SIZE}[S]$ and $(1, 1/3)\text{-}\texttt{CAPP}[v^k \cdot S(v)] \in pr\mathcal{NTIME}[T]$, then any $L \in \mathcal{DTIME}[2^n]$ can be decided (on all input lengths) by $\mathcal{NTIME}[T']$-uniform circuits of size $S(n)$.*

Proof. We first prove Item (1). Fix $L \in \mathcal{DTIME}[2^n]$, and recall that by our hypothesis $L \in \mathcal{SIZE}[S]$. We define a corresponding problem $L\text{-}\texttt{Ckts}$ as the set of size-S circuits that decide L; that is, denoting by $\texttt{ql}(S) = \tilde{O}(S)$ the description length of size-S circuits, on inputs of length $N = n + \texttt{ql}(S(n))$ we define $L\text{-}\texttt{Ckts}$ by

$$L\text{-}\texttt{Ckts}_N = \{(1^n, C) : |C| = \texttt{ql}(S(n)) \wedge \forall x \in \{0,1\}^n, C(x) = L(x)\} \ ,$$

and on inputs of length N that cannot be parsed as $N = n + \texttt{ql}(S(n))$ we define $L\text{-}\texttt{Ckts}$ trivially. Note that $L\text{-}\texttt{Ckts} \in \mathcal{DTIME}[2^N]$, since we can enumerate the $2^n < 2^{o(N)}$ inputs, and for each $x \in \{0,1\}^n$ compute $C(x)$ and $L(x)$ in time $2^n + \text{poly}(|C|) < 2^{o(N)}$.

Given input 1^n, we first guess a circuit $C_n^{(L)}$ of size $S(n)$, in the hope that $C_n^{(L)}$ decides L_n; note that a suitable circuit exists by our hypothesis. Now we consider the problem of deciding if $x = (1^n, C_n^{(L)}) \in L\text{-}\mathtt{Ckts}$, where $x \in \{0,1\}^{N=n+\mathtt{ql}(S(n))}$. Since $L\text{-}\mathtt{Ckts} \in \mathcal{DTIME}[2^N]$, we can reduce $L\text{-}\mathtt{Ckts}$ to the problem $L^{\mathtt{nice}}$ from Proposition 4.4.8; that is, we compute in time $\mathrm{poly}(N)$ an input $x' \in \{0,1\}^{N'=O(N)}$ for $L^{\mathtt{nice}}$ such that $x \in L\text{-}\mathtt{Ckts} \iff x' \in L^{\mathtt{nice}}$.

Now, let $\bar{N} = \ell(N') = O(N)$, where ℓ is the query length of the instance checker $\mathtt{IC}$ for $L^{\mathtt{nice}}$. We guess another circuit, which is of size $S(2\bar{N})$ and denoted $C_{\bar{N}}^{L^{\mathtt{nice}}} : \{0,1\}^{\bar{N}} \to \{0,1\}$, in the hope that $C_{\bar{N}}^{L^{\mathtt{nice}}}$ decides $L_{\bar{N}}^{\mathtt{nice}}$; again, a suitable circuit exists by our hypothesis.[44] We then construct a circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}} : \{0,1\}^{O(\bar{N})} \to \{0,1\}$ that computes the decision of $\mathtt{IC}$ at input x' and with oracle $C_{\bar{N}}^{L^{\mathtt{nice}}}$, as a function of the $O(\bar{N})$ random coins of $\mathtt{IC}$, and maps the outputs $\{0, \perp\}$ of $\mathtt{IC}$ to 0, and the output 1 of $\mathtt{IC}$ to 1.

Note that the circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ is over $v = O(\bar{N})$ input bits and of size $S^{(n)}(n) \stackrel{\text{def}}{=\!=} \mathrm{poly}(N) \cdot S(2\bar{N})$. Also, measuring the size of $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ as a function of its number of input bits (i.e., of v), the size is upper-bounded by $S^{(v)}(v) \stackrel{\text{def}}{=\!=} v^k \cdot S(v)$, where $k \in \mathbb{N}$ is a sufficiently large universal constant (and we assume without loss of generality that $v \geq 2\bar{N}$). By the properties of the instance checker, and using the fact that a suitable circuit $C_{\bar{N}}^{L^{\mathtt{nice}}}$ for $L_{\bar{N}}^{\mathtt{nice}}$ exists, we have that:

1. If $C_n^{(L)}$ decides L then $x' \in L^{\mathtt{nice}}$, and hence for some guess of $C_{\bar{N}}^{L^{\mathtt{nice}}}$ the circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ will have acceptance probability one.

2. If $C_n^{(L)}$ does not decide L then $x' \notin L^{\mathtt{nice}}$, and hence for all guesses of $C_{\bar{N}}^{L^{\mathtt{nice}}}$ the circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ accepts at most $1/6$ of its inputs.

Using our hypothesis about pCAPP and Lemma 4.4.11, there exists an algorithm $M^{co\mathcal{RP}}$ that for infinitely-many values of $n \in \mathbb{N}$ gets input $(1^n, \mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}})$ and solves $(1, 1/3)\text{-}\mathtt{CAPP}$ on $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ in time $\mathrm{poly}(n) \cdot v \cdot \tilde{O}(S(n)) \cdot T\left(\tilde{O}(S^{(n)}(n)\right)$. We run this algorithm on $(1^n, \mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}})$, and if it accepts (i.e., asserts that the acceptance probability of $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ is larger than $1/3$) then we output the circuit $C_n^{(L)}$; otherwise we output $\perp$.

Note that the size of the circuit that we output is $S(n)$, and that our running time

[44] To see this more formally, let $L^{\mathtt{pad}} = \left\{(x, 1^{O(\log(|x|))}) : x \in L^{\mathtt{nice}}\right\}$. Since $L^{\mathtt{nice}} \in \mathcal{DTIME}[\tilde{O}(2^n)]$, we have that $L^{\mathtt{pad}} \in \mathcal{DTIME}[2^n]$. Using our hypothesis, $L^{\mathtt{pad}}$ on inputs of length $N' = \bar{N} + O(\log(\bar{N}))$ has circuits of size $S(N')$, and these circuits can be converted (by hardwiring the last $N' - \bar{N}$ input bits) to $\bar{N}$-bit circuits for $L^{\mathtt{nice}}$ of size $S(N') < S(2\bar{N})$.

is at most

$$\begin{aligned}\mathrm{poly}(n)\cdot v\cdot\tilde{O}(S(n))\cdot T\left(\tilde{O}(S^{(n)}(n)\right) &= \mathrm{poly}(n)\cdot\tilde{O}(S(n))^2\cdot T\left(\tilde{O}(S(\tilde{O}(S(n))))\right)\\ &\leq T\left(\tilde{O}(S(\tilde{O}(S(n))))\right)^{O(1)}\ ,\end{aligned}$$

where the last inequality relied on the fact that $T(n)\geq n^{k'}$ for a sufficiently large constant k'.

Let us now explain how to prove Item (2). We guess $C_n^{(L)}$ and $C_{\tilde{N}}^{L^{\mathtt{nice}}}$ and construct $\mathtt{IC}_{x'}^{C_{\tilde{N}}^{L^{\mathtt{nice}}}}$ as above. However, instead of using Lemma 4.4.11, we run the hypothesized non-deterministic $(1,1/3)$-$\mathtt{CAPP}[v^k\cdot S(v)]$ machine, denoted $M^{co\mathcal{RP}}$, on input $\mathtt{IC}_{x'}^{C_{\tilde{N}}^{L^{\mathtt{nice}}}}$ (the advantage in the current setting being that, in contrast to the proof of Item (1), the machine $M^{co\mathcal{RP}}$ is guaranteed to work on *all* input lengths). When $C_n^{(L)}$ decides L_n there are some non-deterministic choices that will cause $M^{co\mathcal{RP}}$ to accept, whereas when $C_n^{(L)}$ does not decide L_n, all non-deterministic choices will cause $M^{co\mathcal{RP}}$ to reject. Our running time is $T(\tilde{O}(S^{(n)}(n)))$, which can be bounded as above by $T\left(\tilde{O}(S(\tilde{O}(S(n))))\right)^{O(1)}$. ■

Note that in the proof of Proposition 4.4.12 we did not use the fact that $L^{\mathtt{nice}}$ is randomly self-reducible, but only the facts that $L^{\mathtt{nice}}$ is complete for $\mathcal{E}$ under linear-time reductions (such that all n-bit inputs are mapped to n'-bit inputs, for $n'=O(n)$) and that it has an instance checker with query length $\ell(n)=O(n)$.

4.4.4.3 A strengthened Karp-Lipton style result for the "high-end" setting

The result presented next asserts that if $\mathcal{E}\in\mathcal{SIZE}[S]$ and $pr\mathcal{BPP}$ can be derandomized in time T, then $\mathcal{E}$ has $\mathcal{NTIME}[T']$ uniform circuits (with a trivial size bound of $T'(n)$), where $T'\approx T(S(n))$. The main difference between this result and the result presented in Section 4.4.4.3, other than the differences in parameters, is that for this result we will need to assume that $pr\mathcal{BPP}$ can be derandomized *deterministically*, rather than only non-deterministically.

Let us briefly describe the proof idea. We construct a circuit for an $\mathcal{E}$-complete problem $L^{\mathtt{nice}}$ that has an instance checker and that is randomly self-reducible (see Section 4.4.3 for definitions and details). We guess a circuit $C^{L^{\mathtt{nice}}}$ for $L^{\mathtt{nice}}$, which exists by our "collapse" hypothesis, and randomly check whether or not this circuit "convinces" the instance checker on almost all inputs; if it does, we instantiate the instance checker with $C^{L^{\mathtt{nice}}}$ as an oracle, to obtain a "corrupt" version of $L^{\mathtt{nice}}$, denoted $\tilde{L}$. We then construct a probabilistic circuit C' that decides $L^{\mathtt{nice}}$, with high probability, using the random self-reducibility of $L^{\mathtt{nice}}$ and oracle access to $\tilde{L}$.

Now, under the hypothesis $pr\mathcal{BPP}\subseteq pr\mathcal{DTIME}[T]$, we can derandomize the two probabilistic steps in the foregoing construction. Specifically, we derandomize the probabilistic verification that the circuit $C^{L^{\mathtt{nice}}}$ "convinces" the instance checker

on almost all inputs, and we also derandomize the probabilistic circuit itself (i.e., we actually output a deterministic circuit that constructs the probabilistic circuit C' and applies a deterministic $\mathtt{CAPP}$ algorithm to C'). Details follow.

Proposition 4.4.13 (a strengthened "high-end" Karp-Lipton style result)**.** *There exist two constants $k, k' > 1$ such that for any size function $S\colon \mathbb{N} \to \mathbb{N}$ and time function $T\colon \mathbb{N} \to \mathbb{N}$ the following holds. Assume that $\mathcal{DTIME}[2^n] \subset \mathtt{i.o.}\mathcal{SIZE}[S]$ and that $\mathtt{CAPP}[v^{k'} \cdot S(v)] \in pr\mathcal{DTIME}[T]$. Then any $L \in \mathcal{DTIME}[2^n]$ can be decided on infinitely-many input lengths by $\mathcal{NTIME}[T']$-uniform circuits, where $T'(n) = \tilde{O}\left(T\left(n^k \cdot S(k \cdot n)\right)\right)$.*

Note that the actual hypothesis of Proposition 4.4.13 is weaker than the hypothesis $pr\mathcal{BPP} \in pr\mathcal{DTIME}[T]$, since we only require an algorithm for $\mathtt{CAPP}$ for *large* circuits (i.e., for v-bit circuits of size $\mathrm{poly}(v) \cdot S(v)$).

Proof of Proposition 4.4.13. Fixing any $L \in \mathcal{DTIME}[2^n]$, we prove that there exist $\mathcal{NTIME}[T']$-uniform circuits that solve L infinitely-often. In what follows, it will be important to distinguish between the non-deterministic machine M, and the deterministic circuit $C\colon \{0,1\}^n \to \{0,1\}$ that M constructs. The machine M gets input 1^n and constructs C as follows.

Step 1: Reduce L to $L^{\mathtt{nice}}$. As its first step, the circuit C computes the linear-time reduction from L to the problem $L^{\mathtt{nice}}$ from Proposition 4.4.8; that is, C maps its input $x \in \{0,1\}^n$ into $x' \in \{0,1\}^{n'}$, where $n' = O(n)$, such that $x \in L$ if and only if $x' \in L^{\mathtt{nice}}$.

Step 2: Guess-and-verify a circuit for $L^{\mathtt{nice}}_{\bar{n}}$. Let $\mathtt{IC}$ be the instance checker for $L^{\mathtt{nice}}$ and let $\bar{n} = \ell(n')$ be the length of queries that $\mathtt{IC}$ makes to its oracle on inputs of length n'.

Claim 4.4.13.1. *For infinitely-many input lengths n there exists a circuit $C^{L^{\mathtt{nice}}}_{\bar{n}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ that decides $L^{\mathtt{nice}}_{\bar{n}}$.*

Proof. For every $n \in \mathbb{N}$ let $I_n = [2\alpha \cdot n, 2\alpha \cdot (n+1) - 1]$, where $\alpha \in \mathbb{N}$ is the constant such that $\bar{n} = \ell(n') = \alpha \cdot n$. Note that every sufficiently large integer $m \in \mathbb{N}$ belongs to a unique interval I_n (i.e., $n = \lfloor m/2\alpha \rfloor$). We define L' to be the language that on input length $m \in I_n$ considers only its first $\bar{n} = \alpha \cdot n$ input bits and decides $L^{\mathtt{nice}}_{\bar{n}}$ on those input bits. Since L' on input length m can be decided in time $\tilde{O}(2^n) < 2^m$, by our hypothesis there exist an infinite set $\mathcal{M} \subseteq \mathbb{N}$ of input lengths such that for every $m \in \mathcal{M}$ there exist size-$S(m)$ circuits for L'_m. For every such $m \in I_n$, we hard-wire the last $m - \bar{n}$ input bits (to be all-zeroes), and obtain a circuit of size $S(m) < S(4\alpha \cdot n) = S(4\bar{n})$ that decides $L^{\mathtt{nice}}_{\bar{n}}$. □

Thus, if n is one of the infinitely-many input lengths mentioned in Claim 4.4.13.1, then there exists $C^{L^{\mathtt{nice}}}_{\bar{n}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ that decides $L^{\mathtt{nice}}_{\bar{n}}$. The machine

M non-deterministically guesses such a circuit. We define the corruption of $C_{\bar{n}}^{L^{\mathtt{nice}}}$ by

$$\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathtt{nice}}}) = \Pr_{z\in\{0,1\}^{n'}}\left[\Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) = \perp] > 1/6\right],$$

where the internal probability is over the random choices of the machine $\mathtt{IC}$. Let $\mathtt{Dec}$ be the machine underlying the random self-reducibility of $L^{\mathtt{nice}}$, and let $c \in \mathbb{N}$ such that the number of queries that $\mathtt{Dec}$ makes on inputs of length n' is at most $(n')^c$. Consider the following promise problem Π:

- **The input** is guaranteed to be a circuit $C_{\bar{n}}^{L^{\mathtt{nice}}}: \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$.
- **YES instances:** The circuit $C_{\bar{n}}^{L^{\mathtt{nice}}}$ decides $L_{\bar{n}}^{\mathtt{nice}}$, in which case $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathtt{nice}}}) = 0$.
- **NO instances:** It holds that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathtt{nice}}}) > (n')^{-2c}$.

Now, note that $\Pi \in pr\text{-}co\mathcal{RP}$, since a probabilistic algorithm that gets $C_{\bar{n}}^{L^{\mathtt{nice}}}$ as input can decide whether $C_{\bar{n}}^{L^{\mathtt{nice}}}$ is a YES instance or a NO instance by sampling z's and estimating $\Pr\left[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) = \perp\right]$ for each z. Moreover, using the sampler from Theorem 2.5.7, there is a probabilistic $co\mathcal{RP}$ algorithm for Π that on input $C_{\bar{n}}^{L^{\mathtt{nice}}}: \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ uses $m = O(n)$ random bits and runs in time $\mathrm{poly}(n) \cdot S(4\bar{n})$. [45]

Hence, the problem Π is reducible to an instance of $(1, 1/3)$-$\mathtt{CAPP}$ with a circuit C_Π on $v = O(n)$ input bits and of size $n^{O(1)} \cdot S(4\bar{n}) = v^{O(1)} \cdot S(v)$. The machine M runs the hypothesized $\mathtt{CAPP}[v^{k'} \cdot S(v)]$ algorithm on C_Π, which takes time $T\left(n^{O(1)} \cdot S(O(n))\right)$, and rejects iff the $\mathtt{CAPP}$ algorithm rejects. Thus, from now on we can assume that $C_{\bar{n}}^{L^{\mathtt{nice}}}$ is not a NO instance of Π, or in other words that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathtt{nice}}}) \leq (n')^{-2c}$.

Step 3: Transforming a non-corrupt $C_{\bar{n}}^{L^{\mathtt{nice}}}$ into a probabilistic circuit for L. Given that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathtt{nice}}}) \leq (n')^{-2c}$, the machine M now transforms $C^{L^{\mathtt{nice}}}$ into a probabilistic circuit C' that computes L. In high-level, the circuit C' simulates the random self-reducibility algorithm $\mathtt{Dec}$ for L, while resolving the random queries of $\mathtt{Dec}$ by instantiating the instance checker with oracle $C^{L^{\mathtt{nice}}}$. Details follow.

[45] Specifically, the algorithm uses the sampler from Theorem 2.5.7 (with a sufficiently large $\beta, \gamma > 1$ and sufficiently small $\alpha > 0$) to sample $D = \mathrm{poly}(n)$ strings $z_1, ..., z_D \in \{0,1\}^{n'}$, and then uses this sampler again to sample D strings $r_1, ..., r_D \in \{0,1\}^{n+O(\log(n))}$ to be used as randomness for the machine $\mathtt{IC}$. The algorithm rejects $C_{\bar{n}}^{L^{\mathtt{nice}}}$ if and only if $\Pr_{i\in[D]}\left[\Pr_{j\in[D]}\left[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z, r_j) = \perp\right] \geq .01\right] \geq 1/2(n')^{-2c}$, where $\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z, r_j)$ denotes the simulation of $\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z)$ with the fixed randomness r_j. This algorithm always accepts YES instances. Now, assume that $C_{\bar{n}}^{L^{\mathtt{nice}}}$ is a NO instance, and let us call $z \in \{0,1\}^{n'}$ is $\mathtt{bad}$ if $\Pr\left[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) = \perp\right] \geq 1/6$. By the properties of the sampler, with high probability over the choice of $z_1, ..., z_D$, the fraction of bad z's in our sample is at least $1/2(n')^{-2c}$; and for any (fixed) bad z, the probability that $\Pr_{j\in[D]}\left[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z, r_j) = \perp\right] < .01$ is $\exp(-n)$. Hence, $C_{\bar{n}}^{L^{\mathtt{nice}}}$ will be rejected with high probability. The bound on the algorithm's running time follows from standard quasilinear-time algorithms for the Circuit Eval problem (see, e.g., [LW13, Thm 3.1]) and since $\tilde{O}(S(4\bar{n})) < \mathrm{poly}(n) \cdot S(2\bar{n})$.

Lemma 4.4.13.2 (non-corrupt $C_{\bar{n}}^{L^{\mathtt{nice}}} \Rightarrow$ probabilistic circuit for $L^{\mathtt{nice}}$)**.** *There exists an algorithm that gets as input 1^n and a circuit $C_{\bar{n}}^{L^{\mathtt{nice}}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ such that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathtt{nice}}}) \leq (n')^{-2c}$, and outputs a probabilistic circuit $C'\colon \{0,1\}^{n'} \to \{0,1\}$ of size $\mathrm{poly}(n) \cdot S(4\bar{n})$ that uses $O(n)$ random coins such that for every $x' \in \{0,1\}^{n'}$, with high probability over choice of random coins r for C' it holds that $C'(x',r) = L^{\mathtt{nice}}(x')$.*

Proof. We consider an instantiation of $\mathtt{IC}$ on inputs of length n' and with oracle to $C_{\bar{n}}^{L^{\mathtt{nice}}}$, and as a first step we reduce the error of this algorithm. Let $m = O(n)$ be the number of random bits that $\mathtt{IC}$ uses on inputs of length n'. Consider the following probabilistic algorithm $\hat{\mathtt{IC}}\colon \{0,1\}^{n'} \to \{0,1,\perp\}$. Given input $z \in \{0,1\}^{n'}$, the algorithm $\hat{\mathtt{IC}}$ uses the sampler from Theorem 2.5.7, instantiated for output length m and with accuracy $1/n$, to obtain a sample of $D = \mathrm{poly}(n)$ strings $r_1, \ldots, r_D \in \{0,1\}^m$; then $\hat{\mathtt{IC}}$ outputs the majority vote among the values $\{v_i\}_{i \in [D]}$, where v_i is the output of $\mathtt{IC}$ when instantiated on input z with oracle $C_{\bar{n}}^{L^{\mathtt{nice}}}$ and fixed randomness r_i.

Note that $\hat{\mathtt{IC}}$ uses $O(n)$ random bits and runs in time $\mathrm{poly}(n) \cdot S(4\bar{n})$. We claim that there exists a set $G \subseteq \{0,1\}^{n'}$ of density $1 - (n')^{-2c}$ such that for every $z \in G$, with probability at least $1 - \exp(-n)$ over the randomness of $\hat{\mathtt{IC}}$ it holds that $\hat{\mathtt{IC}}(z) = L^{\mathtt{nice}}(z)$. To see this, let G be the set of z's such that $\Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) = \perp] \leq 1/6$, and recall that the density of G is at least $1 - (n')^{-2c}$. Note that for any $z \in G$ we have that $\Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) = L^{\mathtt{nice}}(z) \geq 2/3$, because $\Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) \neq L^{\mathtt{nice}}(z)] \leq \Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) = \perp] + \Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathtt{nice}}}}(z) = \neg C_{\bar{n}}^{L^{\mathtt{nice}}}(z)] \leq 1/3$. Thus, for any fixed $z \in G$, the probability (over the random choices of $\hat{\mathtt{IC}}$) that the majority vote of the v_i's will not equal $L^{\mathtt{nice}}(z)$ is at most $\exp(-n)$.

Now, consider a probabilistic circuit $C'\colon \{0,1\}^{n'} \to \{0,1\}$ that chooses $O(n)$ random bits to be used as randomness for $\hat{\mathtt{IC}}$, and simulates the random self-reducibility algorithm $\mathtt{Dec}$ on its input $x' \in \{0,1\}^{n'}$, while answering its queries using the algorithm $\hat{\mathtt{IC}}$ with the fixed random bits chosen in advance. Note that the circuit C' is of size $\mathrm{poly}(n) \cdot S(4\bar{n})$. We claim that for every $x' \in \{0,1\}^{n'}$, with high probability $C'(x) = L^{\mathtt{nice}}(x')$. To see this, recall that $\mathtt{Dec}$ makes at most $(n')^c$ queries such that each query is uniformly-distributed, and thus the probability that all queries of $\mathtt{Dec}$ lie in the set G is at least $1 - (n')^{-c}$. Conditioned on this event, for each fixed query z, the probability over choice of randomness for $\hat{\mathtt{IC}}$ that $\hat{\mathtt{IC}}(z)$ does not output $L^{\mathtt{nice}}(z)$ is at most $\exp(-n)$. Hence, by another union-bound, with high probability all the queries of $\mathtt{Dec}$ are answered correctly, in which case $C'(x') = L^{\mathtt{nice}}(x')$. □

Step 4: Derandomizing C'. The non-deterministic machine guessed-and-verified a circuit $C_{\bar{n}}^{L^{\mathtt{nice}}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ such that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathtt{nice}}}) \leq (n')^{-2c}$, and transformed it (using the algorithm from Proposition 4.4.13.2) into a probabilistic circuit C'. The machine M then constructs the final circuit C, which gets input $x \in \{0,1\}^n$ and acts as follows:

1. Computes the reduction from L to $L^{\mathtt{nice}}$ to obtain $x' \in \{0,1\}^{n'}$.

2. Hard-wires x' into C' to obtain a description of a circuit $C'_{x'}\colon \{0,1\}^{O(n)} \to \{0,1\}$ such that $C'_{x'}(r) = C'(x',r)$.

3. Runs the hypothesized $\mathtt{CAPP}[v^{k'} \cdot S(v)]$ algorithm on C'_x and outputs its decision.

Note that C'_x is a circuit with $v = O(n)$ input bits and of size $\mathrm{poly}(n) \cdot S(4\bar{n}) = v^{O(1)} \cdot S(v)$, and therefore for an appropriate choice of constant k', the $\mathtt{CAPP}[v^{k'} \cdot S(v)]$ algorithm distinguishes between the case that C' accepts x' with high probability and the case that C' rejects x' with high probability. Thus, for every $x \in \{0,1\}^n$ it holds that $C(x) = L(x)$. Finally, both the size of the circuit C and the running time of our non-deterministic machine are bounded by $\tilde{O}\Big(T\Big((n^{O(1)} \cdot S(O(n))\Big)\Big)$. ■

4.4.5 Proof of Theorems 4.4.2, 4.4.3, and 4.4.4

We first prove Theorem 4.4.2, which refers to the "low-end" parameter setting: Subexponential-time derandomization of $prBPP$ and lower bounds for polynomial-sized circuits against EXP.

Theorem 4.4.14 (Theorem 4.4.2, restated)**.** *Assume that there exists $\delta > 0$ such that $DTIME[2^n]$ cannot be decided by $NTIME[2^{n^\delta}]$-uniform circuits of an arbitrarily large polynomial size, even infinitely-often. Then, denoting $prSUBEXP = \cap_{\epsilon>0} prDTIME[2^{n^\epsilon}]$, we have that*

$$\cup_c \mathsf{pCAPP}[v^c, 4\cdot\log(v)] \subset \mathtt{i.o.}prSUBEXP \iff EXP \not\subset \mathcal{P}/\mathrm{poly}\,.$$

Proof. Let us first prove the first statement. The "$\Longleftarrow$" direction follows from [BFN+93], relying on the fact that $\cup_c \mathsf{pCAPP}[v^c, 4\cdot\log(v)] \in prBPP$. For the "$\Longrightarrow$" direction, assume that for every $c \in \mathbb{N}$ and every $\epsilon > 0$ it holds that $\mathsf{pCAPP}[v^c, 4\cdot\log(v)] \in \mathtt{i.o.}prDTIME[2^{n^\epsilon}]$. Assuming towards a contradiction that $EXP \subset \mathcal{P}/\mathrm{poly}$, we have that $DTIME[2^n] \subset SIZE[n^c]$ for some $c \in \mathbb{N}$. We use Item (1) of Proposition 4.4.12 with parameters $S(n) = n^c$ and $T(n) = 2^{n^\epsilon}$, where $\epsilon > 0$ is sufficiently small. We deduce that $DTIME[2^n]$ can be decided infinitely-often by $NTIME[T']$-uniform circuits of size n^c, where

$$T'(n) \leq T(\tilde{O}(S(\tilde{O}(S(n)))))^{O(1)} < T(n^{O_c(1)})^{O(1)} = 2^{n^{\epsilon\cdot O_c(1)}}\,,$$

which contradicts our hypothesis if ϵ is sufficiently small. ■

We now prove Theorem 4.4.15, which refers to a "high-end" parameter setting (i.e., faster derandomization and lower bounds for larger circuits). We will in fact show that, conditioned on the hypothesis that $\mathcal{E}$ cannot be decided by $NTIME[2^{\Omega(n)}]$-uniform circuits, even a weaker derandomization hypothesis is already equivalent to circuit lower bounds. For example, instead of assuming that $prBPP = prP$, we will only

need to assume that CAPP for v-bit circuits of size $2^{\Omega(v)}$ can be solved deterministically in time $2^{\alpha \cdot v}$, for some small constant $\alpha > 0$. [46]

Theorem 4.4.15 (Theorem 4.4.3, restated)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\delta \cdot n}]$-uniform circuits even infinitely-often. Then:*

1. *There exists a universal constant $c > 1$ such that*

$$\exists \epsilon > 0 : \texttt{CAPP}[2^{\epsilon \cdot v}] \in pr\mathcal{DTIME}[n^{(\delta/c)/\epsilon)}] \iff \exists \epsilon > 0 : \mathcal{E} \not\subset \texttt{i.o.}\mathcal{SIZE}[2^{\epsilon \cdot n}] \,.$$

2. *For every fixed constant $c > 1$ it holds that*

$$\exists \alpha > 1 : \texttt{CAPP}[2^{v^{1/c}}] \in pr\mathcal{DTIME}[2^{\alpha \cdot (\log n)^c}] \iff \exists \epsilon > 0 : \mathcal{E} \not\subset \texttt{i.o.}\mathcal{SIZE}[2^{\epsilon \cdot n^{1/c}}] \,.$$

Proof. We first prove Item (1). The "$\Longleftarrow$" direction follows from [IW99] (or, alternatively, from the more general Corollary 2.4.8). Specifically, the hypothesized circuit lower bound implies that $pr\mathcal{BPP} = pr\mathcal{P}$, and in particular that $\texttt{CAPP} \in pr\mathcal{DTIME}[n^{c'}]$ for some $c' \in \mathbb{N}$. The conclusion then holds for $\epsilon < \frac{\delta}{c \cdot c'}$. For the "$\Longrightarrow$" direction, let $k, k' \in \mathbb{N}$ be as in Proposition 4.4.13, and let $c = 2k$. Assume that for some $\epsilon > 0$ it holds that $\texttt{CAPP}[S'] \in pr\mathcal{DTIME}[T]$, where $T(n) = n^{(\delta/c)/\epsilon)}$, and $S(n) = 2^{\epsilon \cdot n}/n^{k'}$, and $S'(v) = v^{k'} \cdot S(v) = 2^{\epsilon \cdot v}$. Assuming towards a contradiction that $\mathcal{E} \subset \texttt{i.o.}\mathcal{SIZE}[S]$, Proposition 4.4.13 implies that $\mathcal{DTIME}[2^n]$ can be decided infinitely-often by $\mathcal{NTIME}[T']$-uniform circuits, where $T'(n) = \tilde{O}\left(T(n^k \cdot S(k \cdot n))\right) < 2^{\delta \cdot n}$; this is a contradiction.

The proof of Item (2) is similar. The "$\Longleftarrow$" follows from Corollary 2.4.8, instantiated with $S(n) = 2^{\epsilon \cdot n^{1/c}}$, to deduce that $\texttt{CAPP} \in pr\mathcal{DTIME}[T]$ for $T(n) = 2^{\Delta \cdot S^{-1}(n^\Delta)} = 2^{(\Delta/\epsilon)^c \cdot (\log n)^c}$. For the "$\Longrightarrow$" direction, let $\epsilon < (\delta/k\alpha)^{1/c}$ be sufficiently small, let $S(n) = 2^{\epsilon \cdot n^{1/c}}/n^{k'}$, let $S'(v) = v^{k'} \cdot S(v) = 2^{v^{1/c}}$, and let $T(n) = 2^{\alpha \cdot (\log n)^c}$. We use Proposition 4.4.13 as above, and rely on the fact that $T'(n) = \tilde{O}\left(T(n^k \cdot S(k \cdot n))\right) < 2^{\delta \cdot n}$. ■

Next, we prove Theorem 4.4.4, which asserts that if non-deterministic derandomization implies lower bounds against $\mathcal{EXP}$, then $\mathcal{EXP}$ does not have $\mathcal{NP}$-uniform circuits. We will actually prove a stronger result: First, we will use a weaker hypothesis than in Theorem 4.4.4, namely that $pr\mathcal{BPP} \subseteq pr\mathcal{NP}$ implies circuit lower bounds against $\mathcal{EXP}$; and secondly, we will deduce the stronger conclusion that $\mathcal{EXP} \not\subseteq (\mathcal{NP} \cap \mathcal{P}/\text{poly})$. (This conclusion is stronger because the class of problems decidable by $\mathcal{NP}$-uniform circuits is a subclass of $\mathcal{NP} \cap \mathcal{P}/\text{poly}$.)

[46] This is reminiscent of the recent results of Murray and Williams [MW18], who showed that solving CAPP for v-bit circuits of size $2^{\Omega(v)}$ in time $2^{.99 \cdot v}$ suffices to deduce circuit lower bounds. Note that the foregoing CAPP problem can be solved in *deterministic* polynomial time, since the input length is $2^{\Omega(v)}$ (i.e., this CAPP problem lies in $pr\mathcal{BPTIME}[\tilde{O}(n)] \cap pr\mathcal{P}$).

Theorem 4.4.16 (Theorem 4.4.4, restated)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ does not have $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of an arbitrarily large polynomial size. Then,*

$$pr\mathcal{BPP} \subset pr\mathcal{NSUBEXP} \implies \mathcal{EXP} \not\subset \mathcal{P}/\text{poly} \,, \tag{4.4.2}$$

where $pr\mathcal{NSUBEXP} = \cap_{\epsilon>0} pr\mathcal{NTIME}[2^{n^\epsilon}]$. In the other direction, if Eq. (4.4.2) holds,[47] *then $\mathcal{EXP} \not\subseteq (\mathcal{NP} \cap \mathcal{P}/\text{poly})$, and in particular $\mathcal{EXP}$ does not have $\mathcal{NP}$-uniform circuits.*

Proof. The proof of the first statement is similar to the proof of Theorem 4.4.14. We assume that $\mathcal{EXP} \subset \mathcal{P}/\text{poly}$, and use Item (2) of Proposition 4.4.12 with parameters $S(n) = n^c$ and $T(n) = 2^{n^\epsilon}$, where $\epsilon > 0$ is sufficiently small; we deduce that any $L \in \mathcal{E}$ can be decided on all input lengths by $\mathcal{NTIME}[T']$-uniform circuits of size n^c, where $T'(n) < 2^{O(n^{3\epsilon \cdot c})} < 2^{n^\delta}$, which is a contradiction (the last inequality relied on $\epsilon > 0$ being sufficiently small).

To prove the "in the other direction" statement, first recall that $pr\mathcal{EXP} \subseteq pr(\mathcal{NP} \cap \mathcal{P}/\text{poly}) \iff \mathcal{EXP} \subseteq (\mathcal{NP} \cap \mathcal{P}/\text{poly})$, because every exponential-time machine that solves a promise problem also induces a language.[48] Now, assume towards a contradiction that $pr\mathcal{EXP} \subseteq pr(\mathcal{NP} \cap \mathcal{P}/\text{poly})$. Since $pr\mathcal{BPP} \subseteq pr\mathcal{EXP}$, we have that $pr\mathcal{BPP} \subseteq pr(\mathcal{NP} \cap \mathcal{P}/\text{poly})$. By the hypothesized conditional statement, it follows that $\mathcal{EXP} \not\subset \mathcal{P}/\text{poly}$, a contradiction. ■

As mentioned in the introduction, by optimizing the parameters we can show tighter two-way implications between the statement "derandomization and lower bounds are equivalent" and the statement "$\mathcal{E}$ does not have $\mathcal{NTIME}[T]$-uniform circuits". Towards proving this result, we define the following class of growth functions, which lie "in between" quasipolynomial functions and subexponential functions. For every two constants $k, c \in \mathbb{N}$, we denote by $\mathsf{e}^{(k,c)} : \mathbb{N} \to \mathbb{N}$ the function that applies k logarithms to its input, raises the obtained expression to the power c, and then takes k exponentiations of this expression. For example, $\mathsf{e}^{(1,c)}(n) = 2^{(\log n)^c}$ and $\mathsf{e}^{(2,c)}(n) \in 2^{2^{\text{polyloglog}(n)}}$. Note that $\mathsf{e}^{(k+1,c)}$ grows asymptotically faster than $\mathsf{e}^{(k,c')}$ for any constants c, c', and that $\mathsf{e}^{(k,c)}$ is smaller than any subexponential function. Then, we have that:

[47] In fact, for this statement it suffices to assume that $pr\mathcal{BPP} \subseteq pr\mathcal{NP} \implies \mathcal{EXP} \not\subset \mathcal{P}/\text{poly}$. However, since we will show a result with tighter relations between the parameters below (see Theorem 4.4.17), in the current statement we ignore this issue for simplicity.

[48] In more detail, the "$\implies$" direction is trivial, so we prove the "$\impliedby$" direction. For every $\Pi \in pr\mathcal{EXP}$, let M be an exponential-time machine that solves Π, and let L_M be the set of inputs that M accepts. Since $L_M \in \mathcal{EXP}$, there exists an $\mathcal{NP}$-machine that decides L_M and a polynomial-sized circuit family that decides L_M, and the foregoing machine and circuit family also solve Π.

Theorem 4.4.17 (Theorem 4.4.4, a tighter version)**.** *For any constant $k \in \mathbb{N}$ we have that:*

$$\exists \delta > 0 : \mathcal{DTIME}[2^n] \text{ does not have } \mathcal{NTIME}[T]\text{-uniform circuits, for } T = 2^{\mathsf{e}^{(k,\delta)}} \tag{4.4.3}$$

$$\Downarrow$$

$$pr\mathcal{BPP} \subseteq \cap_{\epsilon>0} pr\mathcal{NTIME}[2^{\mathsf{e}^{(k,\epsilon)}}] \implies \mathcal{DTIME}[2^n] \not\subset \cup_{c_0 \in \mathbb{N}} \mathcal{SIZE}[\mathsf{e}^{(k,c_0)}] \tag{4.4.4}$$

$$\Downarrow$$

$$\forall c_0 \in \mathbb{N}, \mathcal{DTIME}[2^n] \not\subset (\mathcal{NTIME}[T] \cap \mathcal{SIZE}[T]), \text{ for } T(n) = \mathsf{e}^{(k,c_0)} \tag{4.4.5}$$

that is, statement (4.4.3) *implies statement* (4.4.4)*, which in turn implies statement* (4.4.5)*.*

We stress that the gap between the values of T in statements (4.4.3) and (4.4.5) is substantial, but nevertheless much smaller than an exponential gap. This is since in statement (4.4.3) the hypothesis is for T that is exponential in $\mathsf{e}^{(k,\delta)}$ where $\delta > 0$ is an *arbitrarily small constant*, whereas in statement (4.4.5) the conclusion is for $T = \mathsf{e}^{(k,c_0)}$ where c_0 is an *arbitrarily large constant*. For example, for $k = 1$ this is the difference between quasipolynomial functions and functions of the form $2^{2^{(\log n)^\epsilon}} \ll 2^{n^\epsilon}$.

Proof of Theorem 4.4.17. To see that statement (4.4.3) implies statement (4.4.4), first observe that for any two constants $c, c' \in \mathbb{N}$ it holds that $(\mathsf{e}^{(k,c)})^{-1}(n) = \mathsf{e}^{(k,1/c)}(n)$ and that $\mathsf{e}^{(k,c)}(\mathsf{e}^{(k,c')}(n)) = \mathsf{e}^{(k,cc')}(n)$. Now, assuming that $pr\mathcal{BPP} \subseteq \cap_\epsilon pr\mathcal{NTIME}[2^{\mathsf{e}^{(k,\epsilon)}}]$ and that $\mathcal{DTIME}[2^n] \subset \cup_{c_0} \mathcal{SIZE}[\mathsf{e}^{(k,c_0)}]$, we will show that Eq. (4.4.3) does not hold. To do so we use Item (2) of Proposition 4.4.12 with $S(n) = \mathsf{e}^{(k,c_0)}$ and with $T(n) = 2^{\mathsf{e}^{(k,\epsilon)}}(n)$ for a sufficiently small $\epsilon > 0$, and rely on the fact that for some $b \in \mathbb{N}$ it holds that $T'(n) < T(S(S(n)^b)^b)^b < T(\mathsf{e}^{(k,2b^2 \cdot c_0)}(n))^b = 2^{\mathsf{e}^{(k,2\epsilon b^3 \cdot c_0)}(n)}$.

To see that statement (4.4.4) implies statement (4.4.5), assume towards a contradiction that for some $c_0 \in \mathbb{N}$ it holds that $pr\mathcal{DTIME}[2^n] \subseteq pr(\mathcal{NTIME}[T] \cap \mathcal{SIZE}[T])$, where $T(n) = \mathsf{e}^{(k,c_0)}(n)$. Hence, $\texttt{CAPP} \in \mathcal{DTIME}[\tilde{O}(2^n)] \subseteq pr(\mathcal{NTIME}[T(\tilde{O}(n))] \cap \mathcal{SIZE}[T(\tilde{O}(n))])$, and it follows that

$$\begin{aligned} pr\mathcal{BPP} &\subseteq \cup_{c \in \mathbb{N}} pr\mathcal{NTIME}[T(n^c)] \\ &\subseteq \cup_{c \in \mathbb{N}} pr\mathcal{NTIME}\left[\mathsf{e}^{(k,c)}\right] \\ &\subseteq \cap_{\epsilon>0} pr\mathcal{NTIME}\left[2^{\mathsf{e}^{(k,\epsilon)}}\right] . \end{aligned}$$

By our hypothesis (i.e., by Eq. (4.4.4)) it follows that $\mathcal{DTIME}[2^n] \not\subset \cup_{c_0 \in \mathbb{N}} \mathcal{SIZE}\left[\mathsf{e}^{(k,c_0)}\right]$, which is a contradiction. Finally, to deduce the statement (i.e. bridge the gap between $pr\mathcal{DTIME}[2^n]$ and $\mathcal{DTIME}[2^n]$), we use the same argument as in Footnote 48. ■

Appendix A

Brief Descriptions of Several Other Works

In this appendix we provide high-level descriptions of several results that were obtained during our doctoral studies, but that were not included above. Specifically, Section A.1 is based on a joint work with Oliveira and Santhanam [OST19], Section A.2 is based on the work [Tel18a], and Section A.3 is based on the work [Tel20].

A.1 Expander-based cryptography meets natural proofs

In a joint work with Oliveira and Santhanam [OST19], we introduce new forms of attack on *expander-based cryptography*, and in particular on Goldreich's pseudorandom generator and one-way function. Our attacks exploit *low circuit complexity* of the underlying expander's neighbor function and/or of the local predicate. Our two key conceptual contributions are:

1. The security of Goldreich's PRG and OWF hinges, at least in some settings, on the *circuit complexity* of the underlying expander's neighbor function and of the local predicate. This sharply diverges from previous works, which focused on the *expansion* properties of the underlying expander and on the *algebraic* properties of the predicate.

2. We uncover new connections between long-standing open problems: Specifically, we tie *the security of Goldreich's PRG and OWF* both to the existence of *unbalanced lossless expanders* with low-complexity neighbor function, and to *limitations on circuit lower bounds* (i.e., natural proofs).

We prove two types of technical results that support the above conceptual messages. First, we *unconditionally break Goldreich's PRG* when instantiated with a specific expander (whose existence we prove), for a class of predicates that match the parameters of the currently-best "hard" candidates, in the regime of quasi-polynomial stretch. Secondly, conditioned on the existence of expanders whose neighbor functions have

extremely low circuit complexity, we present attacks on Goldreich's generator in the *regime of polynomial stretch*. As one corollary, conditioned on the existence of the foregoing expanders, we show that either the parameters of natural properties for several constant-depth circuit classes *cannot be improved, even mildly*; or Goldreich's generator is insecure in the regime of a large polynomial stretch, *regardless of the predicate used*.

In particular, our results further motivate the investigation of average-case lower bounds against DNF-XOR circuits of exponential size, and of the parameters that can be achieved by affine/local unbalanced expanders.

A.2 Lower bounds on black-box reductions of hitting to density estimation

Consider a deterministic algorithm that tries to find a string in an unknown set $S \subseteq \{0,1\}^n$, under the promise that S has large density. The only information that the algorithm can obtain about S is *estimates of the density of* S in adaptively chosen subsets of $\{0,1\}^n$, up to an additive error of $\mu > 0$. This problem is appealing as a derandomization problem, when S is the set of satisfying inputs for a circuit $C : \{0,1\}^n \to \{0,1\}$ that accepts many inputs: In this context, an algorithm as above constitutes a deterministic black-box reduction of the problem of *hitting* C (i.e., finding a satisfying input for C) to the problem of *approximately counting* the number of satisfying inputs for C on subsets of $\{0,1\}^n$.

In the work [Tel18a] we prove tight lower bounds for this problem, demonstrating that naive approaches to solve the problem cannot be improved upon, in general. First, we show a tight trade-off between the estimation error μ and the required number of queries to solve the problem: When $\mu = O(\log(n)/n)$ a polynomial number of queries suffices, and when $\mu \geq 4 \cdot (\log(n)/n)$ the required number of queries is $2^{\Theta(\mu \cdot n)}$. Secondly, we show that the problem "resists" parallelization: Any algorithm that works in iterations, and can obtain $p = p(n)$ density estimates "in parallel" in each iteration, still requires $\Omega\left(\frac{n}{\log(p)+\log(1/\mu)}\right)$ iterations to solve the problem.

This work extends the well-known work of Karp, Upfal, and Wigderson [KUW88], who studied the setting in which S is only guaranteed to be non-empty (rather than dense), and the algorithm can only probe subsets for the *existence* of a solution in them. In addition, our lower bound on parallel algorithms affirms a weak version of a conjecture of Motwani, Naor, and Naor (1994); we also make progress on a stronger version of their conjecture.

A.3 A note on tolerant testing with one-sided error

A tolerant tester with one-sided error for a property is a tester that accepts every input that is close to the property, with probability 1, and rejects every input that is far from the property, with positive probability. In the work [Tel20] we show that such testers require a linear number of queries.

Appendix B

Surveys and Expositions of Known Results

In this appendix we include several surveys and expositions of known results that were initially published in our personal website during our doctoral studies. The following expositions and surveys are included:

1. In Section B.1 we present an overview of the lower bound proof of Li, Razborov, and Rossman [LRR17] for computing subgraph isomorphism in $\mathcal{AC}^0$.

2. In Section B.2 we describe a short and almost elementary proof of Williams' [Wil13] result that "non-trivial" derandomization implies weak circuit lower bounds.

3. In Section B.3 we describe the proof of the results of Bazzi, Razborov, and Braverman [Baz09; Raz09; Bra10], which assert that polylog-wise independent distributions "fool" $\mathcal{AC}^0$ circuits.

4. In Section B.4 we include a survey of Karp-Lipton theorems, with a focus on the underlying technical challenges and proof techniques.

5. In Section B.5 we show that significantly better sub-exponential lower bounds for $\mathcal{AC}^0$ would imply lower bounds for polynomial-sized circuits from classes such as $\mathcal{AC}^0[\mathtt{Sym}]$ or $\mathcal{NC}^1$.

B.1 Overview of the lower bound of Li, Razborov, and Rossman for subgraph isomorphism in $\mathcal{AC}^0$

The following overview is based on the lecture series of Ben Rossman in the Swedish Summer School of Computer Science (S3CS), 2017. It was aided by many helpful answers to my questions by Ben and by Igor Oliveira. Nevertheless, the overview reflects my own understanding of the lower bound proof, and neither Ben nor Igor should be held accountable for any mistakes.

B.1.1 The lower bound: Bird's eye

In 2008, Ben Rossman proved [Ros08] that for any constant $k = O(1)$, any $\mathcal{AC}^0$ circuit that solves the k-clique problem has $n^{\Omega(k)}$ gates. This improved a decades-old lower bound of $n^{\Omega(k/d^2)}$ by Beame, mainly by removing the dependency on the circuit depth d. The new techniques that were introduced in Rossman's work led to a sequence of works proving lower bounds for $\mathcal{AC}^0$ circuits solving the more general subgraph isomorphism problem. In this text I will give an overview of the lower bound for the subgraph isomorphism problem that was proved by Li, Razborov, and Rossman [LRR17]. The reason to survey the more general result of [LRR17] (rather than the original result in [Ros08] for k-clique) is that in this case the abstraction and generalization seem to distill and clarify the underlying ideas.

The lower bound of [LRR17] is parametrized, according to the subgraph in question. That is, for the G-subgraph isomorphism problem, where G is of constant size, the lower bound asserts that $\mathcal{AC}^0$ circuits need $n^{\kappa(G)}$ gates to solve the problem, where κ is a graph-theoretic parameter. In particular, for k-clique (and for many other graphs on k vertices), $\kappa(G) = \Omega(k)$. The lower bound also extends to circuits solving the problem in average-case, under specific distributions, and to circuits of super-constant depth, up to depth $o(\log(n)/\log\log(n))$. However, for simplicity, I will focus on the lower bound for circuits of constant depth that solve the problem in the worst-case.

Of course, sub-exponential lower bounds for $\mathcal{AC}^0$ circuits that compute the parity function in worst-case and in average-case have been known for decades. However, the parity function can be easily computed in circuit classes larger than $\mathcal{AC}^0$ (e.g., in $\mathcal{NC}^1$), whereas the subgraph isomorphism problem is NP-complete. This raises the possibility that the lower bound on subgraph isomorphism can be extended further to circuit classes larger than $\mathcal{AC}^0$.

Moreover, one of the main innovations in this line of work is an interesting technique: Specifically, the core part of the proof shows that we can, in some very loose sense, identify the structure of any $\mathcal{AC}^0$ circuit solving the G-subgraph isomorphism problem with the structure of the graph G itself. This seems exciting, since it gives some kind of intuition as to the structure of circuits solving this problem.

B.1.2 The colorful subgraph isomorphism problem

The lower bound itself is actually for $\mathcal{AC}^0$ circuits solving the *colorful subgraph isomorphism problem*, which is a somewhat contrived variation of the (standard) subgraph isomorphism problem. In general, the colorful version is at least as hard as the standard version, but in many cases (e.g., in the case where G is a clique) the problems are essentially equivalent for circuits; see details below. Let us therefore start by properly defining the colorful subgraph isomorphism problem.

The colorful subgraph isomorphism problem is parametrized by a fixed graph G over $k = O(1)$ vertices. For every $n \in \mathbb{N}$, consider the "blow-up" version of G, denoted by $G^{\uparrow n}$, which is defined as follows. First, replace every vertex $v \in G$ with a "cloud" of n vertices; thus, each vertex in $G^{\uparrow n}$ can be described by a pair (v, i) where $v \in G$ and

$i \in [n]$. Two vertices (u,i) and (v,j) in $G^{\uparrow n}$ are connected iff u and v are connected in G; put otherwise, there are no edges within each cloud in $G^{\uparrow n}$, and two clouds form a biclique iff the corresponding two vertices in G are connected. More formally:

Definition B.1.1 (the "blow-up" version of G)**.** *For a graph G over $k = O(1)$ vertices and $n \in \mathbb{N}$, let $G^{\uparrow n}$ be the graph with vertex-set $V(G^{\uparrow n}) = \{(v,i) : v \in V(G) \wedge i \in [n]\}$ and edge-set $E(G^{\uparrow n}) = \{((u,i),(v,j)) : (u,v) \in E(G) \wedge i,j \in [n]\}$.*

The name of the problem comes from thinking of each "cloud" in $G^{\uparrow n}$ as a "color-class", and of the vertices in $G^{\uparrow n}$ as colored in k distinct colors. In the colorful subgraph isomorphism problem, we are given as input a subgraph $X \subseteq G^{\uparrow n}$, and we need to decide whether or not X contains a "distinctly-colored" copy of G. That is:

Definition B.1.2 (distinctly-colored subgraphs of $G^{\uparrow n}$)**.** *We say that a subgraph G' of $G^{\uparrow n}$ is* distinctly-colored *if for every two vertices (u,i) and (v,j) of G' it holds that $u \neq v$.*

Definition B.1.3 (colorful subgraph isomorphism problem)**.** *For a graph G over $k \in \mathbb{N}$ vertices, the* colorful subgraph isomorphism problem *corresponding to G, denoted by $SUB(G)$, is the following: For every $n \in \mathbb{N}$, given a subgraph $X \subseteq G^{\uparrow n}$ as input, decide whether or not X contains a distinctly-colored subgraph that is isomorphic to G.*

Note that for every $n \in \mathbb{N}$, the input to $SUB(G)$ is of length $\binom{k}{2} \cdot n^2$. The standard subgraph isomorphism problem can be randomly reduced to the colorful version, with constant success probability, by randomly coloring the vertices of an input graph in k distinct colors.[1] On the other hand, for a large class of graphs, the colorful problem is deterministically reducible to the standard problem; specifically, this holds for any graph G such that any homomorphism $G \to G$ is bijective (i.e., an automorphism).[2]

B.1.3 An overview of the proof

Fix some constant-sized graph G; for convenience, one may think of G being the k-clique. A natural strategy to try and prove a lower bound for $SUB(G)$ is to try and emulate the lower bound proof for parity: That is, construct a distribution on restrictions that, on one hand, simplifies every $\mathcal{AC}^0$ circuit to the constant function, with high probability (say, 0.9); and on the other hand, keeps the function $SUB(G)$ alive, with high probability (again, say, 0.9).

The proof follows by showing a distribution $\boldsymbol{\rho}$ over restrictions with similar properties: On the one hand, for any $\mathcal{AC}^0$ circuit C of sufficiently small size (i.e., less than $n^{\kappa(G)}$, when κ is the graph-theoretic parameter that was mentioned in Section B.1.1), with high probability over $\rho \sim \boldsymbol{\rho}$ it holds that $C|_\rho$ is *insensitive* to *some of the living*

[1] Specifically, given an arbitrary graph X, randomly color its vertices with k colors, and remove edges within each color-class. Indeed, if X contains a copy of G, then with probability at least k^{-k} we obtain a subgraph of $G^{\uparrow n}$ that contains a distinctly-colored copy of G.

[2] Given an input $X \subseteq G^{\uparrow n}$ to the colorful problem, observe that every copy G' of G in X is distinctly-colored: This is because any coloring of G' is a homomorphism, and is thus an automorphism. Thus, we can reduce the colorful problem to the standard problem by simply ignoring the coloring.

variables; and on the other hand, with constant probability over $\rho \sim \boldsymbol{\rho}$, the function $SUB(G)\restriction_\rho$ remains *sensitive to all of the living variables*.

B.1.3.1 The distribution over restrictions

The distribution $\boldsymbol{\rho}$ over restrictions will satisfy three properties, which I will now detail. Recall that each input variable indicates whether or not a corresponding edge of $G^{\uparrow n}$ is included in X. The first property of the distribution is the following:

1. Each restriction in the distribution's support leaves exactly $|E(G)|$ variables alive, which correspond to the edges of some distinctly-colored copy of G.

The second property of the distribution will imply that with probability $\Omega(1)$ over $\rho \sim \boldsymbol{\rho}$, the function $SUB(G)\restriction_\rho$ remains sensitive to all of the living variables. Intuitively, we want that the subgraph that corresponds to the fixed variables under ρ will not contain any distinctly-colored copy of G, and that the only way to add a distinctly-colored copy of G to this subgraph will be to add *all the edges* that correspond to the living variables. This requirement can be phrased as follows:

2. With probability $\Omega(1)$, if we fix all the living variables under ρ to one, then there will be a unique distinctly-colored copy of G in the graph.

Indeed, the unique distinctly-colored copy of G mentioned in the second requirement is simply the copy of G that corresponds to the living variables. The first two properties imply that with probability $\Omega(1)$ it holds that $SUB(G)\restriction_\rho$ is just the AND function, and in particular is sensitive to all of the living variables.

The third property is that any "sufficiently small" $\mathcal{AC}^0$ circuit becomes insensitive to some of the living variables under $\rho \sim \boldsymbol{\rho}$:

3. For any $\mathcal{AC}^0$ circuit C of size $n^{\kappa(G)-\Omega(1)}$, where κ is a function that I will formally define later on, with probability $1 - o(1)$ over $\rho \sim \boldsymbol{\rho}$ it holds that $C\restriction_\rho$ is insensitive to some of the living input variables.

Indeed, at first glance the third property seems quite weak: After all, we are fixing all but $O(1)$ of the variables! However, the proof of the third property is far from being a simple application of Håstad's switching lemma. This is the case because neither the choice of variables to keep alive nor the choice of values for the fixed variables are uniform. (The variables that will be kept alive correspond to the edges of a distinctly-colored copy of G, whereas the first two properties suggest, at least intuitively, that the vast majority of the fixed variables will be fixed to zero.)

For any G such that we are able to design a distribution that satisfies the three properties above, we can obtain a corresponding lower bound for $\mathcal{AC}^0$ circuits computing $SUB(G)$: Every sufficiently small $\mathcal{AC}^0$ circuit becomes insensitive to some of the living variables under ρ, with high probability, whereas the function $SUB(G)$ remains sensitive to all of the living variables under ρ, with probability $\Omega(1)$.

B.1.3.2 Constructing a distribution with seemingly-weaker properties

The first step in the proof is to construct a distribution $\boldsymbol{\rho}$ that satisfies Properties (1) and (2), and also satisfies a property that is seemingly-weaker than Property (3). Later on (in the next section) we will see that any such distribution in fact also satisfies Property (3). The distribution $\boldsymbol{\rho}$ will be defined using the notion of a "threshold function" θ, which is defined as follows.

Definition B.1.4 (threhold function; see [LRR17, Def. 2.8])**.** *A* threshold function for a graph *G is a function $\theta : E(G) \to [0,2]$ that satisfies the following properties:*

1. *For every subgraph H of G is holds that $\sum_{e \in E(H)} \theta(e) \leq |V(H)|$.*
2. *$\sum_{e \in E(G)} \theta(e) = |V(G)|$.*

As an example, for any r-regular graph G, the constant function $\theta \equiv 2/r$ is a threshold function. This is the case because for any subgraph $H \subseteq G$ it holds that $\sum_{e \in E(H)} \theta(e) = \frac{2}{r} \cdot |E(H)| \leq |V(H)|$, and it also holds that $\sum_{e \in E(G)} \theta(e) = |V(G)|$. Given a threshold function θ for G, we can now define a corresponding distribution $\boldsymbol{\rho}$:

Definition B.1.5 (the distribution $\boldsymbol{\rho}$)**.** *Given a graph G and a threshold function θ for G, the distribution $\boldsymbol{\rho}$ on restrictions for functions $\{0,1\}^{|E(G^{\uparrow n})|} \to \{0,1\}$ is defined as follows:*

1. *Randomly choose a distinctly-colored copy of G in $G^{\uparrow n}$. The variables that correspond to the edges of this copy of G will be kept alive.*
2. *For every other variable x_i, let e be the edge of G that x_i corresponds to (i.e., x_i corresponds to an edge of $G^{\uparrow n}$, which corresponds to a unique edge $e \in E(G)$). Then, fix x_i to one with probability $n^{-\theta(e)}$, and to zero otherwise.*

Observe that Property (1) holds by the definition of $\boldsymbol{\rho}$ (we keep alive exactly $|E(G)|$ variables, corresponding to the edges of a distinctly-colored copy of G).

Recall that Property (2) asserts that with probability $\Omega(1)$, if we fix all the living variables under ρ to one, there will be a unique distinctly-colored copy of G in the graph. To get some intuition as to why this property holds, let us count the *expected* number of distinctly-colored copies of G when *fixing all the variables* to values chosen as in Item (2) of Definition B.1.5. The number of potential distinctly-colored copies of G in $G^{\uparrow n}$ is $n^{|V(G)|}$, and each copy exists in the graph with probability $\prod_{e \in E(G)} n^{-\theta(e)} = n^{-\sum_{e \in E(G)} \theta(e)} = n^{-|V(G)|}$ (the last equality is since θ is a threshold function). Thus, in expectation, there is exactly one distinctly-colored copy of G when fixing all variables. In particular, if the variance of the RV "the number of distinctly-colored copies of G" is not too small, then the probability that this RV takes the value zero is constant. For a full proof that Property (2) holds, see [LRR17, Lem. B.1.3, Lem. 2.10, Apdx. A.].

Property (3) asserts that any "sufficiently small" $\mathcal{AC}^0$ circuit C becomes insensitive to some of the living input variables under $\boldsymbol{\rho}$, with high probability. As mentioned, the first step is to show a seemingly-weaker property of $\boldsymbol{\rho}$: Namely, that by *fixing a few additional variables* after applying $\boldsymbol{\rho}$, any $\mathcal{AC}^0$ circuit becomes insensitive to some of the living variables, with high probability. We first need the following definitions:

Definition B.1.6 ("fixing a few more variables")**.** *Let G be a graph, let $\sigma \in \{0,1\}^{|E(G)|}$, and let $H \subseteq G$. Then, we denote by χ_H^σ the restriction that fixes the variables that correspond to $E(G) \setminus E(H)$ to values according to the corresponding bits in σ, and leaves all the variables corresponding to $E(H)$ alive.*

Fix a function f over variables that correspond to the edge-set of a graph G. We say that f is sensitive to a subgraph $H \subseteq G$ if f is sensitive to all the variables that correspond to $E(H)$. The following definition refines this notion by imposing a stricter requirement: Intuitively, f is strongly-sensitive to H if f *remains sensitive to the variables* that correspond to $E(H)$ even *after we fix all the other variables* (i.e., after fixing the variables that correspond to $E(G) \setminus E(H)$). More formally,

Definition B.1.7 (strong sensitivity)**.** *Let G be a graph, and let f be a function whose input variables correspond to the edge-set of G. For any subgraph $H \subseteq G$ and $\sigma \in \{0,1\}^{|E(G)|}$, we say that f is σ-strongly-sensitive to H if $f\restriction_{\chi_H^\sigma}$ is sensitive to all of the living input variables.*

The seemingly-weaker property that we will start from is that for some distribution $\boldsymbol{\sigma}$ over $\{0,1\}^{|E(G)|}$, for "many" subgraphs $H \subseteq G$, the probability that $C\restriction_{\boldsymbol{\rho}}$ is $\boldsymbol{\sigma}$-strongly-sensitive to H is very small.

3̃. There exists a non-negative function Δ on the set of subgraphs of G and a distribution $\boldsymbol{\sigma}$ over $\{0,1\}^{|E(G)|}$ such that:

- For any fixed $H \subseteq G$ and any $\mathcal{AC}^0$ circuit C, the probability over $\rho \sim \boldsymbol{\rho}$ and $\sigma \sim \boldsymbol{\sigma}$ that $C\restriction_\rho$ is σ-strongly sensitive to H is at most $n^{-\Delta(H)}$.
- Informally, we want that for "many" subgraphs $H \subseteq G$ it holds that $\Delta(H)$ is "large" (e.g., $\Delta(H) > 1$).

I will be more formal as to the second item in the next section, after presenting some necessary definitions. For the moment, let us see that Property (3̃) holds for the distribution $\boldsymbol{\rho}$ that was defined above with respect to a *specific function* Δ and *specific distribution* $\boldsymbol{\sigma}$ over $\{0,1\}^{|E(G)|}$:

Definition B.1.8 (excess function; see [LRR17, Def. 2.8(i) with $\alpha \equiv 1$ and $\beta = \theta$ and $\Delta = \alpha - \beta$])**.** *Given a graph G and a threshold function θ for G, we define the following function $\Delta = \Delta_\theta$ on subgraphs H of G: For any H it holds that $\Delta(H) = |V(H)| - \sum_{e \in E(H)} \theta(e)$.*

The distribution $\boldsymbol{\sigma}$ over $\{0,1\}^{|E(G)|}$ is obtained by fixing values to each edge similarly to Item (2) of Definition B.1.5 (i.e., for every $e \in E(G)$, the corresponding bit in σ is set to one with probability $n^{-\theta(e)}$). The proof of the first item of Property (3̃) appears in [LRR17, Lem. 3.10]. To get some intuition, fix an $\mathcal{AC}^0$ circuit C, and consider the following process of generating the restriction $\chi_H^\sigma \circ \boldsymbol{\rho}$. First, we apply a restriction $\boldsymbol{\rho}_1$ that fixes all but $n^{\Omega(1)}$ of the variables to values that are chosen as in Item (2) of Definition B.1.5. Then, with overwhelmingly high probability, the restricted circuit $C\restriction_{\rho_1}$ depends on at most n^δ variables, where $\delta > 0$ can be made arbitrarily small.[3]

[3]The choice of variables to be kept alive is not uniform, and again relies the threshold function θ. The analysis of the effect of $\boldsymbol{\rho}_1$ relies on Håstad's switching lemma as well as on the fact that values for the fixed variables in $\boldsymbol{\rho}_1$ are chosen independently.

We then apply a second restriction $\boldsymbol{\rho}_2$, in which we choose a random distinctly-colored copy of H within the subgraph that corresponds to the living variables under $\boldsymbol{\rho}_1$, and fix all the variables except the ones corresponding to the edges of this copy of H (again, to values that are chosen as in Item (2) of Definition B.1.5). Observe that $\boldsymbol{\rho}_2 \circ \boldsymbol{\rho}_1$ is essentially distributed identically to $\chi_H^\sigma \circ \boldsymbol{\rho}$. Also, the only case in which $C\restriction_{\boldsymbol{\rho}_2 \circ \boldsymbol{\rho}_1}$ is sensitive to all the living variables is if the variables corresponding to the copy of H that were left alive *are all in the set of at most* n^δ *variables that* $C\restriction_{\boldsymbol{\rho}_1}$ *depended on*.

Given suitable parameters for the distribution $\boldsymbol{\rho}_1$, the number of copies of H inside the subgraph corresponding to the living variables under $\boldsymbol{\rho}_1$ is extremely likely to be approximately $n^{\Delta(H)-\delta\cdot|E(H)|}$. On the other hand, the number of copies of H inside the subgraph corresponding to the variables that $C\restriction_{\boldsymbol{\rho}_1}$ depends on is at most $\binom{n^\delta}{|E(H)|} < n^{\delta\cdot|E(H)|}$. Thus, the probability that $C\restriction_{\boldsymbol{\rho}_2\circ\boldsymbol{\rho}_1}$ is sensitive to all the living variables is at most $n^{\Delta(H)-2\cdot|E(H)|\cdot\delta}$, where $\delta > 0$ is arbitrarily small.

As for the second requirement in Property ($\tilde{3}$), observe that this requirement actually depends on the *specific choice of* θ. In fact, anticipating ahead, for any graph G we will want to construct θ such that Δ satisfies this requirement, and the final lower bound will depend quantitatively on this choice. In the next section I will define this requirement more formally, and in Section B.1.4 I will describe a nice example of a graph G and a corresponding threshold function for which this is true.

B.1.3.3 Deducing Property (3) from Property ($\tilde{3}$)

The key thing that is left to prove is that any distribution that satisfies Properties (1) and ($\tilde{3}$) also satisfies Property (3). The proof of this claim will contain the technique that was mentioned in Section B.1.1 (of relating the structure of any $\mathcal{AC}^0$ circuit that computes $SUB(G)$ to the structure of G). To begin, let us define a combinatorial object that is called a *pattern* for the graph G. Loosely speaking, a pattern is a procedure to construct G in which the initial "building-blocks" are the edges of G, and in each step we combine (i.e., take a union of) two existing "building-blocks". More formally:

Definition B.1.9 (patterns; see [LRR17, Def. 2.11 of "union sequences"])**.** *A* `pattern` *for a graph* G *is a labeled binary tree that satisfies the following properties:*

1. *Each leaf in the tree is labeled by an edge of* G.
2. *Each non-leaf node in the tree is labeled by the subgraph of* G *obtained from the union of the labels its children.*
3. *The root of the tree is labeled by* G.

As an illustrating example, note for any graph G, a natural pattern is the complete binary tree in which the set of leaves corresponds exactly to the set of edges. However, Definition B.1.9 also allows for less natural patterns, in which an edge might appear in many leaves, leaves might appear in different levels of the tree, etc.

Now, fix any circuit C whose input variables correspond to the edge-set of $G^{\uparrow n}$, and any restriction ρ that keeps alive a set of variables that correspond to the edge-set of G. Our main goal is to *relate the structure of the circuit $C\restriction_\rho$ to a pattern for G*. To do so, we first convert C to a circuit C' in which each gate has fan-in at most 2; this is done by replacing each gate in C with a binary tree, in the natural way (i.e., a gate with fan-in m is converted to a binary tree of depth $\lceil \log(m) \rceil$). We say that $C'\restriction_\rho$ *contains a σ-strongly-sensitive pattern* for G if there exists a pattern P for G such that we can associate each node v in P with a gate g of $C'\restriction_\rho$ that is σ-strongly-sensitive to $\texttt{Label}(v)$.

Definition B.1.10 (associating circuits with patterns)**.** *Let $C'\restriction_\rho$ be a circuit whose input variables correspond to the edge-set of a graph G. For $\sigma \in \{0,1\}^{|E(G)|}$, we say that $C'\restriction_\rho$* contains a σ-strongly-sensitive pattern for *G if there exists a pattern P for G and a mapping Φ from the nodes of P to the gates of $C'\restriction_\rho$ that satisfies the following: For every node v in P it holds that $\Phi(v)$ is σ-strongly-sensitive to $\texttt{Label}(v)$.*

In other words, each node v in the pattern, which is labeled with $H_v = \texttt{Label}(v)$, is associated with a gate g of $C'\restriction_\rho$ that satisfies the following: When fixing all the variables except for the ones corresponding to $E(H_v)$, using the values specified in σ, the gate g still remains sensitive to all of the $|E(H_v)|$ living variables. The crucial observation in the proof is that every circuit $C'\restriction_\rho$ that is sensitive to all its input variables contains a strongly-sensitive pattern for G:

Proposition B.1.11 (see [LRR17, Lem. 3.7])**.** *Let $C'\restriction_\rho$ be a circuit whose input variables correspond to the edges of a graph G, and whose gates have fan-in at most two. If $C'\restriction_\rho$ is sensitive to all of its input variables, then for every $\sigma \in \{0,1\}^{|E(G)|}$ it holds that $C'\restriction_\rho$ contains a σ-strongly-sensitive pattern for G.*

Proof. Assume that $C'\restriction_\rho$ is sensitive to all of its input variables, and fix $\sigma \in \{0,1\}^{|E(G)|}$. We prove the following claim: For every gate g in $C'\restriction_\rho$ and every non-empty subgraph H such that g is σ-strongly-sensitive to H it holds that g contains a σ-strongly-sensitive pattern for H. The proposition follows by applying the claim to the top gate of $C'\restriction_\rho$ with $H = G$. We prove the claim by induction on the depth of g. The base case is when g is a variable; in this case H is a single edge (and g contains a σ-strongly-sensitive pattern for this edge).

For the induction step, let g be a gate of fan-in at most two, and let $H \subseteq G$ be a graph to which g is σ-strongly-sensitive. If there exists a child g' of g such that g' is σ-strongly-sensitive to H, then by the induction hypothesis g' contains a σ-strongly-sensitive pattern for H, and thus g also contains this pattern.

Otherwise, $g = g_1 \wedge g_2$ or $g = g_1 \vee g_2$. For any function f, denote by $H^{\texttt{sens}}(f)$ the graph whose edges correspond to the set of variables that f is sensitive to. For $i \in \{1,2\}$, let $H_i = H^{\texttt{sens}}(g_i\restriction_{\chi^\sigma_H})$, and observe that g_i is σ-strongly-sensitive to H_i. [4] By

[4] In general, any function f is σ-strongly-sensitive to $H^{\texttt{sens}}(f)$, and also σ-strongly-sensitive to $H^{\texttt{sens}}(f\restriction_{\chi^\sigma_H})$ for any H.

the induction hypothesis, for $i \in \{1,2\}$ it holds that g_i contains a σ-strongly-sensitive pattern P_i for H_i. Let P be the pattern whose top gate is connected to P_1 and to P_2, and observe that the top node in P is labeled with $H_1 \cup H_2 = H$, where the equality is due to the following: First, for $i \in \{1,2\}$ it holds that $H^{\mathtt{sens}}(g_i\restriction_{\chi_H^\sigma}) \subseteq H$, and second, $H = H^{\mathtt{sens}}(g\restriction_{\chi_H^\sigma}) \subseteq H^{\mathtt{sens}}(g_1\restriction_{\chi_H^\sigma}) \cup H^{\mathtt{sens}}(g_2\restriction_{\chi_H^\sigma})$. Finally, to see that g contains P, extend the mappings Φ_1 and Φ_2 of H_1 and H_2 to a mapping Φ in the natural way (i.e., map the top node of P to g). ■

Using Proposition B.1.11, we can now prove that any distribution that satisfies Properties (1) and (3̃) also satisfies Property (3). In fact, we can now get rid of the informal parts in Property (3̃); that is, we can replace the requirement that for "many" subgraphs $H \subseteq G$ it holds that $\Delta(H)$ is "large" by a formal requirement, as follows:

Lemma B.1.12 (main lemma)**.** *Let G be a graph on $k = O(1)$ vertices, let Δ be a non-negative function on the subgraphs of G, let $\boldsymbol{\rho}$ be a distribution that satisfies Property (1), and let $\boldsymbol{\sigma}$ be a distribution $\{0,1\}^{|E(G)|}$. Assume that:*

1. *For any fixed $H \subseteq G$ and any $\mathcal{AC}^0$ circuit C, the probability over $\rho \sim \boldsymbol{\rho}$ and $\sigma \sim \boldsymbol{\sigma}$ that $C\restriction_\rho$ is σ-strongly-sensitive to H is at most $n^{-\Delta(H)}$.*

2. *There exists a number $\kappa > 0$ such that in any pattern for G there exists a node labeled by a subgraph H satisfying $\Delta(H) \geq \kappa$.*

Then, for any $\mathcal{AC}^0$ circuit C with $n^{\kappa/2-\Omega(1)}$ gates, with probability $1 - o(1)$ over choice of $\rho \sim \boldsymbol{\rho}$ it holds that $C\restriction_\rho$ is insensitive to some of the living input variables.

Proof. For a constant $\epsilon > 0$, let C be a circuit with $n^{\kappa/2-\epsilon}$ gates and constant depth. Let C' be the circuit that is obtained by replacing every gate in C by a corresponding binary tree (such that every gate in C' is of fan-in at most 2). Since C has at most $n^{\kappa/2-\epsilon}$ gates, the number of gates in C' is at most $n^{\kappa-2\epsilon}$.

Let $\mathcal{E}$ be the event that $C\restriction_\rho$ is sensitive to all of its input variables. We start to upper-bound the probability over $\rho \sim \boldsymbol{\rho}$ of $\mathcal{E}$, using the following claim.

Claim B.1.12.1. *Fix any choice of $\rho \sim \boldsymbol{\rho}$ such that $\mathcal{E}$ happens. Then, for every fixed choice of $\sigma \sim \boldsymbol{\sigma}$, there exists a gate g of C' and a subgraph $H \subseteq G$ such that $\Delta(H) \geq \kappa$ and $g\restriction_\rho$ is σ-strongly-sensitive to H.*

Proof. Fix ρ such that $\mathcal{E}$ happens, and fix any $\sigma \in \{0,1\}^{|E(G)|}$. By Proposition B.1.11 it holds that $C'\restriction_\rho$ contains a σ-strongly-sensitive pattern for G. By our hypothesis, there exists a node v in the pattern that is labeled by H such that $\Delta(H) \geq \kappa$. Thus, the mapping Φ between the pattern and the circuit yields a gate $g' = \Phi(v)$ of $C'\restriction_\rho$ such that g' is σ-strongly-sensitive to H. Finally, any gate g' in $C'\restriction_\rho$ is of the form $g' = g\restriction_\rho$, for some gate g of C'. □

Let $\mathcal{H}_\kappa$ be the set of subgraphs $H \subseteq G$ such that $\Delta(H) \geq \kappa$, let $\mathcal{G}_{C'}$ be the set of gates of C', and let $\mathcal{E}_\sigma(g, H)$ be the event that gate g is σ-strongly-sensitive to H. Then, Claim B.1.12.1 implies that

$$\begin{aligned}\Pr_{\rho\sim\boldsymbol{\rho}}[\mathcal{E}] &\leq \Pr_{\rho\sim\boldsymbol{\rho}}\left[\forall\sigma\in\{0,1\}^{|E(G)|}\ \ \exists g\in\mathcal{G}_{C'}, H\in\mathcal{H}_\kappa : \mathcal{E}_\sigma(g,H)\right] \\ &\leq \Pr_{\rho\sim\boldsymbol{\rho},\sigma\sim\boldsymbol{\sigma}}[\exists g\in\mathcal{G}_{C'}, H\in\mathcal{H}_\kappa : \mathcal{E}_\sigma(g,H)] \\ &\leq \sum_{g\in\mathcal{G}_{C'}}\sum_{H\in\mathcal{H}_\kappa}\Pr_{\rho\sim\boldsymbol{\rho},\sigma\sim\boldsymbol{\sigma}}[\mathcal{E}_\sigma(g,H)]\ . \end{aligned} \tag{B.1.1}$$

Now, recall that each gate $g \in \mathcal{G}_{C'}$ computes an $\mathcal{AC}^0$ function. Relying on the hypotheses of the lemma, each summand in Eq. (B.1.1) is upper-bounded by $n^{-\Delta(H)} \leq n^{-\kappa}$. Thus, Eq. (B.1.1) is upper-bounded by $n^{\kappa-2\epsilon}\cdot 2^{|E(G)|}\cdot n^{-\kappa} = O(n^{-2\epsilon})$. ■

B.1.4 The main theorem, and an example

We are now ready to define the graph-theoretic parameter $\kappa = \kappa(G)$, and to state and prove the lower bound of [LRR17] using this definition. Loosely speaking, Sections B.1.3.2 and B.1.3.3 imply the following: If, for some $\kappa > 0$, we are able to design a threshold function θ such that in any pattern for G there exists a node labeled by a subgraph such that $\Delta_\theta(H) \geq \kappa$, then $\mathcal{AC}^0$ circuits that compute $SUB(G)$ have more than $n^{\kappa/2-\Omega(1)}$ gates. This naturally gives rise to the following definition of the parameter $\kappa = \kappa(G)$:

Definition B.1.13 (the parameter κ; see [LRR17, Def. 2.12(ii)])**.** *For any graph G and threshold function θ for G, let $\kappa(G,\theta)$ be the maximal value such that in any pattern for G there exists a node labeled with subgraph H satisfying $\Delta_\theta(H) \geq \kappa$.* [5] *Let $\kappa = \kappa(G)$ be the maximum, over all threshold functions θ for G, of $\kappa(G,\theta)$.*

Theorem B.1.14 (main theorem)**.** *For any graph G on $k = O(1)$ vertices, $SUB(G)$ cannot be computed by $\mathcal{AC}^0$ circuits with $n^{\kappa(G)/2-\Omega(1)}$ gates.*

Proof. By Lemma B.1.12, for any $\mathcal{AC}^0$ circuit C with $n^{\kappa(G)/2-\Omega(1)}$ gates it holds that $C\restriction_\rho$ is insensitive to some of its input variables, with probability $1-o(1)$ over choice of $\rho\sim\boldsymbol{\rho}$. However, by Property (2) it holds that $SUB(G)\restriction_\rho$ is sensitive to all of the input variables, with probability $\Omega(1)$. ■

Thus, to obtain a lower bound for $\mathcal{AC}^0$ circuits computing $SUB(G)$, it suffices to lower bound $\kappa(G)$. As an illustrating example, let us consider the graph $G = K_k$ that is the clique on k vertices, and show that $\kappa(G) \geq \Omega(k)$. As a threshold function we use the constant function $\theta \equiv 2/(k-1)$. Indeed, θ satisfies the two requirements from a threshold function: For any subgraph H of K_k, we have that $\sum_{e\in E(H)}\theta(e) \leq \binom{|V(H)|}{2}\cdot\frac{2}{k-1} \leq |V(H)|$, and $\sum_{e\in E(G)}\theta(e) = k$.

[5] Equivalently, $\kappa(G,\theta) = \min_{P\in Patterns(G)}\left\{\max_{H\in\{\text{labels of nodes in } P\}}\{\Delta_\theta(H)\}\right\}$.

Observe that in any pattern for G, there exists a node labeled with a subgraph H over $j = |V(H)|$ vertices such that $k/3 \leq j \leq 2k/3$. For any such subgraph H, it holds that $\Delta(H) \geq \Delta(K_j) = j - \binom{j}{2} \cdot \frac{2}{k-1} = j \cdot \left(1 - \frac{j-1}{k-1}\right) > \frac{j}{k} \cdot \left(1 - \frac{j}{k}\right) \cdot k \geq 2k/9$. Therefore, $\kappa(G) \geq 2k/9$, and we obtain the following corollary of Theorem B.1.14:

Corollary B.1.15 ($\mathcal{AC}^0$ lower bounds for k-clique)**.** *Any $\mathcal{AC}^0$ circuit that computes $SUB(K_k)$ has at least $n^{\Omega(k)}$ gates.*

B.2 Non-trivial derandomization implies weak lower bounds: An (almost) elementary proof

A derandomization algorithm for a circuit class $\mathcal{C}$ is a deterministic algorithm that gets as input a circuit $C \in \mathcal{C}$, and distinguishes between the case that the acceptance probability of C is one and the case that the acceptance probability of C is less than half. (Indeed, in this text we focus on derandomization of circuits with one-sided error.) Ryan Williams' fundamental result [Wil13] asserts that for many circuit classes $\mathcal{C}$, any non-trivial derandomization of $\mathcal{C}$ implies weak lower bounds for $\mathcal{C}$.

In Williams' original result, "weak lower bounds" means that $\mathcal{C}$ does not contain the (large) class $\mathcal{NEXP}$. The current text presents an alternative argument, in which "weak lower bounds" means that $\mathcal{C}$ does not contain the (also large) class $\mathcal{E}^{\mathcal{NP}}$. [6] This alternative argument is already implicit in [Wil13, Lem. 3.2 & Thm. 3.5], and it also appears in [BSV14, Thm 1.4].[7] The advantages of this alternative argument are that it applies to more circuit classes $\mathcal{C}$, and that the proof is short and almost completely self-contained. In fact, the only non-elementary part in the proof is a black-box use of a (known) PCP construction.

As in the original result, to obtain lower bounds for a class $\mathcal{C}$ of circuits of some fixed size and depth, one needs a non-trivial derandomization algorithm for a class $\widehat{\mathcal{C}}$ of circuits of polynomially-larger size and of somewhat larger depth. The exact overhead in the proof (i.e., the difference between $\mathcal{C}$ and $\widehat{\mathcal{C}}$) is flexible, since it depends on various technical details. Since these details are of secondary importance, I will first state the main theorem slightly informally, without fully defining what $\widehat{\mathcal{C}}$ is with respect to $\mathcal{C}$, and elaborate on this point immediately after the proof.

Theorem B.2.1 (non-trivial derandomization implies weak lower bounds; informal)**.** *Let $\mathcal{C}$ be a class of circuits of size less than $2^{n/4}$, and let $\widehat{\mathcal{C}}$ be a class of circuits that are "slightly larger" than circuits in $\mathcal{C}$. Assume that there exists a deterministic algorithm D that, when given as input a circuit $C \in \widehat{\mathcal{C}}$ over n input bits, distinguishes in time $2^n/n^{\omega(1)}$ between the case that C accepts all of its inputs and the case that C rejects most of its inputs. Then, there exists a function $f \in \mathcal{E}^{\mathcal{NP}}$ such that $f \notin \mathcal{C}$.*

[6] As far as I know, the two results are incomparable, since I don't know of any containment $\mathcal{E}^{\mathcal{NP}} \subseteq \mathcal{NEXP}$ or $\mathcal{NEXP} \subseteq \mathcal{E}^{\mathcal{NP}}$. Note that both classes are contained in $\mathcal{EXP}^{\mathcal{NP}}$.

[7] The work of Ben-Sasson and Viola [BSV14] allowed to extend the alternative argument to significantly more circuit classes than what was known before; see further details after the proof of Theorem B.2.1.

Proof. Assume towards a contradiction that $\mathcal{E}^{\mathcal{NP}} \subseteq \mathcal{C}$. Fixing an arbitrary set $L \in NTIME(2^n)$, we will construct a non-deterministic machine that decides L in time $2^n/n^{\omega(1)}$. This will contradict the non-deterministic time hierarchy.

Towards describing the machine, first fix a PCP system for L with the following properties: The verifier in this system, denoted V, is given an input $x \in \{0,1\}^n$, randomness $r \in \{0,1\}^{n'}$, and a proof-oracle $\pi_x \in \{0,1\}^{2^{n'}}$ for x, where $n' = n + O(\log(n))$; the verifier issues $\text{poly}(n)$ queries to π_x, verifies the answers to the queries in time $\text{poly}(n)$, and has perfect completeness and soundness $1/2$. A PCP construction with such properties appears in [BGH+05], which builds on **[bss05]**. Observe that if we denote $N = 2^n$ (such that $L \in NTIME(N)$), then the proof length is $\tilde{O}(N)$, and the number of queries and running time of V are polylogarithmic in N.

The strategy of the machine is as follows. Assume, for a moment, that for any $x \in L$ *there exists a proof π_x for x that has a concise representation by a circuit from $\mathcal{C}$*. Specifically, assume that there exists a circuit $P_x \in \mathcal{C}$ such that for $i \in [n']$ it holds that $P_x(i)$ is the i^{th} bit of some (fixed) proof π_x for x. Since circuits in $\mathcal{C}$ are of size significantly less than 2^n, the representation length of P_x is much smaller than the length of proofs in the PCP system (i.e., $2^{n'}$). Then, given input $x \in \{0,1\}^n$, the machine can non-deterministically guess the circuit P_x, and construct a circuit $C_{x,P_x} \in \widehat{\mathcal{C}}$ such that $C_{x,P_x}(r) = V^{P_x}(x,r)$; that is, the circuit C_{x,P_x} implements the verification procedure of V, while resolving oracle queries using copies of the circuit P_x. Observe that if $x \in L$, then for some guess of P_x it holds that C_{x,P_x} has acceptance probability one, and if $x \notin L$, then for any guess of P_x it holds that C_{x,P_x} has acceptance probability at most $1/2$. The machine can distinguish between these two cases in time $2^{n'}/(n')^{\omega(1)} = 2^n/n^{\omega(1)}$ using the algorithm D from the theorem's hypothesis.

The key observation is that *if $\mathcal{E}^{\mathcal{NP}} \subseteq \mathcal{C}$, then for any $x \in L$ there indeed exists a concise representation of a proof π_x for x*. To see this, consider the function Π that gets as input x and a location $i \in [2^{n'}]$, and outputs the i^{th} bit of the lexicographically-first correct proof for x, if such a proof exists. Note that the function Π is in $\mathcal{E}^{\mathcal{NP}}$, since the lexicographically-first proof for x can be constructed in its entirety by an $\mathcal{E}^{\mathcal{NP}}$ machine, bit-by-bit.[8] Therefore, relying on the hypothesis that $\mathcal{E}^{\mathcal{NP}} \subseteq \mathcal{C}$, there exists a circuit $P \in \mathcal{C}$ such that $P(x,i) = \Pi(x,i)$. By hard-wiring x into P, there exists a circuit $P_x \in \mathcal{C}$ such that $P_x(i) = \Pi(x,i)$.

In the theorem's hypothesis, we assumed that the algorithm D works when given a circuit from a class $\widehat{\mathcal{C}}$ of circuits that we informally referred to as "slightly larger" than circuits in $\mathcal{C}$. What we will actually rely on is that the algorithm D works when given the circuit C_{x,P_x}. To conclude, let us bound the running-time of the non-deterministic machine. Since the verifier V runs in time $\text{poly}(n)$, and P_x has size at most $2^{(1-\Omega(1))\cdot n}$, [9]

[8] Specifically, the $\mathcal{E}^{\mathcal{NP}}$ algorithm works in $2^{n'}$ iterations: In the i^{th} iteration, the algorithm extends the prefix of length $i-1$ of the lexicographically-first proof for x by a single bit, using the $\mathcal{NP}$ oracle to decide whether or not the prefix can be extended by zero. (Given an i-bit prefix $\tau \in \{0,1\}^i$, the non-deterministic machine that implements the oracle function "guesses" a suffix $\tau' \in \{0,1\}^{n'-i}$, and simulates the PCP verifier V on all $2^{n'}$ possible coin tosses to check whether $\tau\tau'$ is a correct proof for x.)

[9] The size bound is because the circuit P is a $\mathcal{C}$-circuit over $n + n' < 3n$ input bits, and therefore P and

the circuit C_{x,P_x} can be constructed in non-deterministic time $2^{(1-\Omega(1))\cdot n}$. The running time is thus dominated by the running time of the algorithm D, which is at most $2^{n'}/(n')^{\omega(1)} = 2^n/n^{\omega(1)}$. ■

Digest. The proof is based on the following *unconditional lower bound*: Given a representation P_x of a proof in the aforementioned PCP system (for an arbitrary set in $NTIME(2^n)$), it is impossible to decide in time $2^n/n^{\omega(1)}$ whether the acceptance probability of $C_{x,P_x}(r) = V^{P_x}(x,r)$ is one or is at most half. Therefore (disregarding for a moment the difference between $\mathcal{C}$ and $\widehat{\mathcal{C}}$), a circuit class $\mathcal{C}$ cannot be both so strong such that we are able to get $P_x \in \mathcal{C}$ and construct $C_{x,P_x} \in \mathcal{C}$, while still susceptible to derandomization in non-trivial time.

Let me now be more specific with respect to the definition of the circuit class $\widehat{\mathcal{C}}$. The exact requirements from $\widehat{\mathcal{C}}$ depend on the specific PCP system that we use in the proof; that is, fixing a suitable PCP system with a verifier V, the class $\widehat{\mathcal{C}}$ can be defined as the class of circuits that, when given input r, implement the verification procedure of V on some input x with randomness r, while resolving each query to the proof oracle by a sub-circuit from $\mathcal{C}$. Subsequent research indeed focused on constructing PCP systems with extremely efficient verifiers; in particular, Ben-Sasson and Viola [BSV14] constructed a PCP in which the query-locations are *projections* of bits of the randomness, and the verification procedure (of the answers) is just a 3CNF. This allows, for example, to apply the argument in Theorem B.2.1 to classes of circuits of constant depth (since the depth overhead when constructing $\widehat{\mathcal{C}}$ from $\mathcal{C}$ is constant).

The only property of $\mathcal{E}^{\mathcal{NP}}$ that is used in the proof is that the function Π is in $\mathcal{E}^{\mathcal{NP}}$ (which allowed us to deduce that for any $x \in L$ there exists a concise representation of a proof π_x for x by a $\mathcal{C}$-circuit). Thus, using the same proof, we can separate $\mathcal{C}$ from any class that contains Π. As mentioned above, in the original proof of Williams, instead of separating the class $\mathcal{C}$ from $\mathcal{E}^{\mathcal{NP}}$, the class $\mathcal{C}$ is separated from $\mathcal{NEXP}$. Indeed, for many specific circuit classes $\mathcal{C}$, if $\mathcal{NEXP} \subseteq \mathcal{C}$, then for any $x \in L$ there exists a concise representation of a proof π_x for x by a $\mathcal{C}$-circuit (see [Wil13; Wil11; SW13] for further details). However, this fact is highly non-trivial, and follows from the work of Impagliazzo, Kabanets, and Wigderson [IKW02].

Finally, one can immediately strengthen Theorem B.2.1 in two ways. First, we don't have to unconditionally assume that the algorithm D exists; in fact, it suffices to only assume that D exists *under the hypothesis that* $\mathcal{E}^{\mathcal{NP}} \subseteq \mathcal{C}$. Secondly, since we are using the algorithm D as a sub-routine of a non-deterministic algorithm, we can allow D to be non-deterministic itself. However, the non-determinism of D should be such that there exists a proof that leads D to accept circuits with acceptance probability one, whereas circuits with low acceptance probability are rejected by D (regardless of the proof); it's not a-priori clear how non-determinism might be useful for this task.

P_x are of size at most $2^{(3n)/4}$.

B.3 The Bazzi-Razborov-Braverman Theorems: Polylogarithmic Independence Fools $\mathcal{AC}^0$

A distribution $\mathbf{w}$ over $\{0,1\}^n$ is called t-wise independent if for every set $S \subseteq [n]$ of size $|S| = t$, the marginal distribution $\mathbf{w}_S$ is uniform. Linial and Nisan [LN90] conjectured that any $\mathcal{AC}^0$ circuit $C : \{0,1\}^n \to \{0,1\}$ of depth d cannot distinguish, up to error ϵ, between the uniform distribution $\mathbf{u}_n$ over $\{0,1\}^n$ and any (arbitrary) distribution $\mathbf{w}$ over $\{0,1\}^n$ that is t-wise independent, where $t = \log^{O(d)}(n/\epsilon)$; that is, $|\Pr[C(\mathbf{u}_n) = 1] - \Pr[C(\mathbf{w}) = 1]| \leq \epsilon$. [10] After decades of the conjecture being an open problem, it was finally proved in a sequence of works by Bazzi, Razborov, and Braverman [Baz09; Raz09; Bra10]. In this text I survey their proofs. (I will not survey a subsequent improvement of t to $\log^{O(d)}(m) \cdot \log(1/\epsilon)$ by [HS16].)

Bird's eye. In high-level, $\mathcal{AC}^0$ circuits are "fooled" by polylog-wise independent distributions because $\mathcal{AC}^0$ circuits can be approximated by real polynomials of polylogarithmic degree, and such polynomials are "fooled" by polylog-wise independent distributions (since each monomial depends only on a polylogarithmic number of variables, and using linearity of expectation). However, this is only a very rough intuition. In particular, various approximations of $\mathcal{AC}^0$ circuits by polynomials of polylogarithmic degree have been known since the 80's, but the proof of the conjecture hinges on a *specific type of approximation* that we will need to carefully construct.

Starting point. The starting point for the proofs is the well-known fact, first proved in [LMN93], that any depth-d circuit C can be approximated *in ℓ_2-distance* by a real polynomial $p : \{0,1\}^n \to \mathbb{R}$ of degree $t = O(\log^d(n/\epsilon))$; that is, $\|C - p\|_2^2 = \mathbb{E}_{x \sim \mathbf{u}_n}\left[(C(x) - p(x))^2\right] \leq \epsilon$. [11] Let us try to directly use the existence of p to prove the conjecture, and see where we get stuck. In the following expression, $\mathbf{w}$ is a distribution that is t-wise independent, and we abbreviate $\mathbb{E}[C] = \mathbb{E}[C(\mathbf{u}_n)]$ and $\mathbb{E}_{\mathbf{w}}[C] = \mathbb{E}[C(\mathbf{w})]$:

$$\left|\mathbb{E}[C] - \mathbb{E}_{\mathbf{w}}[C]\right| \leq \left|\mathbb{E}[C] - \mathbb{E}[p]\right| + \left|\mathbb{E}[p] - \mathbb{E}_{\mathbf{w}}[p]\right| + \left|\mathbb{E}_{\mathbf{w}}[p] - \mathbb{E}_{\mathbf{w}}[C]\right| . \tag{B.3.1}$$

The second term in Eq. (B.3.1) is zero (since $\mathbf{w}$ "fools" p), and the first term is upper-bounded by $\mathbb{E}[|C - p|] = \|C - p\|_1 \leq \|C - p\|_2$ (using Cauchy-Schwarz), which is smaller than ϵ if we take p to be an ℓ_2-approximation of C with error ϵ^2. However, it is not clear how to upper-bound the last term in Eq. (B.3.1). This is the case since p only approximates C in ℓ_2-distance, which is measured with respect to the *uniform distribution* on the inputs (i.e., the measure distance is the expected value of $(C(x) - p(x))^2$ where x is sampled uniformly). In contrast to the uniform distribution, the distribution $\mathbf{w}$ might have very small support, and in particular $\mathbf{w}$ might put a lot of weight on inputs where $p(x)$ significantly differs from $C(x)$.

[10] The original conjecture was for $t = O(\log^{d-1}(n))$, but this stronger form was proven incorrect [LV96].

[11] The celebrated result of [LMN93] is based on a Fourier-analytical reduction to Håstad's switching lemma [Hås87]; their analysis was later improved to yield a tight result, see [Bop97; Tal17].

The first observation in the proof, which is due to Bazzi, is the following. Assume that instead of one approximating polynomial p we have *two polynomials* p_ℓ and p_u that both ϵ-approximate C in ℓ_2-distance, and for every $x \in \{0,1\}^n$ we have that $p_\ell(x) \leq C(x) \leq p_u(x)$. That is, p_ℓ and p_u and "lower-sandwiching" and "upper-sandwiching" for C, respectively. Then, we deduce that

$$\mathbb{E}[C] - \mathbb{E}_{\mathbf{w}}[C] \leq \mathbb{E}[C] - \mathbb{E}[p_\ell] + \mathbb{E}[p_\ell] - \mathbb{E}_{\mathbf{w}}[p_\ell] + \mathbb{E}_{\mathbf{w}}[p_\ell] - \mathbb{E}_{\mathbf{w}}[C] \leq \epsilon \,, \qquad \text{(B.3.2)}$$

where the inequality is since the first term and the second term are bounded by ϵ as above, whereas the third term is non-positive since p_ℓ is "lower-sandwiching". Similary, using p_u instead of p_ℓ, we can deduce that $\mathbb{E}_{\mathbf{w}}[C] - \mathbb{E}[C] \leq \epsilon$. Put together, we can then deduce that $\left|\mathbb{E}[C] - \mathbb{E}_{\mathbf{w}}[C]\right| \leq \epsilon$, which is what we wanted.

In fact, we don't even have to assume that the "sandwiching" polynomials p_ℓ and p_u approximate C in ℓ_2-distance, and it suffices to assume that they approximate C in ℓ_1-distance; that is, we just need that $\|C - p_\ell\|_1 = \mathbb{E}_x[|C(x) - p_\ell(x)|] \leq \epsilon$, and ditto for p_u. (Recall that for any f we have that $\|f\|_1 \leq \|f\|_2$, by Cauchy-Schwarz, and so an approximation in ℓ_2-distance is stronger than an approximation in ℓ_1-distance.)

B.3.1 Bazzi's Lemma: ℓ_2-approximations with one-sided error

The initial lemma that the proof relies on is the following: If there exists a low-degree polynomial p_0 of degree d that **ϵ-approximates C in ℓ_2-distance and has one-sided error** (i.e., p_0 vanishes on $C^{-1}(0)$), then there exists a polynomial p_ℓ of degree $2d$ that ϵ-approximates C in ℓ_1-distance and is "lower-sandwiching" for C. As explained above, it follows that $\mathbb{E}[C] - \mathbb{E}_{\mathbf{w}}[C] \leq \epsilon$ for any $(2d)$-wise independent distribution $\mathbf{w}$.

Moreover, since we are dealing with $\mathcal{AC}^0$, then a "lower-sandwiching" p_ℓ suffices, and we don't need an "upper-sandwiching" approximation p_u. This is because for any class $\mathcal{F}$ of functions that is closed to negations, if we know that $\mathbb{E}[f] - \mathbb{E}_{\mathbf{w}}[f] \leq \epsilon$ for every $f \in \mathcal{F}$, then we also know that $\mathbb{E}_{\mathbf{w}}[f] - \mathbb{E}[f] = \mathbb{E}[1-f] - \mathbb{E}_{\mathbf{w}}[1-f] \leq \epsilon$ (where we used the fact that $1 - f = \neg f \in \mathcal{F}$).

Armed with this lemma, which will be proved in a moment, the gap between what we know and what we need to prove is the following: We know that any circuit C of depth d and size m can be ϵ-approximated in ℓ_2-distance by a polynomial p of degree $O(\log^d(m/\epsilon))$, and we want to have such an approximation by a polynomial p_0 that *also vanishes on* $C^{-1}(0)$. Constructing p_0 will indeed be the focus of the proof, and we will be able to construct p_0 with degree $\log^{O(d)}(m/\epsilon)$.)

Let us now formally state Bazzi's lemma, which constructs a "lower-sandwiching" ℓ_1-approximation p_ℓ given a one-sided error ℓ_2-approximation p_0. Afterwards, we state a useful corollary, which refers to classes $\mathcal{F}$ that are closed to negations.

Lemma B.3.1 (Bazzi's lemma; the core argument)**.** *Let $f : \{0,1\}^n \to \{0,1\}$. Assume that there exists $p_0 : \{0,1\}^n \to \mathbb{R}$ such that $\|f - p_0\|_2^2 \leq \epsilon$ and p_0 vanishes on $f^{-1}(0)$. Then, the polynomial $p_\ell = 1 - (1 - p_0)^2$ is a "lower-sandwiching" ϵ-approximation of f in ℓ_1-distance*

(i.e., $\mathbb{E}_x[|f(x) - p_\ell(x)|] \leq \epsilon$, and $p_\ell(x) \leq f(x)$ for every $x \in \{0,1\}^n$). Consequently, for any distribution $\mathbf{w}$ *over* $\{0,1\}^n$ *that is* ϵ*-pseudorandom for* p_ℓ *it holds that* $\mathbb{E}[f] - \mathbb{E}_{\mathbf{w}}[f] \leq 2\epsilon$.

Proof. We first claim that for every $x \in \{0,1\}^n$ it holds that $f(x) - p_\ell(x) = (f(x) - p_0(x))^2$. This is the case since for every $x \in f^{-1}(0)$ we have that $p_0(x) = p_\ell(x) = 0$, whereas for every $x \in f^{-1}(1)$ we have that $f(x) - p_\ell(x) = f(x) - 1 + (1 - p_0(x))^2 = (1 - p_0(x))^2$. It follows that $p_\ell(x) \leq f(x)$ for all $x \in \{0,1\}^n$ (since $f - p_\ell$ is a non-negative function), and that $\|f - p_\ell\|_1 = \mathbb{E}_x[|f(x) - p_\ell(x)|] = \mathbb{E}_x[(f(x) - p_0(x))^2] = \|f - p_0\|_2^2 \leq \epsilon$. The "consequently" part follows by bounding the expression in Eq. (B.3.2) as in the initial overview. ■

Corollary B.3.2 (a useful corollary of Lemma B.3.1)**.** *Let* $\mathcal{F} \subseteq \{\{0,1\}^n \to \{0,1\}\}$ *be a class of functions that is closed to negations. Assume that for every* $f \in \mathcal{F}$ *there exists* $p_0 : \{0,1\}^n \to \mathbb{R}$ *of degree* d *such that* $\|f - p_0\|_2^2 \leq \epsilon$ *and* p_0 *vanishes on* $f^{-1}(0)$*. Then, any distribution* $\mathbf{w}$ *that is* $(2d)$*-wise independent is* ϵ*-pseudorandom for* $\mathcal{F}$.

Let me mention in advance that the proof will not rely on Corollary B.3.2 as a "black-box", but will need variations of it.[12] However, Corollary B.3.2 seems like a clean statement that still captures the main idea in the underlying argument.

B.3.2 Low-degree approximations in ℓ_2 distance with one-sided error for small-depth circuits

Our goal now is to approximate every $\mathcal{AC}^0$ circuit C in ℓ_2 distance with one-sided error by a low-degree polynomial p_0. I'll first present Bazzi's proof for the special case of depth-two circuits (as simplified by Razborov and by Wigderson), and then present Braverman's proof for $\mathcal{AC}^0$ circuits of any constant depth. (The special case of depth-two circuits has both a simpler proof, and better parameters: The distribution for depth-two circuits is $O(\log^2(m/\epsilon))$-wise independent, whereas the distribution for depth-d circuits is $\log^{O(d)}(m/\epsilon)$-wise independent.)

B.3.2.1 Depth-two circuits: Wigderson's simplification of Razborov's simplification of Bazzi's construction

We will prove that all DNFs are "ϵ-fooled" by any distribution $\mathbf{w}$ that is $O(\log^2(m/\epsilon))$-wise independent. It follows that all CNFs are also "ϵ-fooled" by $\mathbf{w}$. Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a depth-two circuit, and assume that $f = A_1 \vee ... \vee A_m$, where each A_i is a conjunction of $O(\log(m/\epsilon))$ literals.[13]

[12] The main source of trouble for Bazzi's argument is that the class $\mathcal{F}$ of DNFs is *not* closed to negations. In Braverman's argument, we do not construct p_0 directly for a circuit C, but rather for auxiliary circuits that approximate C.

[13] More formally, we carry out the analysis with respect to the DNF $\widetilde{f}$ that is obtained by trimming each of the m clause of f to consist of at most $w = O(\log(m/\epsilon))$ literals. We can analyze $\widetilde{f}$ instead of f since $\left|\mathbb{E}[f] - \mathbb{E}[\widetilde{f}]\right| \leq \epsilon$ and $\left|\mathbb{E}_{\mathbf{w}}[f] - \mathbb{E}_{\mathbf{w}}[\widetilde{f}]\right| \leq \epsilon$ (the inequalities are since each clause is satisfied only if its first w literals are true, which happens with probability $2^{-w} \leq \epsilon/m$ both under $\mathbf{u}_n$ and under $\mathbf{w}$).

Our first step is to construct a polynomial p_0 of degree $O(\log^2(m/\epsilon))$ that ϵ-approximates f in ℓ_2-distance while vanishing on all points in $f^{-1}(0)$. To do so, for any $i \in [m]$, denote by $f^{(i)}$ the function $f^{(i)}(x) = A_1(x) \vee ... \vee A_i(x)$, and also denote by $f^{(0)}$ the constant zero function. Then, a useful observation is that for every $x \in \{0,1\}^n$ it holds that $f(x) = \vee_{i\in[m]} \left(A_i(x) \wedge \neg f^{(i-1)}(x)\right) = \sum_{i\in[m]} A_i(x) \cdot \left(1 - f^{(i-1)}(x)\right)$, where the arithmetic is over the reals. This is because if $f(x) = 0$, then for every $i \in [m]$ it holds that $A_i(x) = 0$; whereas if $f(x) = 1$ then there exists a unique $i \in [m]$ such that $A_i(x) = 1$ and that $f^{(i-1)}(x) = 0$ (i.e., $A_j(x) = 0$ for all $j < i$).

The approximating polynomial p_0 of f is $p_0(x) = \sum_{i\in[m]} a_i(x) \cdot \left(1 - p^{(i-1)}(x)\right)$, where a_i is the multiplication of the $O(\log(m/\epsilon))$ literals in A_i, and $p^{(i-1)}(x)$ is the standard (known) ℓ_2 approximation of the sub-DNF $f^{(i-1)}(x)$, with error $\delta = \epsilon/m^2$ and degree $O(\log(m/\delta))$ (see [Bop97]). So overall p_0 has degree $O(\log^2(m/\epsilon))$.

To see that p_0 has one-sided error in approximating f, note that if $f(x) = 0$ then $A_i(x) = a_i(x) = 0$ for all $i \in [m]$, which implies that $p_0(x) = 0$. To see that p_0 is an approximation in ℓ_2-distance for f, note that

$$\begin{aligned}
\|f - p_0\|_2^2 &= \mathbb{E}_x\left[(f(x) - p_0(x))^2\right] \\
&= \mathbb{E}_x\left[\left(\sum_{i\in[m]} a_i(x) \cdot \left(p^{(i-1)}(x) - f^{(i-1)}(x)\right)\right)^2\right] && (A_i(x) = a_i(x)) \\
&\le \mathbb{E}_x\left[m \cdot \sum_{i\in[m]} a_i^2(x) \cdot \left(p^{(i-1)}(x) - f^{(i-1)}(x)\right)^2\right] && \text{(Cauchy-Schwartz)} \\
&\le m \cdot \sum_{i\in[m]} \cdot \left\|p^{(i-1)} - f^{(i-1)}\right\|_2^2, && (a_i(x) \in \{0,1\})
\end{aligned}$$

which is upper-bounded by $\delta \cdot m^2 = \epsilon$.

Now, due to our construction of p_0, we can rely on Lemma B.3.1 to deduce that $\mathbb{E}[f] - \mathbb{E}_{\mathbf{w}}[f] \le \epsilon$ for any $O(\log^2(m/\epsilon))$-wise independent distribution $\mathbf{w}$. However, we still need to prove that $\mathbb{E}_{\mathbf{w}}[f] - \mathbb{E}[f] \le \epsilon$, and we cannot apply Corollary B.3.2, since the class of DNFs is not closed to negations. To overcome this problem, Bazzi used the existence of p_0 to explicitly construct an "upper-sandwiching" polynomial p_u for f such that $\|f - p_u\|_2^2 \le \epsilon$. Specifically, let $p_u(x) = 1 - (1 - \sum_{i\in[m]} A_i(x)) \cdot (1 - p_0(x))^2$. One can verify that p_u vanishes on $f^{-1}(0)$ and that for every $f^{-1}(1)$ it holds that $p_u(x) \ge 1$; hence, p_u is indeed "upper-sandwiching" for f. Additionally, we have that $\mathbb{E}_x[p_u(x) - f(x)] \le \mathbb{E}_x[p_u(x) - p_\ell(x)] = \mathbb{E}_x\left[\sum_{i\in[m]} A_i(x) \cdot (1 - p_0)^2\right] = \mathbb{E}_x\left[\sum_{i\in[m]} A_i(x) \cdot (f - p_0)^2\right] \le m \cdot \|f - p_0\|_2^2$, where the penultimate equality can be verified by separately considering $x \in f^{-1}(1)$ and $x^{-1}(0)$. Thus, if we take $\delta = \epsilon/m^3$, we have $\mathbb{E}_x[p_u(x) - f(x)] \le \delta \cdot m^3 \le \epsilon$, and it follows that $\mathbb{E}_{\mathbf{w}}[f] - \mathbb{E}[f] \le \epsilon$.

A follow-up: The PRG of De *et al.* [DET+10]. Bazzi's proof framework was later on used by De *et al.* [DET+10] (see also improvement by Tal [Tal17]) to construct what are currently the best-known pseudorandom generators for depth-two circuits.

Specifically, for every DNF f, De *et al.*constructed a polynomial p_0 that ϵ-approximates f in ℓ_2-distance and vanishes on $f^{-1}(0)$ such that p_0 has *small spectral norm*, rather than low degree (i.e., p_0 has small ℓ_1-norm in the Fourier basis). Then, they used Bazzi's constructions of p_ℓ and of p_u, while arguing that the spectral norm does not significantly increase, and deduced that any distribution that is pseudorandom for polynomials with small spectral norm is also pseudorandom for depth-two circuits. (In particular, any small-biased set is pseudorandom for depth-two circuits.) In fact, their construction of p_0 is very similar to the construction above, the main difference being that the $p^{(i)}$'s are not the standard ℓ_2-approximations of the $f^{(i)}$'s, but rather different (known) ℓ_2-approximations by Mansour [Man95] that have small spectral norm.

B.3.2.2 Constant depth circuits: Braverman's idea

I'll present the proof in a slightly different way than in the original paper, mainly by introducing a preliminary conceptual step.

A preliminary step: Randomly computing a function by a distribution over simpler functions We say that a distribution $\mathbf{g}$ over functions $g : \{0,1\}^n \to \{0,1\}$ randomly computes a function $f : \{0,1\}^n \to \{0,1\}$ with error ϵ if for every $x \in \{0,1\}^n$ it holds that $\Pr_{g\sim\mathbf{g}}[g(x) = f(x)] \geq 1-\epsilon$. The following claim asserts that if $\mathbf{g}$ randomly computes f, then any distribution $\mathbf{w}$ that is pseudorandom for all the functions g in the support of $\mathbf{g}$ is also pseudorandom for f. In fact, the conclusion also holds if $\mathbf{w}$ is pseudorandom for *almost all* the functions g in the support of $\mathbf{g}$. This follows the ideas outlined in Section 3.2.

Lemma B.3.3 (randomized tests; a basic form)**.** *Let $f : \{0,1\}^n \to \{0,1\}$, and let $\mathbf{g}$ be a distribution over functions $g : \{0,1\}^n \to \{0,1\}$ that randomly computes f with error ϵ. Let $\mathbf{w}$ be a distribution such that the probability over $g \sim \mathbf{g}$ that $\mathbf{w}$ is ϵ-pseudorandom for g is at least $1-\epsilon$. Then, $\mathbf{w}$ is 4ϵ-pseudorandom for f. Moreover, if we only assume that for $1-\epsilon$ of the g's in the support of $\mathbf{g}$ it holds that $\mathbb{E}[g] - \mathbb{E}_{\mathbf{w}}[g] \leq \epsilon$, then we can deduce that $\mathbb{E}[f] - \mathbb{E}_{\mathbf{w}}[f] \leq 4\epsilon$.*

Proof. Note that $\left|\mathbb{E}[f(\mathbf{u}_n)] - \mathbb{E}[f(\mathbf{w})]\right|$ is upper-bounded by

$$\left|\mathbb{E}[f(\mathbf{u}_n)] - \mathbb{E}[\mathbf{g}(\mathbf{u}_n)]\right| + \left|\mathbb{E}[\mathbf{g}(\mathbf{u}_n)] - \mathbb{E}[\mathbf{g}(\mathbf{w})]\right| + \left|\mathbb{E}[\mathbf{g}(\mathbf{w})] - \mathbb{E}[f(\mathbf{w})]\right| .$$

The first term $\left|\mathbb{E}[f(\mathbf{u}_n)] - \mathbb{E}[\mathbf{g}(\mathbf{u}_n)]\right|$ is at most ϵ, since for any fixed $x \in \{0,1\}^n$ it holds that $\Pr[f(x) \neq \mathbf{g}(x)] \leq \epsilon$. The same reasoning implies that the third term is also upper-bounded by ϵ. To see that the second term is also upper-bounded by ϵ,

consider a choice of $g \sim \mathbf{g}$, and denote by $\mathcal{E}$ the event that $\mathbf{w}$ is ϵ-pseudorandom for g; then, we have that

$$\begin{aligned}\left|\mathbb{E}[\mathbf{g}(\mathbf{u}_n)] - \mathbb{E}[\mathbf{g}(\mathbf{w})]\right| &\leq \mathbb{E}_{g\sim\mathbf{g}}\left[\left|\mathbb{E}[g(\mathbf{u}_n)] - \mathbb{E}[g(\mathbf{w})]\right|\right] \\ &\leq \Pr_{g\sim\mathbf{g}}[\neg\mathcal{E}] + \max_{g\sim\mathbf{g}|\mathcal{E}}\left\{\left|\mathbb{E}[g(\mathbf{u}_n)] - \mathbb{E}[g(\mathbf{w})]\right|\right\} ,\end{aligned}$$

which is upper-bounded by 2ϵ. The "moreover" part uses essentially the same proof, the only difference being that all the expressions are without absolute values. ■

Lemma B.3.3 is useful when we want to construct a pseudorandom generator for a function f. The lemma asserts that if f can be computed by a distribution $\mathbf{g}$ over "simpler" functions, then it suffices to construct a pseudorandom generator for the "simpler" functions (since such a generator "fools" f). The point is that the distribution $\mathbf{g}$ might have very high entropy (i.e., we can use a lot of randomness to compute f), but this distribution is still only part of the *analysis*, and thus we don't "pay" for this randomness when constructing the generator itself.

The proof outline Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of depth d and size m. Our proof strategy will be to randomly compute C by a distribution $\mathbf{C}$ such that almost all of the functions $C_p : \{0,1\}^n \to \{0,1\}$ in the support of $\mathbf{C}$ have approximations in ℓ_2-distance with one-sided error by a low-degree polynomial. We then rely on Lemma B.3.1 to claim that a distribution with limited independence is pseudorandom for almost all the functions C_p in the support of $\mathbf{C}$, and rely on Lemma B.3.3 to deduce that this distribution is also pseudorandom for C. That is, the proof overview is:

1. **New claim:** The circuit C can be randomly computed by a distribution $\mathbf{C}$ such that $1-\epsilon$ of the functions in the support of $\mathbf{C}$ can be ϵ-approximated in ℓ_2 distance with one-sided error by a polynomial of degree $t = \log^{O(d)}(m/\epsilon)$.

2. **Lemma B.3.1:** For $1-\epsilon$ of the functions C_p in the support of $\mathbf{C}$ it holds that $\mathbb{E}[C_p] - \mathbb{E}_{\mathbf{w}}[C_p] \leq \epsilon$, where $\mathbf{w}$ is a t-wise independent distribution.

3. **Lemma B.3.3:** It holds that $\mathbb{E}[C] - \mathbb{E}_{\mathbf{w}}[C] = O(\epsilon)$.

Finally, since the class of depth-d circuits is closed to negations, the argument above also holds for $\neg C = 1 - C$, and thus $\mathbb{E}_{\mathbf{w}}[C] - \mathbb{E}[C] = \mathbb{E}[\neg C] - \mathbb{E}_{\mathbf{w}}[C] = O(\epsilon)$. [14]

[14] Braverman's original proof did not rely on a lemma similar to Lemma B.3.3. In the original proof, Braverman relied on the existence of $\mathbf{C}$ to deduce that for every distribution μ over the inputs there exists a single "good" $C_p \sim \mathbf{C}$ that agrees with C on $1-\epsilon$ of the inputs according to μ (this claim is then used with $\mu = \mathbf{u}_n$ and $\mu = \mathbf{w}$). When using Lemma B.3.3 we avoid this argument.

Randomly computing $\mathcal{AC}^0$ circuits by functions that can be approximated in ℓ_2 distance with one-sided error The main thing to prove is the "new claim" from the outline above: Every depth-d circuit C of size m can be randomly computed by a distribution $\mathbf{C}$ such that $1-\epsilon$ of the functions C_p in the support of $\mathbf{C}$ can be ϵ-approximated in ℓ_2 distance with one-sided error by a polynomial of degree $t = \log^{O(d)}(m/\epsilon)$.

The starting point of the proof is the following claim, which asserts the existence of a distribution $\mathbf{p}$ of low-degree polynomials that randomly compute C, as well as an accompanying small-depth circuit E_p for each choice of $p \sim \mathbf{p}$. The circuit E_p acts as an "error-detector" for p; that is, whenever $p(x) \neq C(x)$ we have that $E_p(x) = 1$.

Lemma B.3.4 (Razborov-Smolensky polynomials with an error-detector)**.** *Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of depth d and size m, and let $\epsilon > 0$. Then, there exists a distribution $\mathbf{p}$ over real polynomials $p : \{0,1\}^n \to \mathbb{R}$, and a mapping $p \mapsto E_p$ where $E_p : \{0,1\}^n \to \{0,1\}$ is computable by a circuit of depth $d + O(1)$, that has the following properties:*

1. *All polynomials p in the support of $\mathbf{p}$ have degree $r = O(\log(m/\epsilon))^{d+1}$.*
2. *For every $x \in \{0,1\}^n$ it holds that $\max_{p\sim\mathbf{p}}\{|p(x)|\} = 2^{\log^{O(d)}(m/\epsilon)}$.*
3. *For every $x \in \{0,1\}^n$ and $p \sim \mathbf{p}$ such that $p(x) \neq C(x)$ it holds that $E_p(x) = 1$.*
4. *For every $x \in \{0,1\}^n$ it holds that $\Pr_{p\sim\mathbf{p}}[E_p(x) = 1] \leq \epsilon^2$.*

Lemma B.3.4 is not stated as-is in Braverman's paper, but Braverman's proofs readily yield it (see [Bra10, Thm. 8, Prop. 9]). The proof relies on a modification of classical constructions of probabilistic polynomials for $\mathcal{AC}^0$ and for $\mathcal{AC}^0[\oplus]$ circuits (i.e., the constructions of [Raz87; Smo87; BRS91; Tar93]; the constructions are surveyed in many good sources, e.g. [AB09, Sec. 14.2]). The main new observation of Braverman is the function E_p can be computed by a small-depth circuit; one should think of E_p as an "error-detector" that gets input $x \in \{0,1\}^n$ and detects the "error" $p(x) \neq C(x)$ such that E_p never misses but has (few) false alarms.

Indeed, by an averaging argument, one can see that almost all polynomials p in the support of $\mathbf{p}$ agree with C on almost all inputs. However, any such polynomial p might not be a good approximator in ℓ_2-distance for C, since p might take very large values (i.e., up to $\pm 2^{\log^{O(d)}(m/\epsilon)}$) on inputs on which it disagrees with C.

The plan and intuition for the rest of the proof are as follows (the overview that comes next roughly corresponds to the one in [Bra10, Sec. 1.3]). We first "approximate" C by considering the distribution $\mathbf{C} = C \vee E_{\mathbf{p}}$; that is, the distribution obtained by sampling $p \sim \mathbf{p}$ and outputting $C_p = C \vee E_p$. Due to Items (3) and (4) in Lemma B.3.4, this distribution computes C with error ϵ^2. Using an averaging argument, we will deduce that for $1-\epsilon$ of the functions $C_p = C \vee E_p$ in the support of $\mathbf{C}$ it holds that $\Pr_x[E_p(x) = 1] \leq \epsilon$. Finally, we approximate any such $C_p = C \vee E_p$ by the polynomial $p \cdot (1 - \tilde{E_p})$, where $\tilde{E_p}$ is the standard ℓ_2-approximation of the circuit E_p

(i.e., without one-sided error), with a small error δ to be determined later.[15] The intuitive idea behind this approximation is that in the (rare) event that $p(x) \neq C(x)$, the multiplication of $p(x)$ by $1 - \tilde{E_p}(x) \approx 0$ will "suppress" the potentially-large values of p.

Let us see that $p \cdot (1 - \tilde{E_p})$ is indeed an ϵ-approximation in ℓ_2-distance for C_p and that it vanishes on $C_p^{-1}(0)$. To do so, first note that when $C_p(x) = 0$ we have that $E_p(x) = C(x) = 0 \Rightarrow p(x) = 0$, and hence $p \cdot (1 - \tilde{E_p})$ vanishes on $C_p^{-1}(0)$. Now, intuitively, we replaced C with p, and replaced the disjunction with E_p with a multiplication by $(1 - \tilde{E_p})$. On inputs $x \in E_p^{-1}(0)$, this replacement shouldn't matter much, since $C(x) = p(x)$ and $1 - \tilde{E_p}(x) \approx 1$ (and the latter approximation is in ℓ_2-distance). On inputs $x \in E_p^{-1}(1)$, the replacement of C by p might create an error as large as $2^{\log^{O(d)}(m/\epsilon)}$ (due to Item (2) in Lemma B.3.4). However, note that the contribution of each such input to the error is the (square of the) difference between $C_p(x) = 1$ and $p \cdot (1 - \tilde{E_p})$, where $1 - \tilde{E_p}(x) \approx 0$. In particular, if we take $\tilde{E_p}$ to be an approximation of E_p with error $\delta = 2^{-\log^{O(d)}(m/\epsilon)}$, which requires degree about $\log(1/\delta) = \log^{O(d)}(m/\epsilon)$, the contribution of each such input is approximately $C_p^2(x) = 1$, and thus their overall contribution is approximately $\Pr_x[E_p(x) = 1] \leq \epsilon$. Details follow.

To sample $C_p \sim \mathbf{C}$, we sample p from the distribution $\mathbf{p}$ in Lemma B.3.4, and let $C_p = C \vee E_p$. First note that $\mathbf{C}$ indeed randomly computes C with error ϵ (i.e., for every $x \in \{0,1\}^n$ it holds that $\Pr_{C_p \sim \mathbf{C}}[C_p(x) = C(x)] \geq 1 - \epsilon$); this is the case since $C_p(x) \neq C(x)$ only when $E_p(x) = 1$, which happens with probability at most $\epsilon^2 < \epsilon$.

Let $C_p = C \vee E_p$ be in the support of $\mathbf{C}$, and let $\tilde{E_p}$ be the standard ℓ_2 approximation of the $\mathcal{AC}^0$ circuit E_p (which has depth $d + O(1)$), with error $\delta = 2^{-\log^{O(d)}(m/\epsilon)}$ and degree $\log^{d+O(1)}(m) \cdot \log(1/\delta) = \log^{O(d)}(m/\epsilon)$. [16] We define the approximating polynomial for C_p to be $\tilde{C_p}(x) = p(x) \cdot (1 - \tilde{E_p}(x))$. Note that the degree of $\tilde{C_p}$ is at most $\deg(p) + \deg(\tilde{E_p}) = \log^{O(d)}(m/\epsilon)$. Also note that $\tilde{C_p}$ vanishes on $C_p^{-1}(0)$, since if $C_p(x) = 0$ then $C(x) = E_p(x) = 0$, which implies that $p(x) = 0$ and hence $\tilde{C_p}(x) = 0$.

Now, observe that the probability over $p \sim \mathbf{p}$ that $\Pr_x[E_p(x) = 1] \leq \epsilon$ is at least $1 - \epsilon$. This follows by an averaging argument: Specifically, since for every $x \in \{0,1\}^n$ we have that $\Pr_p[E_p(x) = 1] \leq \epsilon^2$, it follows that $\mathbb{E}_x[\Pr_p[E_p(x) = 1]] \leq \epsilon^2$, and hence $\mathbb{E}_p[\Pr_x[E_p(x) = 1]] \leq \epsilon^2$. Thus, the probability over $p \sim \mathbf{p}$ (and hence also on $C_p \sim \mathbf{C}$) that $\Pr_x[E_p(x) = 1] \geq \epsilon$ is at most ϵ. For every C_p such that $\Pr_x[E_p(x) = 1] \leq \epsilon$, we will show that $\tilde{C_p} = p \cdot (1 - \tilde{E_p})$ ϵ-approximates C_p in ℓ_2-distance.

Claim B.3.5. *For every $C_p \sim \mathbf{C}$ such that $\Pr_x[E_p(x) = 1] \leq \epsilon$ it holds that $\left\|C_p - \tilde{C_p}\right\|_2^2 \leq 4\epsilon$.*

[15] It would have been nicer to define $C_p = C \wedge (\neg E_p)$ and approximate C_p by $p \cdot (1 - \tilde{E_p})$. However, the latter polynomial does not have one-sided error with respect to $C \wedge (\neg E_p)$.

[16] To get such a good dependence of the degree on δ, the classical results of [LMN93; Bop97] do not suffice, and we need the recent improvement from [Tal17].

Proof. We upper-bound $\left\|C_p - \tilde{C}_p\right\|_2$ by $2\sqrt{\epsilon}$, as follows:

$$\left\|C_p - \tilde{C}_p\right\|_2 \leq \left\|C_p - p \cdot (1 - E_p)\right\|_2 + \left\|p \cdot (1 - E_p) - \tilde{C}_p\right\|_2 . \tag{B.3.3}$$

To upper-bound the left term, let $\ell = C_p - p \cdot (1 - E_p)$. Note that for every $x \in E_p^{-1}(0)$ it holds that $p(x) = C_p(x)$, and hence $\ell(x) = 0$; whereas for every $x \in E_p^{-1}(1)$ it holds that $\ell(x) = C_p(x) \in \{0,1\}$. Thus, $\|\ell\|_2 \leq \sqrt{\Pr_x[E_p(x) = 1]} \leq \sqrt{\epsilon}$. To upper-bound the right-term, observe that $p \cdot (1 - E_p) - \tilde{C}_p = p \cdot \left(E_p - \tilde{E}_p\right)$; recalling that $|p(x)| \leq 2^{\log^{O(d)}(m/\epsilon)}$ for every $x \in \{0,1\}^n$ (by Item (2) of Lemma B.3.4), we have that

$$\left\|p \cdot (1 - E_p) - \tilde{C}_p\right\|_2 \leq 2^{\log^{O(d)}(m/\epsilon)} \cdot \left\|E_p - \tilde{E}_p\right\|_2 < 2^{\log^{O(d)}(m/\epsilon)} \cdot \sqrt{\delta} ,$$

and since $\delta = 2^{-\log^{O(d)}(m/\epsilon)}$ the expression can be upper-bounded by $\sqrt{\epsilon}$. ■

We can now complete the proof, as in the outline described above. Let $\mathbf{w}$ be a distribution that is t-wise independent, for $t = \log^{O(d)}(m/\epsilon)$. Using Lemma B.3.1, for $1 - \epsilon$ of the functions C_p in the support of $\mathbf{C}$ it holds that $\mathbb{E}[C_p] - \mathbb{E}_{\mathbf{w}}[C_p] \leq 8\epsilon$. Using the "moreover" part of Lemma B.3.3, it holds that $\mathbb{E}[C] - \mathbb{E}_{\mathbf{w}}[C] \leq 32\epsilon$. And since the class of depth-d circuits is closed to negations, the same argument applies for $1 - C$, and hence $\mathbb{E}_{\mathbf{w}}[C] - \mathbb{E}[C] \leq 32\epsilon$.

B.4 Karp-Lipton Theorems: Translating Non-Uniform Collapses into Uniform Collapses

Abstract. Assume that there exists an infinite sequence of small circuits, one for each input length, that solves some "hard" problem. Can we use this fact to construct a single efficient algorithm that solves some (possibly different) "hard" problem on all input lengths? Put otherwise, does a collapse of some strong uniform class into small *non-uniform circuits* imply an analogous collapse of some (possibly different) strong uniform class into a *seemingly-weaker uniform class*?

This is the general question that underlies many results that are typically referred to as "Karp-Lipton theorems", following the classic result in this spirit by Karp and Lipton [KL80]. My goal in this text is to describe the various ideas that were introduced over the years to try and answer the question.

This survey is written from a technical perspective: More emphasis is given to proof ideas rather than to the results themselves and their implications. Moreover, the text is a bit advanced, and does not fully explain many notions (e.g., PCPs, MIPs, the Nisan-Wigderson PRG, and natural proofs). Nevertheless, I hope that the text can be useful, perhaps for more experienced researchers or for students interested in the specific area.

My warm thanks to Eylon Yogev and to Lijie Chen for very helpful comments on drafts of the text, and to my advisor Oded Goldreich for detailed useful feedback!

B.4.1 Overview, context, and a few examples

The context of Karp-Lipton theorems is the project of trying to prove circuit lower bounds; that is, of trying to prove that "small" *non-uniform* circuits cannot decide every problem in some "strong" *uniform* complexity class.

Karp-Lipton theorems can be thought of as intermediate results towards this goal: Such theorems start from a hypothesis that small non-uniform circuits are unexpectedly strong (we think of this as a "towards a contradiction hypothesis"), and derive conclusions that seem even more unlikely. The latter conclusions are typically in the *uniform* world, and assert that some uniform class is also unexpectedly strong. That is, we begin with the hypothesis that some "hard" problem can be solved by an infinite sequence of small circuits, one for every input length, and we want to deduce that some (possibly different) "hard" problem can be solved by a single efficient uniform algorithm, which handles all (or at least infinitely-many) input lengths. Of course, we hope to continue deriving conclusions that are more and more unlikely, and end up with a contradiction, hence proving circuit lower bounds.

The point in a Karp-Lipton theorem as above is that given such a result, we are now *completely in the uniform world*, and thus hope to have an easier time progressing towards a contradiction. The source of hope is that instead of comparing the *uniform* complexity of the "hard" problem to the *non-uniform* complexity of the circuit family, we are now comparing the *uniform* complexity of the "hard" problem to the *uniform* complexity of the algorithm; that is, we are closer to comparing "apples-to-apples". Ideally, we would like to compare the complexity of the problem and of the algorithm along a *single complexity measure* (e.g., time or space) and obtain a contradiction by a hierarchy theorem; for example, if the "hard" problem is complete for $DTIME[T]$, and we are able to solve it in time $T' \ll T$, then we obtain a contradiction and deduce circuit lower bounds. For a setting where this actually happens, see Section B.4.5.

As pointed out by Oded Goldreich, this way of thinking about Karp-Lipton theorems is relatively new. Originally, the unlikely conclusions of Karp-Lipton theorems were seen as providing evidence that circuit lower bounds are even possible (see, e.g., [Gol08, Sec. 3.2.3]). In contrast, the description above considers the unlikely conclusions as intermediate results on our way to obtain a contradiction and prove circuit lower bounds (where the latter are a-priori believed to be true).

My goal in this text is to present the main technical challenge underlying the proofs of Karp-Lipton theorems, and to describe various ideas that were introduced over the years to overcome this challenge (and that I'm familiar with). Along the way, I'll mention some of the Karp-Lipton theorems that I know of and describe their proofs.

A few examples. Let us begin by stating a few well-known examples of Karp-Lipton theorems and parsing their statements, mostly to get used to the type of results in this field. Instead of beginning with the original theorem by Karp and Lipton, I want to start from subsequent theorems, which more closely resemble recent results.

As a first example, Babai *et al.* [BFN+93] showed that if $\mathcal{EXP} \subset \mathcal{P}/\text{poly}$ (i.e., if

polynomial-sized circuits can decide every problem in $\mathcal{EXP}$), then $\mathcal{MA} = \mathcal{EXP}$. It's useful to read the latter conclusion as "$\mathcal{MA}$ is unexpectedly strong"; this is since $\mathcal{MA}$ is the randomized version of $\mathcal{NP}$, and so we expect it to equal $\mathcal{NP}$ under a sufficiently strong derandomization hypothesis. There are several results that are very similar to this one, which stem from the works of Impagliazzo, Kabanets, and Wigderson [IKW02], of Lund *et al.* [LFK+92] and Shamir [Sha92], and of Kabanets and Impagliazzo [KI04]:[17]

Theorem B.4.1 ([BFN+93; LFK+92; Sha92; IKW02; KI04])**.** *For any uniform class $\mathcal{U} \in \{\mathcal{EXP}, \mathcal{NEXP}, \mathcal{PSPACE}, \mathcal{PP}\}$, if $\mathcal{U} \subset \mathcal{P}/\text{poly}$, then $\mathcal{MA} = \mathcal{U}$. Similarly, if $\#\mathcal{P}$ has polynomial-sized arithmetic circuits, then $\mathcal{MA} = \mathcal{P}^{\#\mathcal{P}}$.*

Another Karp-Lipton theorem underlies the results of Williams [Wil13] and of Murray and Williams [MW18]. For simplicity, let's focus on [Wil13]: Loosely speaking, the underlying Karp-Lipton theorem asserts that for any "typical" circuit class $\mathcal{C}$, if $\mathcal{NEXP} \subset \mathcal{C}$, then $\mathcal{NE}$ can be decided by an $\mathcal{MA}$ protocol that uses a linear number of random coins, and whose residual decision as a function of its random coins can be computed by a $\mathcal{C}$-circuit (for precise details see Section B.4.3.2). These results are especially interesting since for some classes $\mathcal{C}$ we can "push forward" this intermediary conclusion to actually derive a contradiction, and deduce lower bounds for $\mathcal{C}$.

Yet another example comes from the work of Impagliazzo and Wigderson's [IW99]. The Karp-Lipton theorem that underlies their main result asserts that if $\mathcal{EXP} \subset \mathcal{P}/\text{poly}$ *and* the Nisan-Wigderson generator, instantiated with the permanent function and for polynomial stretch, does *not* fool uniform polynomial-time algorithms, then $\mathcal{BPP} = \mathcal{EXP}$ (for precise details and various extensions see Section B.4.4.2). Note that this result differs from the results above, since it uses an additional hypothesis (that the NW generator doesn't fool uniform algorithms) but is then able to construct a $\mathcal{BPP}$ algorithm, rather than just an $\mathcal{MA}$ verifier, for $\mathcal{EXP}$.

As a last example (in this non-exhaustive list), we can also consider the *original* theorem by Karp and Lipton [KL80] as following the same template: If $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, then "$\Sigma_2$ is too strong", in the sense that Σ_2 can decide Π_2 and therefore all of the polynomial-time hierarchy. I discuss this theorem in Section B.4.3.1, and mention various extensions and improvements in Sections B.4.3.1, B.4.4.1 and B.4.6.

Lower bounds via a win-win analysis. A classic approach to deduce unconditional lower bounds from Karp-Lipton theorems is a win-win analysis first introduced by Kannan [Kan82]. As an example, recall the original Karp-Lipton theorem "$\mathcal{NP} \subset \mathcal{P}/\text{poly} \Rightarrow \Sigma_2 = \mathcal{PH}$". The win-win analysis in Kannan's theorem allows us to deduce from this result that for any fixed $k \in \mathbb{N}$ it holds that $\Sigma_2 \not\subset SIZE[n^k]$. This is done by analyzing two cases: Either $\mathcal{NP} \not\subset \mathcal{P}/\text{poly}$, in which case we are done; or $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, which implies that $\Sigma_2 = \mathcal{PH}$. Now, since we already know that $\mathcal{PH} \not\subset SIZE[n^k]$ for any fixed k (by a straightforward diagonalization argument, see

[17] Some of the results aren't stated in this fashion in the original papers, but are well-known to follow as corollaries; I'll explain how in Section B.4.3.

e.g. [Juk12, Lem. 20.12]), in the second case we can deduce that $\Sigma_2 \not\subset SIZE[n^k]$ for any fixed k. Put together, for any fixed k it holds that $\Sigma_2 \not\subset SIZE[n^k]$.

Similar win-win analyses has been used to deduce various unconditional lower bounds from Karp-Lipton theorems (see, e.g., [KW99; Vin05; Aar06; San09]); in fact, it turns out that in some settings Karp-Lipton theorems are *necessary* in order to deduce the lower bounds that can be obtained by this type of win-win analysis (see the recent paper by Chen *et al.* [CMM+19]). However, as my focus in this text is on how to *prove* Karp-Lipton theorems, I will not survey these results here.

Organization. In Section B.4.2 I'll describe the main technical challenge underlying Karp-Lipton theorems, and in Sections B.4.3 and B.4.4 I'll survey some general ideas that were introduced to overcome this challenge. In Section B.4.5 I'll survey one way of "pushing forward" the conclusions of Karp-Lipton theorems (other than win-win analyses as described above), namely by also considering a *derandomization* hypothesis; this approach yields proofs that derandomization imply circuit lower bounds. Finally, in Section B.4.6 I'll briefly mention a few results that consider natural variations of the basic question underlying Karp-Lipton theorems.

B.4.2 The core technical challenge

The core technical challenge that underlies Karp-Lipton theorems is the following. Assume that some small circuit class is unexpectedly strong; for example, assume that $\mathcal{EXP} \subset \mathcal{P}/\text{poly}$. Using this hypothesis, we want to show that some uniform class, say $\mathcal{MA}$, is unexpectedly strong. Of course, to do so we want to leverage the hypothesis that there exists a family of small circuits for some "very hard" problem $L \in \mathcal{EXP}$. The core issue is that, while we are guaranteed that circuits for L exist, we are not guaranteed that we can *efficiently find* such a circuit that solves L. So how can a uniform algorithm make any use of the hypothesis that the circuit exists, if it might be difficult for the algorithm to get its hands on such a circuit?

The naive solution is to go over all relevant circuits, and for each circuit test whether or not it has the correct functionality. The trouble with this approach is not only that exhaustive search is inefficient, but also that it is not even clear how to efficiently verify that a given circuit has the expected functionality. Thus, we are faced with two separate problems: The problem of *verifying* that a given circuit has the expected functionality, and the problem of *finding* a circuit with the functionality that we want. That is, we have a decision problem and a corresponding search problem.

The main point of this text is to survey some known general ideas for solving these problems, and to state corresponding results that are obtained. It's worthwhile to note in advance that the majority of solutions only address the verification problem, without solving the search problem. Such solutions yield uniform algorithms that are *non-deterministic*: The algorithms use their non-determinism to obtain a candidate circuit, then efficiently verify the circuit's functionality, and finally use the circuit to solve some hard problem. For some (partial) explanation why most known algorithms

solve only the verification problem and not the search problem, see Section B.4.4.

Let me stress that we are interested in verifying that the circuit correctly computes the corresponding function on *all inputs*; that is, we are verifying that the circuit is correct in "worst-case". One may also consider a relaxed setting, in which we only wish to verify that the circuit is correct on average-case (e.g., that the circuit behaves correctly on 99% of the inputs). Indeed, the latter notion has been extensively studied, and is related to program checking, property testing, error-correcting codes, and PCPs; needless to say, I have no intention of trying to survey those here. A natural strategy to solve the worst-case problem is to try and first verify that a circuit is correct on average-case, and then use a worst-case to average-case reduction to obtain a circuit that is correct on worst-case. One result that uses such a strategy (by Impagliazzo and Wigderson [IW98]) will be presented in Section B.4.4.

B.4.3 Verifying the functionality of a circuit

In this section, I'll survey some of the ideas used to construct algorithms that verify that a given circuit has the expected functionality. As one might expect when dealing with such verification, *the main technical tool used by most of the solutions is* **proof systems** *with specific useful properties.*

The specific proof systems that are used in the results in this section are the interactive proof underlying $\mathcal{IP} = \mathcal{PSPACE}$ (see [LFK+92; Sha92]), the two-prover MIP for $\mathcal{EXP}$ (see [BFL91]), and PCPs with almost-linear proof length and very efficient verifiers (see [BGH+05; BSV14]).

B.4.3.1 Proof systems with efficient provers

Assume that we want to decide a set L in some large uniform class $\mathcal{U}$, under the "collapse" hypothesis that $\mathcal{U}$ (or some other large uniform class $\mathcal{U}'$) has small circuits. The first idea leverages the existence of a proof system for $\mathcal{U}$ such that *the strategy of the prover (or provers) in this proof system can be implemented by an efficient algorithm* (e.g., the prover strategy is in $\mathcal{U}$). The idea is that instead of expecting to receive (and verify) a small circuit that computes L, for every input x, we will expect to receive a small circuit P *that implements the prover strategy* for L at x. Indeed, the main point is that now we can efficiently verify that the circuit P has the expected functionality (since, by definition, this expected functionality can be verified in a proof system). To recap the main idea, *instead of verifying circuits for L, we verify "prover-circuits" for L.*

The main advantage in this approach is that the resulting verifier is more efficient than the verification protocol in the original proof system. For example, the original verifier might have needed many rounds of interaction with the prover (as in the proof system underlying $\mathcal{IP} = \mathcal{PSPACE}$), or might have needed to simultaneously interact with several provers that do not communicate between them (i.e., the original system was an MIP, as in the two-prover MIP for $\mathcal{EXP}$). In contrast, in our setting, the verifier just receives a small circuit P that implements the prover strategy, and simulates the

proof system by itself, by "interacting" only with P.[18]

Indeed, this essentially sums up the proofs of three of the results stated in Theorem B.4.1. Specifically, the result that $\mathcal{EXP} \subset \mathcal{P}/\text{poly} \Rightarrow \mathcal{MA} = \mathcal{EXP}$ implements this proof approach using the MIP proof system of [BFL91] for $\mathcal{EXP}$ whose prover strategies are computable in $\mathcal{EXP}$; the result that $\mathcal{PSPACE} \subset \mathcal{P}/\text{poly} \Rightarrow \mathcal{MA} = \mathcal{PSPACE}$ uses this proof approach with the proof system of [Sha92; LFK+92] for $\mathcal{PSPACE}$, whose prover strategy is computable in $\mathcal{PSPACE}$; and similarly, the result that $\mathcal{PP} \subset \mathcal{P}/\text{poly} \Rightarrow \mathcal{MA} = \mathcal{PP}$ uses this proof approach with the proof system of [LFK+92] for $\mathcal{PP}$, whose prover strategy is computable in $\mathcal{P}^{\#\mathcal{P}} = \mathcal{P}^{\mathcal{PP}}$.

This proof approach fails when trying to prove that if $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$ then $\mathcal{NEXP} = \mathcal{MA}$: The reason is that we do not know of an MIP proof system for $\mathcal{NEXP}$ with provers whose strategies are computable in $\mathcal{NEXP}$. (In the MIP proof system for $\mathcal{NEXP}$ of [BFL91] the complexity of the provers isn't known to be in $\mathcal{NEXP}$, since the two provers need to *independently* construct a *single, identical proof* that serves as the basis for their communication with the verifier.) However, we can still use this strategy to prove a seemingly-weaker statement:

Proposition B.4.2 (folklore; see, e.g., [Wil13],[FSW09, Sec. 4])**.** *If* $\mathcal{EXP}^{\mathcal{NP}} \subset \mathcal{P}/\text{poly}$ *then* $\mathcal{MA} = \mathcal{NEXP}$*.*

Proof. Consider a PCP proof system for $\mathcal{NEXP}$ with proof length $2^{\text{poly}(n)}$ and a verifier that runs in polynomial time (e.g., the PCP of [BGH+05]). Observe that for every $L \in \mathcal{NEXP}$ there exists an $\mathcal{EXP}^{\mathcal{NP}}$ algorithm that, on input (x, i) where $x \in L$, finds the lexicographically-first proof for x in the foregoing PCP system, and returns the i^{th} bit in this proof.[19] Now, under the hypothesis $\mathcal{EXP}^{\mathcal{NP}} \subset \mathcal{P}/\text{poly}$ there exists a polynomial-sized circuit that solves the latter problem. Thus, the verifier in the PCP system can be turned into an $\mathcal{MA}$ verifier: On input $x \in \{0,1\}^n$ the $\mathcal{MA}$ verifier gets a polynomial-sized circuit $P : \{0,1\}^{n+\text{poly}(n)} \to \{0,1\}$ that represents a PCP proof, and simulates the execution of the PCP verifier on x while answering its queries using the circuit P (i.e., query $i \in [2^{\text{poly}(n)}]$ is answered by $P(x, i)$). ■

Proposition B.4.2 demonstrates another way in which this proof approach can yield a verifier that is more efficient in the original proof system. Specifically, recall that in the previous examples the advantage was that we replaced complicated interaction (e.g., many rounds, or interaction with two provers) with a single "static" proof. In contrast, in Proposition B.4.2 the main advantage is that the *new proof is shorter than the original one* (i.e., instead of a PCP proof of size $2^{\text{poly}(n)}$, the new proof is a description

[18] As Ron Rothblum eloquently put it, we "pack the prover in a box" (the box is the small circuit) and ship it to the verifier.

[19] Specifically, the $\mathcal{EXP}^{\mathcal{NP}}$ algorithm constructs the lexicographically-first proof bit-by-bit, where each decision problem is of the form "does there exist a continuation π of the current prefix σ such that the PCP verifier accepts x with proof π". To solve each such decision problem, the $\mathcal{EXP}$ algorithm sends a query of the form $(1^{\text{poly}(n)}, x, \sigma)$ to the $\mathcal{NP}$ oracle, and the $\mathcal{NP}$ oracle guesses a continuation π of σ and enumerates over the $2^{\text{poly}(n)}$ possible values for coin tossess of the PCP verifier to determine the decision of the verifier at input x with proof π.

of a circuit of size poly(n)). That is, in Proposition B.4.2 we used our hypothesis to deduce that in a suitable proof system, valid witnesses can be represented in a concise manner (i.e., by a small circuit). This approach is the basis for the "easy witness" method, which will be presented in Section B.4.3.2.

One can also view the proof of the original Karp-Lipton theorem as following the same proof approach; that is, utilizing a "proof system with efficient provers" for Π_2 in order to compute any $L \in \Pi_2$ in Σ_2, under the collapse hypothesis. To do so, recall that for $L \in \Pi_2$ we have that $x \in L$ iff $\forall w_1 \exists w_2 : V(x, w_1, w_2) = 1$, where V is an algorithm running in time poly$(|x|)$. Now, consider a "proof system" in which to verify that $x \in L$, the verifier issues a challenge (x, w_1) to a prover, and the prover must reply with a correct answer w_2 such that $V(x, w_1, w_2) = 1$; the system "accepts" x if for every challenge w_1 there is a valid answer w_2. The point is that the prover strategy in this system is efficient, in the sense that it is in $\mathcal{NP}$ (i.e., the relation $\{((x, w_1), w_2) : V(x, w_1, w_2) = 1\}$ is an $\mathcal{NP}$-relation). In particular, if $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, then there is a polynomial-sized circuit that gets as input (x, w_1) and outputs a valid w_2. In this case, the "Σ_2-verifier" in this proof system can first guess a circuit P for the "prover", and then choose a challenge w_1, feed it to this circuit, obtain w_2, and verify that $V(x, w_1, w_2) = 1$. Indeed, we thus have that $x \in L$ iff there exists a "prover-circuit" P such that for all w_1, the residual verification procedure (that chooses w_1, feeds it to P, and outputs $V(x, w_1, P(x, w_1))$) accepts. It follows that $\Sigma_2 = \Pi_2$. [20]

An "overkill" in the approach. In the above results, I hand-waved a bit when referring to the notion of "prover strategy". This notion can be formalized in a straightforward way, for example by considering the set (x, r, i) where x is the input, r is the communication from the verifier, and i is the index of the bit in the prover's response. Observe, however, that when using this approach the hypothesis gives us more than we actually need: Specifically, when using this approach we deduce that there exists *a single circuit* P that gets x as part of its input and implements the corresponding prover strategy for x. This is an "overkill", since the verifier only needs that *for every input* x *there exists a circuit* P_x that implements the corresponding prover strategy.

B.4.3.2 Concise representations of proofs ("easy witnesses")

In this section our goal is to prove Karp-Lipton theorems in which the resulting "unexpectedly strong" uniform algorithm (which will again be essentially an $\mathcal{MA}$ verifier) decides a class $\mathcal{U}$ for which *we are not able to construct proof systems with sufficiently efficient provers* (i.e., the collapse hypothesis does not immediately suffice to deduce that there exist small circuits implementing the prover's strategy).

[20] In fact, as noted by Sengupta and reported by Cai [Cai07], the same proof actually yields that Π_2 can be simulated in the "symmetric alternations class" $\mathcal{S}_2$, rather than only in Σ_2. This is the case because we can consider one prover that sends w_1, and another prover that sends the circuit P. If $x \in L$, then no matter what w_1 the first prover sends, the second prover can send a P that we will accept; and if $x \notin L$, then there exists a w_1 that the first prover can send, for which no acceptable P exists.

To overcome this challenge, the solutions in this section exploit the relaxation mentioned in the end of the previous section. Specifically, for any $L \in \mathcal{U}$, we will fix a suitable proof system for L and use the collapse hypothesis to deduce the following: For every $x \in L$ there exists a circuit P_x that gets as input (r, i) where r is the communication from the verifier and i is a location in the prover's response (the input r can be omitted in proof systems where the verifier doesn't sends any message to the prover), and outputs the i^{th} bit of the corresponding response from the prover. That is, instead of one circuit P for every input x, we only deduce that for every input x there exists a "prover circuit" P_x. Armed with such a result, we can proceed just as in the previous section: The verifier gets input x, guesses a prover-circuit P_x, and interacts with P_x instead of with an actual prover. However, the main challenge will be *proving* that such P_x's exist, which is far less straightforward than in the previous section.

Note that when the proof system is just a "static" proof (i.e., the verifier doesn't send any message to the prover, and the input of P_x is just the location i), the circuit P_x can be thought of as a concise representation of a potentially-long proof; in other words, the truth-table of P_x is the "static" proof. In this case, the proof/witness is "easy" in the sense that is has a concise representation by a circuit.

The original "easy witness lemma", proved by Impagliazzo, Kabanets, and Wigderson [IKW02], asserts the following: *If $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$, then for every $L \in \mathcal{NEXP}$, every exponential-time verifier for L, and every $x \in L$, there exists a witness $w_x \in \{0,1\}^{2^{\text{poly}(n)}}$ for x that the verifier accepts such that w_x can be concisely represented by a circuit P_x of polynomial size.* This is already enough to prove the following:

Theorem B.4.3 ([IKW02]). *If $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$, then $\mathcal{MA} = \mathcal{NEXP}$.*

Proof. Let $L \in \mathcal{NEXP}$, and consider the PCP proof system of [BGH+05] for $\mathcal{NEXP}$, in which the proof length is $2^{\text{poly}(n)}$ and the verifier V runs in polynomial time. Using the easy witness lemma, we deduce that for every $x \in L$ there exists a witness for x in this proof system that can be represented by a polynomial-sized circuit.[21] Now, the $\mathcal{MA}$ verifier gets a polynomial-sized circuit as proof, and simulates the PCP verifier while using this circuit to simulate the proof.[22] ■

The original easy witness lemma used a strong hypothesis (i.e., $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$) and deduced a strong conclusion (i.e., $\mathcal{NEXP}$ has witnesses of polynomial size). The state-of-the-art easy witness lemma, proved by Murray and Williams [MW18], allows to scale the parameters in both the hypothesis and the conclusion; specifically, it allows to replace "exponential-time" and "polynomial-size" with any two functions t and s

[21] To do so, consider the deterministic exponential-time verifier V' that enumerates over the coins of the PCP verifier V. The easy witness lemma implies that for every $x \in L$ there exists a witness w_x that V' accepts and that can be represented by a polynomial-sized circuit. Since V' and V accept precisely the same witnesses, the witness w_x is also accepted by V.

[22] The original proof of [IKW02] is different. They note that from the easy witness lemma it follows that $\mathcal{NEXP} = \mathcal{EXP}$ (since one can enumerate over all polynomial-sized circuits to search for a witness). Then, since $\mathcal{EXP} = \mathcal{NEXP} \subset \mathcal{P}/\text{poly}$ we have (by Theorem B.4.1) that $\mathcal{NEXP} = \mathcal{EXP} = \mathcal{MA}$.

(respectively) such that $t \gg s$. (For simplicity I will omit the precise parameters, but the requirements are roughly that $t(n) \geq s^{O(1)}(s^{O(1)}(s^{O(1)}(n)))$.) That is:

Lemma B.4.4 (the easy witness lemma of [MW18]; informal)**.** *Let $\hat{t} \gg t \gg \hat{s} \gg s$ be "nice" functions (i.e., time-computable functions that are bounded by $2^{\epsilon \cdot n}$ for some universal $\epsilon > 0$). Then, assuming that $\mathcal{NTIME}[\hat{t}] \subset SIZE[s]$, for every set $L \in \mathcal{NTIME}[t]$, and every time-t verifier, and every $x \in L$, there exists a witness for x that can be concisely represented by a circuit of size $\hat{s}$.*

Note that the circuit of size $\hat{s}$ in Lemma B.4.4 is indeed a concise representation of the witnesses in the proof system, since the latter are potentially of size $t \gg \hat{s}$. Moreover, Lemma B.4.4 also holds if the hypothesis refers not to general circuits, but to circuits from some restricted circuit class $\mathcal{C}$ (e.g., $\mathcal{C}$ can be $\mathcal{AC}^0, \mathcal{ACC}^0, \mathcal{TC}^0$), i.e. $\mathcal{NTIME}[\hat{t}] \subset \mathcal{C}\text{-}SIZE[s]$, in which case we get the stronger conclusion that $\mathcal{NTIME}[t]$ has witness circuits from $\mathcal{C}$ of size $\hat{s}$.

Let us now see how to use the easy witness lemma of [MW18], along with PCPs with almost-linear proof length, in order to get a Karp-Lipton theorem that is more refined than the result stated in Theorem B.4.3. Loosely speaking, the following result asserts that for any "typical" circuit class $\mathcal{C}$, if $\mathcal{NTIME}[\hat{t}] \subset \mathcal{C}$, then $\mathcal{NTIME}[t]$ can be decided by an $\mathcal{MA}$ verifier that uses "few" random coins, and whose residual decision as a function of its random coins can be computed by a $\mathcal{C}$-circuit of size approximately $\hat{s}$. More accurately, using the proof approach of Williams [Wil13] with the PCP of Ben-Sasson and Viola [BSV14] and the easy witness lemma of Murray and Williams [MW18], we get that:

Theorem B.4.5 (informal, implicit in [Wil13; BSV14; MW18])**.** *Let $\hat{t} \gg t \gg \hat{s} \gg s$ be "nice" functions, and assume that $\mathcal{NTIME}[\hat{t}] \subset \mathcal{C}\text{-}SIZE[s]$. Then, for every $L \in \mathcal{NTIME}[t]$ there exists an $\mathcal{MA}$ verifier for L with the following properties. On input $x \in \{0,1\}^n$:*

1. *The verifier runs in time* $\mathrm{poly}(\log(t(n)), \hat{s}(n), n)$.
2. *The verifier receives a proof, constructs a $\mathcal{C}$-circuit $C_x : \{0,1\}^{\log(t)+O(\log\log(t))} \to \{0,1\}$ of size* $\mathrm{poly}(\hat{s}, n)$*, chooses random coins $r \in \{0,1\}^{\log(t)+O(\log\log(t))}$, and outputs $C_x(r)$.*

Note that in Theorem B.4.5, for any proof π that the verifier receives, the acceptance probability of the circuit C_x equals the probability that the verifier accepts x with proof π. Thus, if we could distinguish between $\mathcal{C}$-circuits with high acceptance probability and $\mathcal{C}$-circuits with low acceptance probability (where "high" and "low" correspond to the completeness and to the soundness of the $\mathcal{MA}$ verifier, respectively) by an *efficient deterministic* algorithm, we could replace the $\mathcal{MA}$ verifier with a deterministic verifier. (For further details see Section B.4.5.)

Proof sketch for Theorem B.4.5. The proof is a refinement of the proof of Theorem B.4.3. Using the PCP of Ben-Sasson and Viola [BSV14], every $L \in \mathcal{NTIME}[t]$ has a PCP verifier that runs in time $\mathrm{poly}(\log(t), n)$, uses $\log(t) + O(\log\log(t))$ random coins, and

can be implemented by a very simple $\mathcal{AC}^0$ circuit. Using the easy witness lemma of [MW18], for every $x \in L$ there exists a $\mathcal{C}$-circuit P_x of size $\hat{s}$ that represents a valid witness for x in this proof system. The machine thus guesses this circuit, and then constructs a circuit C_x that gets as input random coins $r \in \{0,1\}^{\log(t)+O(\log\log(t))}$, computes the corresponding queries made by the PCP verifier on coins r, answers these queries using P_x as a sub-circuit, and finally computes and outputs the corresponding decision by the PCP verifier. ■

To recap, we stated easy witness lemmas and saw how corresponding Karp-Lipton theorems can be derived from those lemmas. For a (very rough) proof sketch of the state-of-the-art easy witness lemma of [MW18] see Appendix B.4.7.

B.4.3.3 Arithmetic circuit lower bounds against "highly structured" functions

A completely different strategy aims at proving lower bounds for *arithmetic circuits*, instead of Boolean circuits. In this context arithmetic circuits are circuits with gates labeled by $\{+,\times\}$ that are evaluated as polynomials (in their inputs) over $\mathbb{Z}$. The main advantage in this setting is that *one can efficiently test some specific functionalities of a given arithmetic circuit*. Specifically, relying on the Schwartz-Zippel lemma (and on an additional trick), one can test in probabilistic polynomial time whether or not a given circuit computes the constant zero polynomial (see [IM83; KI04]).

Of course, assuming that small arithmetic circuits can solve some "hard" function f, our goal is not to test whether a given arithmetic circuit computes the zero polynomial, but rather to test whether or not the circuit computes f. Nevertheless, if the function f has some specific useful structure, we can reduce the latter task to the former task; that is, we can test whether or not a given arithmetic circuit computes f by testing whether or not certain auxiliary arithmetic circuits compute the zero polynomial. Kabanets and Impagliazzo [KI04] showed how to do so in the case of the permanent function (since their proof is short and self-contained, I'm including it below). It follows that if the permanent has small arithmetic circuits, then a verifier can get a small arithmetic circuit as proof, efficiently verify that this circuit computes the permanent, and then use the circuit to efficiently solve any problem in $\mathcal{P}^{\#\mathcal{P}}$. Hence, if the permanent has polynomial-sized arithmetic circuits, then $\mathcal{MA} = \mathcal{P}^{\#\mathcal{P}}$.

Lemma B.4.6 (verifying that an arithmetic circuit computes the permanent [KI04])**.** *The task of verifying that a given arithmetic circuit $p_n : \mathbb{Z}^{n^2} \to \mathbb{Z}$ computes the permanent function reduces in deterministic polynomial time to the task of testing whether certain (auxiliary) arithmetic circuits compute the zero polynomial.*

Proof. We think of the n^2 input variables to p_n as an $n \times n$ matrix. For every $i \in [n]$, let p_i be the polynomial that is obtained from p_n by fixing all input variables outside the bottom-right $i \times i$ matrix such that the $n - i$ variables along the top-left diagonal are fixed to one, and all other $n^2 - i^2 - (n - i)$ variables are fixed to zero.

Note that for every $i \in [n]$ we can easily convert the circuit that computes p_n to a circuit that computes p_i (by fixing $n^2 - i^2$ of the variables). Also observe that p_n

computes the permanent if and only if $p_1(x) = x$, and for every $i \in \{2,...,n\}$ it holds that $p_i(X) = \sum_{j\in[i]} x_{1,j} \cdot p_{i-1}(X_j)$, where X is an $i \times i$ input matrix, and X_j is the matrix obtained from X by erasing the top row and the j^{th} column. Thus, verifying that p_n computes the permanent is equivalent to verifying that the circuit $h_1(x) \stackrel{\text{def}}{=\!=} p_1(x) - x$ computes the zero polynomial, and that for every $i \in \{2,...,n\}$ it holds that the circuit $h_i(X) \stackrel{\text{def}}{=\!=} p_i(X) - \sum_{j\in[i]} x_{1,j} \cdot p_{i-1}(X_j)$ computes the zero polynomial. ■

As noted in [KI04], one can in fact "merge" the n tests for $h_1,...,h_n$ in the proof of Lemma B.4.6 into a single test: Specifically, it suffices to test whether the circuit $h(X^{(1)}, X^{(2)}, ..., X^{(n)}, y) \stackrel{\text{def}}{=\!=} h_1(X^{(1)}) \cdot y^{n-1} + h_2(X^{(2)}) \cdot y^{n-2} \cdot ...h_n(X^{(n)})$ computes the zero polynomial, where each $X^{(i)}$ is a set of i^2 variables and y is an auxiliary variable.

B.4.4 Efficiently finding a circuit with prescribed functionality

This section focuses on the problem of *finding* a circuit with prescribed functionality, assuming that such a circuit exists. That is, we are now interested in a *search* problem.

Formalization of the problem. For a fixed target function $f : \{0,1\}^* \to \{0,1\}$, we define a corresponding search problem in which the inputs are pairs of the form $(1^n, 1^s)$, and the output is either a size-s circuit that computes $f_n = f \cap \{0,1\}^n$, or "fail", if no such circuit exists. In other words, the "yes" inputs are such that a size-s circuit exists for f_n, and the "no" inputs are such that no size-s circuit can compute f_n.

We will also consider two relaxations of this problem. The first relaxation is a "promise" version of the problem, in which the "yes" inputs are defined as above (i.e., a size-s circuit exists for f_n), but the "no" inputs are $(1^n, 1^s)$ such that no circuit of size $r(s(n)) > s(n)$ can compute f_n, for some predetermined function $r : \mathbb{N} \to \mathbb{N}$ (e.g., $r(s(n)) = s^2(n)$). The second relaxation is that on "yes" inputs we will allow the algorithm to output circuits of size larger than s, where this again refers to some predetermined function $r' : \mathbb{N} \to \mathbb{N}$ that defines the allowed circuit size $r'(s(n)) > s(n)$. We say that an algorithm constructs circuits for f if it solves the relaxed search problem with functions r and r' that will typically be clear from context.

One may think of algorithms that construct circuits for f as sub-routines of other algorithms. For example, we can use an algorithm that constructs circuits for f to obtain an algorithm that gets input 1^n and outputs a circuit for f_n of approximately minimal size (by trying to construct circuits with increasing values of s); this is particularly interesting when the "minimal size function" for f behaves oddly (e.g., the "minimal size function" might not be time-computable; for a natural example of a setting where this problem arises see [San09]). And, continuing our motivation from Sections B.4.2 and B.4.3, algorithms that construct circuits for f can be used to *decide* f. However, in this section I'll focus on the search problem per-se.

Is the search problem difficult? Let's begin by asking why this search problem seems so difficult in the first place; after all, shouldn't a search-to-decision reduction be possible in this context? An initial observation is that the standard reduction, in which we construct a string bit-by-bit, reduces the search problem to a more difficult decision problem than merely verifying that a given circuit has the expected functionality; specifically, the target of the standard reduction is the decision problem of deciding whether a given string is a *prefix* of such a circuit, and it is not clear how to efficiently solve the latter decision problem.

A second observation is that constructing circuits is at least as difficult as *learning*. Specifically, in this section when we say learning we mean the problem of learning a function f with membership queries over the uniform distribution: In this problem we are given oracle access to f, and want to efficiently find a "small" circuit that agrees with f on almost all inputs. Constructing a circuit for f is obviously at least as hard as learning f, and in fact seems even much *more difficult*, since:

1. We may not have oracle access to f. (Recall that our goal when constructing circuits for f is typically to decide f, i.e. evaluate f at a given input.)

2. We typically want to find a circuit that computes f in worst-case, rather than just in average-case (as is the task of a learning algorithm.)

Moreover, even if we can somehow solve the two foregoing obstacles, we are still faced with a major obstacle, since *efficient learning algorithms for "hard" functions are unlikely to exist unconditionally*. Specifically, recall that under reasonable hypotheses, there does not exist a learning algorithm for any sufficiently rich circuit class (e.g., for the class of all polynomial-sized circuits; this is since a learning algorithm can be used to break a pseudorandom function). To recap this obstacle, any algorithm that efficiently constructs circuits also yields an efficient learning algorithm, but an efficient learning algorithm probably does not unconditionally exist.

The first set of solutions that I'll present are unconditional constructions of *relatively inefficient* algorithms for constructing circuits, where the inefficiency is what allows us to bypass the PRF obstacle. Specifically, note that there is a naive algorithm that constructs circuits for *any* function f, and works in Σ_2^f (i.e., the Σ_2^f algorithm asks "does there exist a size-s circuit that for every input correctly computes f?").[23] The first set of algorithms improve on this naive construction, by improving the "base class" from Σ_2 to various subclasses of Σ_2 that contain $\mathcal{P}^{\mathcal{NP}}$.

The second idea to solve the problem, by Impagliazzo and Wigderson [IW98], *uses its unlikely hypotheses* to deduce that there exists a suitable learning algorithm for a "hard" function. Relying on the (conditional) existence of such a learning algorithm, they are indeed able to efficiently construct a (probabilistic) circuit for any f that has several useful properties (see Section B.4.4.2 for details).

[23] Here and throughout the section, I abuse the notation of classes of decision problems (such as Σ_2) by using it to refer to the underlying machines (which are now used to solve search problems).

B.4.4.1 Unconditional algorithms between $\mathcal{P}^{\mathcal{NP}}$ and Σ_2

In this section I'll present some *unconditional* constructions of algorithms that construct circuits for *general functions*; that is, the algorithms that I'll present can construct circuits for *any* function $f : \{0,1\}^* \to \{0,1\}$.

As mentioned above, there is a naive solution in Σ_2^f. Intuitively, in the algorithms that I'll describe, the "base class" is improved from Σ_2 to various subclasses of Σ_2 that contain $\mathcal{P}^{\mathcal{NP}}$; for example, the "base class" is improved to $\mathcal{ZPP}^{\mathcal{NP}}$, or to the "alternating symmetric class" $\mathcal{S}_2$ (see below). However, none of these algorithms actually improves the "base class" all the way to $\mathcal{P}^{\mathcal{NP}}$ (i.e., the "base class" is always larger than $\mathcal{P}^{\mathcal{NP}}$). In fact, proving the unconditional existence of such an algorithm would imply the unconditional lower bound $\mathcal{P}^{\mathcal{NP}} \not\subset SIZE[n^k]$ for any fixed $k \in \mathbb{N}$, which is currently not known (for a precise statement and a proof see the end of Section B.4.4.1).

Indeed, even a $\mathcal{P}^{\mathcal{NP},f}$ algorithm might seem a-priori redundant after seeing the verification solutions in Section B.4.3: Since we use non-determinism, we might as well just get a circuit as proof and verify its functionality, as in the verification algorithms surveyed in Section B.4.3. However, the solutions in Section B.4.3 all used randomness, whereas some of the solutions in this section (let alone, a solution in $\mathcal{P}^{\mathcal{NP},f}$) do not use randomness. In addition, the solutions in Section B.4.3 required the existence of a proof system for f with specific useful properties, whereas the algorithms in the current section work for any function that has small circuits. And lastly, most of the solutions in Section B.4.3 allowed us to merely *compute* the target function f (with the assistance of a "prover-circuit"), whereas the solutions in this section allow us to obtain a small circuit that in itself computes f on all inputs in $\{0,1\}^n$.

A simple algorithm in $\mathcal{S}_2^f$. Recall that a set $L \subseteq \{0,1\}^*$ is in the "alternating symmetric class" $\mathcal{S}_2$ if there exists a polynomial-time algorithm V such that the following two conditions hold:

- For every $x \in L$ there exists w_1 such that for all w_2 we have that $V(x, w_1, w_2) = 1$.
- For every $x \notin L$ there exists w_2 such that for all w_1 we have that $V(x, w_1, w_2) = 0$.

Also recall that $\mathcal{P}^{\mathcal{NP}} \subseteq \mathcal{S}_2 \subseteq \mathcal{ZPP}^{\mathcal{NP}}$ (the second containement is by a result of Cai [Cai07]; see also [FIK+08]), and so under the derandomization hypothesis $\mathcal{ZPP}^{\mathcal{NP}} = \mathcal{P}^{\mathcal{NP}}$ we have that $\mathcal{S}_2 = \mathcal{P}^{\mathcal{NP}}$.

The first algorithm that I'll present constructs circuits for any function f and works in $\mathcal{S}_2^f$. This construction is natural and is probably known, but I am not aware of any previous written reference to it.

Theorem B.4.7 (constructing a circuit for f in $\mathcal{S}_2^f$)**.** *There exists a polynomial-time verifier V such that for every $f : \{0,1\}^* \to \{0,1\}$, when given input $(1^n, 1^s)$ and oracle access to f on n-bit inputs, the verifier satisfies the following:*

1. *If there exists a size-s circuit for f_n, then there exists $w_1 \in \{0,1\}^{\mathrm{poly}(s)}$ such that for every $w_2 \in \{0,1\}^{\mathrm{poly}(s)}$ it holds that $V(1^n, 1^s, w_1, w_2)$ outputs such a circuit.*

2. *If every circuit of size $O(s \cdot n)$ fails to compute f_n, then there exists $w_2 \in \{0,1\}^{\mathrm{poly}(s)}$ such that for every $w_1 \in \{0,1\}^{\mathrm{poly}(s)}$ it holds that $V(1^n, 1^s, w_1, w_2)$ outputs "fail".*

Proof. The verifier V expects to get w_1 that is a description of a circuit $C : \{0,1\}^n \to \{0,1\}$ of size s and w_2 that is a set of $r = \mathrm{poly}(s)$ inputs $x_1, ..., x_r \in \{0,1\}^n$. The verifier accepts, and outputs $C = w_1$, if and only if $C(x_i) = f(x_i)$ for every $i \in [r]$. Note that V is computable in polynomial time (in the input length s) with oracle access to f.

Note that if there exists a circuit of size s that computes f on $\{0,1\}^n$, then w_1 can be a description of this circuit, and no matter which inputs w_2 represents, the verifier will accept and output C.

On the other hand, if every circuit of size $O(s \cdot n)$ fails to compute f on $\{0,1\}^n$, then by a variation of a result of Lipton and Young [LY94, Thm. 6] the following holds: There exists a collection $S \subseteq \{0,1\}^n$ of r inputs such that every circuit of size s fails to compute f somewhere on S. [24] Given such an S, the verifier will reject any size-s circuit C that is represented by w_1. ∎

Algorithms that use a non-deterministic oracle. I'll now mention several efficient algorithms that make crucial use of a non-deterministic oracle (i.e., an $\mathcal{NP}$ oracle or an $\mathcal{AM}$ oracle) and/or of oracles that are related to the target function f. Let me note in advance that some of these algorithms are superseded by the simple algorithm from Theorem B.4.7, but it's likely that the ideas underlying their proofs can nevertheless be useful. (The fact that these algorithms are superseded by Theorem B.4.7 relies on the fact that $\mathcal{S}_2 \subseteq \mathcal{ZPP}^{\mathcal{NP}}$, which wasn't known until 2007 [Cai07].)

The first algorithm by Bshouty *et al.* [BCG+96] constructs circuits for any function f in $\mathcal{ZPP}^{\mathcal{NP},equiv(f)}$, where $equiv(f)$ denotes an *equivalence* oracle to f; that is, an oracle that gets as input a circuit $C : \{0,1\}^n \to \{0,1\}$, and outputs $x \in \{0,1\}^n$ such that $C(x) \neq f(x)$, if such x exists. Since the construction in [BCG+96] is simple and elegant, I review it in full in Appendix B.4.8; the underlying technical ideas are reminiscent of the ideas in the proof that was presented in Footnote 24.

The second algorithm is implicit in Fortnow *et al.* [FIK+08], and constructs circuits for any function f in $\mathcal{ZPP}^{\mathcal{NP}^f}$. (Note that this result is weaker than that of [BCG+96], since an equivalence oracle can be simulated by an $\mathcal{NP}^f$ oracle.) I won't provide full details of their construction, but for a short explanation on how to deduce the

[24] This statement can also be proved without going through [LY94]: Assuming that every circuit of size $O(s \cdot n)$ fails to compute f, we find a set S of $r = \mathrm{poly}(s)$ inputs such that every size-s circuit fails to compute f on at least one input in S. To do so, let $\mathcal{C}_0$ be the set of all size-s circuits. We construct S iteratively: In each iteration $i \in [r]$, we find an input x such that at least $1/4$ of the circuits in $\mathcal{C}_{i-1}$ fail to compute f at x; add x to S; and remove all the circuits that fail to compute f at x from $\mathcal{C}_{i-1}$ to obtain $\mathcal{C}_i$. The invariant in this process is that every size-s circuit that is not in $\mathcal{C}_i$ fails to compute f at some $x \in S$; therefore, after $r = \mathrm{poly}(s)$ iterations $\mathcal{C}_r$ will be empty, and hence S "fails" all size-s circuits. It is only left to see that in each iteration $i \in [r]$ we can find an appropriate x. To see this, note that by a probabilistic argument, there exists a set of $O(n)$ circuits from $\mathcal{C}_{i-1}$ such that on every input, the circuit C' that computes their majority agrees with at least $1/4$ of the circuits in $\mathcal{C}_{i-1}$. The point is that C' is of size $O(s \cdot n)$, so by our hypothesis, there exists an input on which C' fails to compute f.

algorithm for constructing circuits from their results see Appendix B.4.8. (In a nutshell, the problem of constructing circuits for f reduces to the problem of efficiently finding small near-optimal strategies in a *zero-sum game*; Fortnow *et al.* [FIK+08] showed a solution for the latter in $\mathcal{ZPP}^{\mathcal{NP}}$ that relies on techniques from learning theory, and this solution can be adapted to yield the construction mentioned above.)

I believe that a third algorithm can be derived from the work of Chakaravarthy and Roy [CR11]. Specifically, as above, we first reduce the problem of constructing circuits to a zero-sum game (again, see Appendix B.4.8 for details); and then, instead of using [FIK+08], we use the algorithm of [CR11] that finds small near-optimal strategies in a zero-sum game in $\mathcal{P}^{prAM}$ (rather than in $\mathcal{ZPP}^{\mathcal{NP}}$ as in [FIK+08]).

Deducing Karp-Lipton theorems. Building on the learning algorithm of [BCG+96], Watanabe [BCG+96, Thm. 24] deduced the following Karp-Lipton theorem: If $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, then $\mathcal{ZPP}^{\mathcal{NP}} = \mathcal{PH}$. Indeed, since we now know that $\mathcal{S}_2 \subseteq \mathcal{ZPP}^{\mathcal{NP}}$ (which wasn't known at the time), this result is superseded by the strengthening of the original Karp-Lipton theorem mentioned in Footnote 20 (i.e., $\mathcal{NP} \subset \mathcal{P}/\text{poly} \Rightarrow \mathcal{S}_2 = \mathcal{PH}$).

Nevertheless, let me sketch the proof. First, to solve $\mathcal{PH}$ in $\mathcal{ZPP}^{\mathcal{NP}}$ it suffices to construct a polynomial-sized circuit for SAT in $\mathcal{ZPP}^{\mathcal{NP}}$. To do so, we want to simulate the $\mathcal{ZPP}$ algorithm that uses an $\mathcal{NP}$ oracle and equivalence queries to SAT, where our simulation will run in $\mathcal{ZPP}^{\mathcal{NP}}$. The main challenge is that in $\mathcal{ZPP}^{\mathcal{NP}}$ we do not actually have an equivalence oracle to SAT, but only an $\mathcal{NP}$ oracle. However, since $\mathcal{NP} \subset \mathcal{P}/\text{poly}$, we can simulate the equivalence oracle using the $\mathcal{NP}$ oracle, and thus the entire algorithm runs in $\mathcal{ZPP}^{\mathcal{NP}}$.[25]

The exact same proof approach can be used to deduce that if there exists an algorithm that constructs circuits for any function f in $\mathcal{P}^{\mathcal{NP},equiv(f)}$ (i.e., by a polynomial-time algorithm that makes queries to $\mathcal{NP}$ and equivalence queries to f), then we have that $\mathcal{NP} \subset \mathcal{P}/\text{poly} \Rightarrow \mathcal{P}^{\mathcal{NP}} = \mathcal{PH}$. By a win-win analysis as in the end of Section B.4.1, the latter theorem would imply that $\mathcal{P}^{\mathcal{NP}} \not\subset SIZE[n^k]$ for every fixed $k \in \mathbb{N}$. (And in fact, as alluded to in Section B.4.1, the former Karp-Lipton theorem is essentially necessary to prove the latter lower bound; see [CMM+19].)

B.4.4.2 An "unreasonable" learning algorithm: The idea of [IW98]

Impagliazzo and Wigderson [IW98] considered not only a collapse hypothesis, but also an *additional* hypothesis, which allowed them to deduce – conditionally – that there exists a learning algorithm for their target function. Their algorithm of was originally presented as constructing a circuit for the permanent function, but was later on shown to work in more general settings (see [TV07; CNS99]); I present the general setting.

[25] Specifically, for any $C \in \mathcal{C}$, to construct a counter-example where $x \in L$ and $C(x) = 0$ consider the set $A = \{(\langle C \rangle, 1^n, \sigma)\}$ such that σ is a prefix of an n-bit input that can be extended to $x \in \{0,1\}^n$ satisfying $C(x) = 0 \wedge x \in L$. Since $A \in \mathcal{NP}$, we can construct a counter-example bit-by-bit using the $\mathcal{NP}$ oracle. To construct a counter-example where $x \notin L$ and $C(x) = 1$, consider the set $B = \{\langle C \rangle, 1^n, \sigma\}$ such that σ cannot be extended to $x \in \{0,1\}^n$ satisfying $C(x) = 1 \wedge x \in L$, and note that $B \in co\mathcal{NP}$ (so we can again construct a counter-example using the $\mathcal{NP}$ oracle).

Theorem B.4.8 ([IW98]). *Let $f : \{0,1\}^* \to \{0,1\}$ be a function that is randomly self-reducible and downwards self-reducible, and assume that:*

1. *There exists a polynomial-sized circuit family that computes f.*
2. *For every $\epsilon > 0$, let NW^f be the Nisan-Wigderson PRG, instantiated for stretch $n^\epsilon \mapsto n$ with the function f. We assume that there exists a polynomial-time algorithm D that for all $n \in \mathbb{N}$ satisfies $\left| \Pr_{s \in \{0,1\}^{n^\epsilon}}[D(NW^f(s)) = 1] - \Pr[D(U_n)] = 1 \right| > 1/n$.*

Then, there exists a probabilistic polynomial-time algorithm that on input 1^n constructs a circuit $C : \{0,1\}^n \to \{0,1\}$ that computes f on $\{0,1\}^n$.

I will not survey all the details of their proof, as those are explained well in standard textbooks, e.g. in [AB09]. But I do want to discuss their ideas in high level, to understand what allows them to solve the problem. Their construction has three parts:

1. They use the additional hypothesis (regarding the NW generator) to deduce that *there exists a learning algorithm* for f (recall that by "learning" I mean learning with membership queries over the uniform distribution). Specifically, a crucial observation in their proof is that any "distinguisher" algorithm D for the NW generator can be efficiently transformed into a *learning* algorithm for the "hard" function with which the NW generator is instantiated (i.e., f in this case).

2. Now, to use the learning algorithm we still need to find a way to answer the membership queries that the algorithm issues. However, since f is downwards self-reducible, they use a "bootstrapping" argument: Specifically, they use the learning algorithm to construct circuits for f with i-bit inputs, for $i = 1, ..., n$. Whenever invoking the learning algorithm for i-bit inputs, we answer its queries using the circuit that we already have for inputs with $i - 1$ bits.

3. At this point we have a circuit that computes the function f correctly on *most inputs* (since we used a learning algorithm, whose guarantee is to yield a circuit that is correct on most inputs). Indeed, since f is randomly self-reducible, we can now modify the circuit to a probabilistic circuit that on each input computes f correctly, with high probability. Finally, by modifying the circuit to also implement standard error-reduction, we construct a probabilistic circuit that, with high probability, correctly computes f on every input; so we can now conclude by randomly choosing random coins for the probabilistic circuit and hard-wiring them (to obtain a deterministic circuit).

Recapping, we used the additional hypothesis (referring to the Nisan-Wigderson generator) to obtain a learning algorithm for f; we used the downwards self-reducibility of f to overcome the challenge of answering the queries of the learning algorithm; and we used random self-reducibility to transform the circuit into a probabilistic circuit that is correct on every input (whp). Instantiating this construction for specific settings, we get the following Karp-Lipton theorems:

Theorem B.4.9 ([IW98; CNS99; TV07]; informal)**.** *We say that* there exists a uniform distinguisher for the NW-PRG with a function f *if for every $\epsilon > 0$ there exists a polynomial-time algorithm D such that for all $n \in \mathbb{N}$ it holds that $\left| \Pr_{s \in \{0,1\}^{n^\epsilon}}[D(NW^f(s)) = 1] - \Pr[D(U_n)] = 1 \right| > 1/n$. Then,*

1. *If $\#\mathcal{P} \subset \mathcal{P}/\text{poly}$ and there exists a uniform distinguisher for the NW-PRG with the permanent, then $\mathcal{BPP} = \mathcal{P}^{\#\mathcal{P}}$.*
2. *If $\mathcal{PSPACE} \subset \mathcal{P}/\text{poly}$ and there exists a uniform distinguisher for the NW-PRG with the $\mathcal{PSPACE}$-complete set of Trevisan and Vadhan [TV07], then $\mathcal{BPP} = \mathcal{PSPACE}$.*
3. *If $\mathcal{EXP} \subset \mathcal{P}/\text{poly}$ and there exists a uniform distinguisher for the NW-PRG with the permanent, then $\mathcal{BPP} = \mathcal{EXP}$.*[26]

Let me also mention that Oliveira and Santhanam noted [OS17, Sec. 6.1] that this proof can also be adapted to work under the weaker hypothesis that there exists a distinguisher for the NW-PRG with f that uses a *small amount of non-uniform advice*. Loosely speaking, in the "bootstrapping" step, for each input length $i = 1, ..., n$, when using the learning algorithm to construct a circuit over $\{0,1\}^i$, we can try out all potential advice strings for the learning algorithm, and *efficiently test* which advice string yielded the best-performing circuit (using downwards self-reducibility). Indeed, the running time of this algorithm grows exponentially with the length of the advice.

B.4.5 Adding a derandomization hypothesis: Derandomization implies lower bounds

How can we "push forward" the unlikely conclusions of Karp-Lipton theorems in order to obtain a contradiction? Recall that the conclusions in the theorems that we surveyed were that a uniform algorithm was unexpectedly strong. Also recall that in almost all of our solutions, the uniform algorithm crucially uses randomness.

The main observation underlying this section is that if, in addition to the collapse hypothesis, certain *derandomization* hypotheses also hold, then we can transform the uniform probabilistic algorithm into a *deterministic algorithm*. Now, since a deterministic construction is a stronger conclusion than a probabilistic construction, this brings us "one inch closer" to a contradiction; in fact, in some settings (which will be detailed below) this already suffices to get a contradiction! The resulting theorems in such cases assert that derandomization implies circuit lower bounds (since the derandomization hypothesis and the collapse hypothesis cannot be simultaneously true).

Let us begin with a well-known example of such a case. Recall the result of [IKW02] that if $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$ then $\mathcal{MA} = \mathcal{NEXP}$. Now, let us consider both the collapse

[26] In fact, in Items (1) and (2) one can forego the collapse hypothesis, since this hypothesis follows from the hypothesis that there exists a uniform distinguisher for the NW-PRG with a function that is complete for the corresponding class. In contrast, in Item (3) we use the collapse hypothesis to deduce that the permanent is complete (under this hypothesis) for $\mathcal{EXP}$.

hypothesis $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$ and a derandomization hypothesis, say, that $pr\mathcal{BPP} = pr\mathcal{P}$ (which implies that $\mathcal{MA} = \mathcal{NP}$). Under the joint hypotheses, we get that $\mathcal{NP} = \mathcal{MA} = \mathcal{NEXP}$, which contradicts the non-deterministic time hierarchy. In fact, even a much weaker derandomization hypothesis would still yield a contradiction; that is, any separation of $\mathcal{NEXP}$ from $\mathcal{MA}$ (which we think of as derandomization of $\mathcal{MA}$) would yield a contradiction to the hypothesis that $\mathcal{NEXP} \subset \mathcal{P}/\text{poly}$.

This basic approach was taken one step further by Williams [Wil13]. Recall that the underlying Karp-Lipton theorem in his work (i.e., Theorem B.4.5) asserts that if $\mathcal{NEXP} \subset \mathcal{C}$, then $\mathcal{NTIME}[2^n]$ can be decided by an $\mathcal{MA}$ verifier that runs in polynomial time (i.e., in time $\text{poly}(n)$), uses $n + O(\log(n))$ bits of randomness, and whose residual decision as a function of its random coins can be computed by a $\mathcal{C}$-circuit. Now, assume that there exists an algorithm that gets as input $C \in \mathcal{C}$ over $n' = n + O(\log(n))$ input bits, runs in time $2^{n'}/(n')^{\omega(1)} = 2^n/n^{\omega(1)}$, and distinguishes between the case that C accepts at least $2/3$ of its inputs and the case that C rejects at least $2/3$ of its inputs (i.e., a "non-trivial" derandomization algorithm for $\mathcal{C}$). Then, we can use this algorithm to derandomize the $\mathcal{MA}$ verifier, and deduce that $\mathcal{NTIME}[2^n] \subseteq \mathcal{NTIME}[2^n/n^{\omega(1)}]$, a contradiction to the non-deterministic time hierarchy. Thus, "non-trivial" derandomization of a circuit class $\mathcal{C}$ implies lower bounds for $\mathcal{C}$.

Finally, recall that Kabanets and Impagliazzo [KI04] proved that if the permanent function can be computed by polynomial-sized arithmetic circuits, then $\mathcal{P}^{\#\mathcal{P}}$ can be decided by an $\mathcal{MA}$ verifier that gets an arithmetic circuit as proof, verifies in probabilistic polynomial time that the circuit computes the permanent, and then uses the circuit to solve $\mathcal{P}^{\#\mathcal{P}}$. Let us now add to the collapse hypothesis a derandomization hypothesis, namely that $\mathcal{P} = \mathcal{BPP}$ (the hypothesis $co\mathcal{RP} \subseteq \mathcal{NP}$ also suffices for our purposes). Under this hypothesis, the probabilistic verification of the circuit can be replaced with a deterministic verification procedure (see [KI04, Lem. 9 & 12]). Therefore, we can decide all of $\mathcal{P}^{\#\mathcal{P}}$ in $\mathcal{NP}$, which means that $\mathcal{PH} \subseteq \mathcal{P}^{\#\mathcal{P}} = \mathcal{NP}$ (the containment is by Toda's theorem [Tod91]), in which case (by a padding argument) $\mathcal{EH} = \mathcal{NE}$. Since $\mathcal{EH}$ contains functions with maximal circuit complexity $\Omega(2^n/n)$ (by an elementary diagonalization argument), we have that $\mathcal{NE}$ contains functions with maximal circuit complexity. Thus, we get the main result of [KI04], as improved by [KMS12]: If $\mathcal{P} = \mathcal{BPP}$, then either the permanent does not have polynomial-sized arithmetic circuits, or $\mathcal{NE}$ contains functions with maximal circuit complexity $\Omega(2^n/n)$.

B.4.6 Variations on the collapse hypothesis

Collapse to non-uniform circuits. Several works consider a collapse hypothesis in which a large uniform class has small *non-deterministic* circuits (i.e., circuits that also use non-determinism). Specifically, Yap **[yap83]** generalized the original Karp-Lipton theorem by proving that for all $i \geq 0$ it holds that $\Sigma_{i+1} \subset \Pi_i/\text{poly} \Rightarrow \Sigma_{i+2} = \Pi_{i+2}$; indeed, the original theorem is obtained when $i = 0$. (For a strengthening of Yap's theorem see **[ccho05]**.) Also, Aydınlıoğlu and van Melkebeek **[avm12]** showed that, under

suitable hypotheses referring to derandomization of $\mathcal{AM}$, if $\mathcal{PSPACE} \subset \mathcal{NP}/\text{poly}$ then $\Sigma_2 = \mathcal{PSPACE}$; and similarly, under the same derandomization hypothesis, if $co\mathcal{NP} \subset \mathcal{NP}/\text{poly}$ then $\mathcal{P}^{\Sigma_2} = \mathcal{PH}$.

Collapse to quantum circuits. Aaronson [Aar06] considered a collapse hypothesis in which $\mathcal{PP}$ has small *quantum* circuits. Quoting his text (since that is the best I can provide...), he proved that if $\mathcal{PP}$ has polynomial-sized quantum circuits, then the counting hierarchy (which consists of $\mathcal{PP}$, $\mathcal{PP}^{\mathcal{PP}}$, $\mathcal{PP}^{\mathcal{PP}^{\mathcal{PP}}}$ etc.) collapses to $\mathcal{QMA}$.

B.4.7 Addendum: Proof idea for the easy witness lemma of [MW18]

In this appendix I'll sketch the proof idea for the easy witness lemma of [MW18]. For simplicity I'll focus on the case of general circuits (rather than restricted circuit classes), and will not be precise about the parameter values. Then, the lemma asserts the following: For "well-behaved" functions $\hat{t} \gg t \gg \hat{s} \gg s$, if $\mathcal{NTIME}[\hat{t}] \subset SIZE[s]$, then for every set $L \in \mathcal{NTIME}[t]$, and every time-t verifier, and every $x \in L$, there exists a witness for x that can be concisely represented by a circuit of size $\hat{s}$.

The main idea in the proof dates back to a sequence of works by Kabanets and Cai [**kc00**], Kabanets [Kab01], and Impagliazzo, Kabanets, and Wigderson [IKW02]. The proof approach of [MW18], which I'll present, is more elegant than the original one in [IKW02], but is nevertheless based on the same main idea. The starting point of the proof is the fact that *derandomization implies circuit lower bounds*. Specifically, as a corollary of (a generalization of) a result of Santhanam [San09], if $\mathcal{MATIME}[\hat{s}] \subseteq \mathcal{NTIME}[\hat{t}]/n$ then $\mathcal{NTIME}[\hat{t}] \not\subset SIZE[s]$. [27] That is, if $\mathcal{MATIME}[\hat{s}]$ can be derandomized in time $\hat{t}$ and with a linear amount of non-uniform advice, then we have the circuit lower bound $\mathcal{NTIME}[\hat{t}] \not\subset SIZE[s]$.

Loosely speaking, the main observation in the proof is that if the conclusion of the lemma does *not* hold, then we can construct a pseudorandom generator that will allows us to derandomize $\mathcal{MATIME}[\hat{s}]$ in $\mathcal{NTIME}[\hat{t}]/n$. Thus, relying on the result above, if the conclusion of the easy witness lemma does *not* hold, then we can deduce that $\mathcal{NTIME}[\hat{t}] \not\subset SIZE[s]$. The easy witness lemma is the contrapositive statement: Assuming that $\mathcal{NTIME}[\hat{t}] \subset SIZE[s]$, we deduce that the conclusion of the lemma holds.

Let me now briefly explain how to prove the foregoing main observation. If the conclusion of the easy witness lemma does not hold, then there *exists* $L \in \mathcal{NTIME}[t]$, a t-time verifier V for L, and an infinite sequence of inputs $\{x_n \in \{0,1\}^n\}_{n \in S \subseteq \mathbb{N}}$ such that *every* witness $w \in \{0,1\}^t$ for x that V accepts *cannot* be concisely represented by circuits of size $\hat{s}$. In other words, every such $w \in \{0,1\}^t$ is the truth-table of a function

[27] The generalization of Santhanam's result asserts that for $\hat{s} \gg s$ it holds that $\mathcal{MATIME}[\hat{s}]/1 \not\subset SIZE[s]$. Therefore, if $\mathcal{MATIME}[\hat{s}] \subseteq \mathcal{NTIME}[\hat{t}]/n$, then $\mathcal{NTIME}[\hat{t}]/(n+1) \not\subset SIZE[s]$. At this point we can use an advice elimination trick to deduce that $\mathcal{NTIME}[\hat{t}] \not\subset SIZE[s']$, where s' is mildly smaller than s (in this overview I ignore the difference between s' and s). (For more specific details see Theorem 4.2.20.)

with circuit complexity more than $\hat{s}$. Intuitively, we will use this hypothesis to guess-and-verify a truth-table of a function w with circuit complexity more than $\hat{s}$, and then use w for derandomization of the verifier in the $\mathcal{MATIME}[\hat{s}]$ protocol.

More specifically, given the above, for infinitely-many input lengths n, given $x_n \in \{0,1\}^n$ as non-uniform advice, we can non-deterministically guess a witness $w \in \{0,1\}^t$ for x_n in L, verify that $V(x,w) = 1$, and if that is indeed the case, we now *know* that w has circuit complexity at least $\hat{s}$. Now, using standard "hardness-randomness" results, given such a function we can derandomize $\mathcal{MATIME}[\hat{s}]$ in $\mathcal{NTIME}[\hat{t}]$, where $\hat{t} = \text{poly}(t)$. (The specific "hardness-randomness" result that we use is the one of Umans [Uma03], following the sequence of works initiated by Nisan [Nis91] and Nisan and Wigderson [NW94].) Thus, we can solve $\mathcal{MATIME}[\hat{s}]$ infinitely-often in $\mathcal{NTIME}[\hat{t}]/n$, by getting x_n as advice, guessing and verifying a "hard" function w, and using w to construct a pseudorandom generator for the MA verifier.

There is one last detail missing in the above proof sketch: We showed that $\mathcal{MATIME}[t]$ can be simulated in $\mathcal{NTIME}[\hat{t}]/n$ only *infinitely-often*, whereas the hypothesis of the "derandomization $\Rightarrow$ lower bounds" argument that relies on [San09] needs the hypothesis that $\mathcal{MATIME}[t]$ can be simulated in $\mathcal{NTIME}[\hat{t}]/n$ on *all* input lengths. Bridging this gap is the main technical contribution of [MW18], and it is achieved by both improving the result from [San09] and slightly refining the proof idea above; see [MW18, Sec. 3] for details.

B.4.8 Addendum: The algorithms of [BCG+96] and [FIK+08]

In this appendix we give some more details for the algorithms of Bshouty *et al.* [BCG+96] and of Fortnow *et al.* [FIK+08], which were mentioned in Section B.4.4.1. Let us first shortly review the construction of Bshouty *et al.* [BCG+96]:

Theorem B.4.10 ([BCG+96])**.** *Let $f : \{0,1\}^* \to \{0,1\}$ be any function. Then, there exists a probabilistic polynomial-time algorithm that gets as input $(1^n, 1^s)$, makes oracle queries to $\mathcal{NP}$ and to an equivalence oracle for f,*[28]*, and satisfies the following:*

- *If there exists a size-s circuit that computes f_n (i.e., f on $\{0,1\}^n$), then with high probability the algorithm outputs a circuit of size $O(s \cdot n)$ for f_n. Moreover, the algorithm never outputs an incorrect circuit, but might output "fail" with low probability.*
- *If every circuit of size $O(s \cdot n)$ fails to compute f_n, then the algorithm outputs "fail" with probability one.*

The proof below is similar to the proof that was presented in Footnote 24. However, a crucial difference is that in the latter proof the set of "bad" inputs is constructed non-efficiently, whereas in the proof below this set is constructed by an efficient algorithm (with an $\mathcal{NP}$ oracle).

[28] Specifically, it issues queries of the form "given a circuit $C : \{0,1\}^n \to \{0,1\}$ of size $O(s \cdot n)$, find $x \in \{0,1\}^n$ such that $C(x) \neq f(x)$, or output 'fail' if not such input exists".

Proof Sketch for Theorem B.4.10. The algorithm works in $r = \text{poly}(s)$ iterations, where in each iteration it "tests" a different candidate circuit C of size $O(s \cdot n)$ for f_n, using the equivalence oracle. Note that if f_n cannot be computed by circuits of size $O(s \cdot n)$, then the algorithm cannot find a candidate circuit C that correctly computes f_n, and will thus always output "fail". On the other hand, we will show that if f_n can be computed by a size-s circuit, then with high probability one of the r candidate circuits will correctly compute C. Moreover, since the algorithm verifies each candidate circuit using the equivalence oracle, the algorithm never outputs an incorrect circuit.

Let us now describe the iterative process. Initialize $S_0 \subseteq \{0,1\}^n$ as an empty set of inputs, and let $\mathcal{C}_0$ be the set of all size-s circuits. In each iteration $i \in [r]$, the algorithm will send a candidate circuit C to the equivalence oracle, and if the equivalence oracle returns x such that $C(x) \neq f_n(x)$, then the algorithm will add x to S_{i-1} to obtain a new set S_i; the set $\mathcal{C}_i \subseteq \mathcal{C}_{i-1}$ will then be defined as the set of size-s circuits that agree with f on S_i. We will show that with high probability, in each iteration i where the algorithm adds an input x to S_{i-1}, the set $\mathcal{C}_i$ "shrinks" by a multiplicative constant factor; that is $|\mathcal{C}_i| = \frac{3}{4} \cdot |\mathcal{C}_{i-1}|$. Thus, with high probability, if the process continues for $r = \text{poly}(s)$ iterations, then we have that $\mathcal{C}_r = \emptyset$, which implies that no size-s circuit can compute f_n (since every such circuit disagrees with f on S_r). Taking the contrapositive of the latter statement, if there exists a size-s circuit for f_n, then with high probability the algorithm will find a circuit C that computes f_n after at most r iterations.

Being more specific, in each iteration $i \in [r]$, the algorithm uniformly samples a subset of $O(n)$ circuits $C_1, ..., C_{O(n)} \in \mathcal{C}_{i-1}$, and considers the function $C(x) = MAJ(C_1(x), ..., C_{O(n)}(x)$. Observe that, with high probability, for every $x \in \{0,1\}^n$ it holds that $C(x)$ agrees with at least $1/4$ of the circuits in $\mathcal{C}_{i-1}$. Also note that C can be implemented as a circuit of size $O(s \cdot n)$. Now, the algorithm considers C to be its candidate circuit, and sends C as a query to the equivalence oracle. If $C \equiv f_n$ then the algorithm successfully outputs C; and otherwise, the algorithm gets an input x such that $1/4$ of the circuits in $\mathcal{C}_{i-1}$ disagree with f_n at x. In this case the algorithm adds x to S_{i-1}, which implies that $|\mathcal{C}_i| \leq \frac{3}{4} \cdot |\mathcal{C}_{i-1}|$.

The only challenge in implementing the above procedure is that in each iteration $i \in [r]$ we need to uniformly sample $O(n)$ circuits from $\mathcal{C}_{i-1}$, where the latter is a set of size up to $2^{\text{poly}(s)}$, which is only defined "implicitly" by the constraints imposed by S_{i-1}. Fortunately, this "sampling from a structured set" task can be solved in probabilistic time $\text{poly}(s)$ with an $\mathcal{NP}$ oracle, relying on the results of Jerrum, Valiant, and Vazirani [JVV86].[29] Thus, we use the $\mathcal{NP}$ oracle to solve the problem of sampling from $\mathcal{C}_{i-1}$ in each step, and we use the equivalence oracle to test each candidate circuit (and update S_{i-1}) in each step. ■

Turning to the construction that is implicit in Fortnow *et al.* [FIK+08], let me first

[29] Actually, the work of [JVV86] reduces *approximately-uniform sampling* to *approximate counting*, where the latter can be solved in time $\text{poly}(s)$ with an $\mathcal{NP}$ oracle. However, the algorithm of [BCG+96] works well even if the samples $C_1, ..., C_{O(n)} \in \mathcal{C}_{i-1}$ are approximately uniform, rather than uniform (see [BCG+96] for details).

mention that (as mentioned in Section B.4.4.1) their algorithm relies on techniques from learning theory. I won't review the construction itself, but will explain how to derive the learning algorithm from their construction:

Theorem B.4.11 (implicit in [FIK+08]). *Assume that $f : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit of size at most s. Then, there exists a probabilistic algorithm that runs in time $\mathrm{poly}(s)$, makes oracle queries to $\mathcal{NP}^f$, and with high probability produces a circuit of size $\mathrm{poly}(s)$ that computes f. Moreover, the algorithm never outputs an incorrect circuit (but might fail and halt, with low probability).*

Sketch of how to derive Theorem B.4.11 from [FIK+08]. Consider a matrix whose rows are indexed by all size-s circuits and whose columns are $\{0,1\}^n$. The $(\langle C \rangle, x)$ entry in the matrix, where C is a size-s circuit and $x \in \{0,1\}^n$, is zero if $C(x) = f(x)$ and one if $C(x) \neq f(x)$. This matrix describes a zero-sum game (for a definition see, e.g., [FIK+08, Sec. 1.1]) that has value zero, since there is a circuit C that correctly computes f on all inputs in $\{0,1\}^n$ (i.e., an all-zero row).

Now, Fortnow *et al.* [FIK+08, Thm. 10] constructed a $\mathcal{ZPP}^{\mathcal{NP}}$ algorithm that, given as input a circuit $G : \{0,1\}^{\mathrm{poly}(s)} \times \{0,1\}^n \to \{0,1\}$ such that $G(\langle C \rangle, x)$ is the $(\langle C \rangle, x)$ entry in the matrix (i.e., G is a concise description of the zero-sum game), outputs a set of $r = \mathrm{poly}(s)$ row indices $i_1, ..., i_r$ such that for every column j it holds that $\mathbb{E}_{k \in [r]} G(i_k, j) < 1/4$. [30] Going through their proof, their algorithm works also when G is not given explicitly, but only available as black-box to the $\mathcal{NP}$ oracle; that is, their algorithm works when the input is $(1^n, 1^s)$ and the oracle is $\mathcal{NP}^G$. In other words, their algorithm construction is in fact in $\mathcal{ZPP}^{\mathcal{NP}^G}$.

Now, in our setting the rows are circuits and the columns are inputs, and so their algorithm outputs a set of $\mathrm{poly}(s)$ circuits such that the majority of these circuits is correct on any input. The algorithm can therefore output a circuit that computes the majority of this set of $\mathrm{poly}(s)$ circuits. And finally, we can easily compute G at any point given oracle access to f, and so the algorithm is in $\mathcal{ZPP}^{\mathcal{NP}^f}$. ∎

B.5 On implications of better sub-exponential lower bounds for $\mathcal{AC}^0$

Abstract. Lower bounds for $\mathcal{AC}^0$ circuits of sub-exponential size have been known since the '80s. The point of this text is to highlight the (known) fact that lower bounds for $\mathcal{AC}^0$ circuits of significantly larger sub-exponential size imply lower bounds for polynomial-sized circuits from circuit classes such as $\mathcal{AC}^0[\mathtt{Sym}]$, linear threshold circuits of constant depth, and $\mathcal{NC}^1$.

Comment: The contents of this text does not contain technical innovation, and may be known to some experts; for example, related results of similar spirit were discussed

[30] In the statement of [FIK+08, Thm. 10] they bound the number of rows by $\mathrm{poly}(|G|)$, but their proof actually guarantees a bound of $\mathrm{poly}\log(m)$ rows, where m is the total number of rows in the matrix.

in [Vio17; Vio19]. However, given that some researchers are not familiar with the facts presented here, it seems beneficial to publicly share them. Needless to say, it was also helpful for me to flesh out the details while writing the text.

B.5.1 Context and main results

Recall that the best currently-known lower bounds for $\mathcal{AC}^0$ circuits against explicit functions[31] assert that circuits of depth $d \geq 2$ require size $2^{\Omega(n^{1/(d-1)})}$ to compute the parity function (see [Hås87], following [FSS84; Ajt83; Yao85]). It is well-known that lower bounds for much larger $\mathcal{AC}^0$ circuits – that is, $\mathcal{AC}^0$ circuits of almost-exponential size $2^{\omega(n/\log\log(n))}$, even just of depth three – would imply lower bounds for linear-sized circuits of depth $O(\log(n))$, which are currently unknown (see [Val77]).

The purpose of this text is to highlight the fact that even before considering almost-exponential lower bounds of the form $2^{n/\log\log(n)}$, "just" proving *significantly better sub-exponential lower bounds* for $\mathcal{AC}^0$ would already yield lower bounds for *polynomial-size* circuits from classes such as $\mathcal{AC}^0[\mathtt{Sym}]$, constant-depth $\mathtt{LTF}$ circuits, or $\mathcal{NC}^1$.

Below I state three such implications, where in all cases I will concretely define the relevant class before spelling out the relevant parameters. The first lower bound refers to $\mathcal{AC}^0[\mathtt{Sym}]$ circuits of fixed polynomial size; the second lower bound refers to $\mathtt{LTF}$ circuits of fixed polynomial size; and the third lower bound refers to $\mathcal{NC}^1$ circuits of fixed logarithmic depth (and hence also fixed polynomial size). The proofs of these will appear in Section B.5.2, and amount to the same (well-known) observation: That $\mathcal{AC}^0$ circuits of large sub-exponential size can simulate polynomial-size circuits from the relevant circuit classes (i.e., $\mathcal{AC}^0[\mathtt{Sym}]$, $\mathtt{LTF}$ circuits, or $\mathcal{NC}^1$).

In Section B.5.3 I desrcibe a similar phenomenon in a different context: Improving our lower bounds for $\mathtt{LTF}$ circuits (rather than for $\mathcal{AC}^0$), even very mildly, would imply new lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$ circuits. (For details see Section B.5.3.)

Implication 1: Lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$. Recall that $\mathcal{AC}^0[\mathtt{Sym}]$ is the class of constant-depth circuits whose gates can compute arbitrary symmetric functions, and that we currently do not know a lower bound for $\mathcal{AC}^0[\mathtt{Sym}]$ circuits of depth two and size n^2 against an explicit function. (The best currently-known lower bound is for circuits of size $\tilde{\Omega}(n^2)$; see [Tam16].) The following result asserts that such lower bounds would follow from lower bounds for $\mathcal{AC}^0$ circuits of depth d and size $\approx 2^{n^{3/(d-1)}}$:

Theorem B.5.1 ($\mathcal{AC}^0[\mathtt{Sym}]$ lower bounds would follow from better $\mathcal{AC}^0$ lower bounds)**.** *Lower bounds for $\mathcal{AC}^0$ circuits of (sufficiently large) constant depth d and size $2^{\Omega(n^{3/(d-1)}\cdot\log(n))}$*

[31] When referring to circuits of arbitrarily large polynomial size, let us say that a function is explicit if it can be computed in $\mathcal{EXP}^{\mathcal{NP}}$. The reason for this choice is that if we go "higher up" in time or in alterations, then lower bounds are already known (i.e., we can diagonalize against $\mathcal{P}/\mathrm{poly}$ in time $2^{n^{\omega(1)}}$, and $\Sigma_2[n^{\omega(1)}]$ is hard for $\mathcal{P}/\mathrm{poly}$ by Kannan's theorem [Kan82, Thm. 4]). When referring to circuits of size n^k for a fixed k, an explicit function is one that is computable in $\mathcal{E}^{\mathcal{NP}}$ or in $\mathcal{DTIME}[2^{n^k}]^{\mathcal{NP}}$, for the same reason (i.e., both $\mathcal{EXP}$ and $\Sigma_2 = \cup_c \Sigma_2[n^c]$ are hard for such circuits, using the same arguments).

would imply lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$ circuits of depth two and size n^2. The latter lower bounds are also implied by lower bounds for $\mathcal{AC}^0$ circuits of depth 6 and size $2^{\Omega(n^{2/3}\cdot\log(n))}$.

Theorem B.5.1 extends to more general parameters, asserting that lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$ circuits of depth d_0 and size n^k would follow from lower bounds for $\mathcal{AC}^0$ circuits of large constant depth d and size $2^{\tilde{\Omega}(n^{\Delta/(d-1)})}$, where $\Delta \approx k \cdot d_0$ (see Theorem B.5.6).

Note that there is a *gap* between the known lower bounds for $\mathcal{AC}^0$, which are of the form $2^{\Omega(n^{1/(d-1)})}$ and hold for every $d \geq 2$, and the lower bounds that would imply new lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$ circuits, which are of the form $2^{\tilde{\Omega}(n^{3/(d-1)})}$ and should hold for large constant depths $d \geq 6$. Thus, one could improve the known $\mathcal{AC}^0$ lower bounds without proving new lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$ circuits (e.g., by considering circuits of depth $d < 6$, or of size $2^{n^{2/(d-1)}}$).

Implication 2: Lower bounds for $\mathcal{TC}^0$ and for $\mathtt{LTF}$ circuits. Theorem B.5.6 implies in particular that lower bounds for *subclasses* of $\mathcal{AC}^0[\mathtt{Sym}]$ would follow from significantly better sub-exponential lower bounds for $\mathcal{AC}^0$. This specifically applies to $\mathcal{TC}^0$, which is the class of constant-depth circuits with unweighted majority gates.

Extending this fact, consider the class of constant-depth circuits whose gates can compute arbitrary linear threshold functions (i.e., $\mathtt{LTF}$s) rather than only unweighted majorities; the best currently-known lower bound for $\mathtt{LTF}$ circuits is for circuits with $n^{1+c^{-d}}$ wires, where $c \approx 2.41$ (see [IPS97], and also [CSS16; CT19]). While $\mathtt{LTF}$ gates are not symmetric, it is well-known that polynomial-sized $\mathtt{LTF}$ circuits can be simulated by $\mathcal{AC}^0[\mathtt{Sym}]$ circuits (in fact by circuits with unweighted majority gates) of larger polynomial size. Thus, the class of $\mathtt{LTF}$ circuits of fixed polynomial size n^k is a subclass of $\mathcal{AC}^0[\mathtt{Sym}]$ of larger fixed polynomial size $n^{k'}$, which means that lower bounds for $\mathcal{AC}^0$ circuits of large sub-exponential size would imply lower bounds for $\mathtt{LTF}$ circuits of polynomial size. (See Corollary B.5.7 for precise details.)

Implication 3: Lower bounds for $\mathcal{NC}^1$. Consider the class $\mathcal{NC}^1$ of circuits with fan-in two and depth $O(\log(n))$ (the size of such circuits is trivially bounded by $\mathrm{poly}(n)$).[32] As far as I am aware, we currently do not know of lower bounds for this class even when the depth is only $2 \cdot \log(n)$. The following result asserts that such lower bounds would follow from lower bounds for $\mathcal{AC}^0$ circuits of depth d and size $\approx 2^{n^{4/(d-1)}}$:

Theorem B.5.2 ($\mathcal{NC}^1$ lower bounds would follow from better $\mathcal{AC}^0$ lower bounds)**.** *Lower bounds for $\mathcal{AC}^0$ circuits of constant depth $d \geq 6$ and size $2^{\Omega(n^{4/(d-1)})}$ would imply lower bounds against circuits with fan-in two and depth $2 \cdot \log(n)$.*

[32] An alternative definition, ignoring polynomial overheads in the size, would be the class of formulas of depth $O(\log(n))$. Recall that circuits of depth $d = O(\log(n))$ can be simulated by formulas of depth d, which are in turn of size at most $2^d = \mathrm{poly}(n)$. However, this simulation potentially incurs a polynomial overhead in the size, since the size of the initial circuit might have been smaller than 2^d.

For a statement of Theorem B.5.2 with more general paramters, see Theorem B.5.4. As in Theorem B.5.1, there is a gap between the currently-known $\mathcal{AC}^0$ lower bounds, which are of the form $2^{n^{1/(d-1)}}$ and hold for any $d \geq 2$, and the lower bounds that would imply new $\mathcal{NC}^1$ lower bounds, which are of the form $2^{n^{4/(d-1)}}$ and hold only for $d \geq 6$. Also note that the gap in Theorem B.5.1 is smaller than the gap in Theorem B.5.2.

On optimistic and pessimistic interpretations. As usual in cases of implications between well-known open problems, an optimistic interpretation of these results is that improving our $\mathcal{AC}^0$ lower bounds may be an appealing way to prove better lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$, $\mathtt{LTF}$ circuits and $\mathcal{NC}^1$; whereas a pessimistic interpretation treats the latter lower bounds as "barriers" that we should focus on before addressing $\mathcal{AC}^0$.

One point to note in this context is that the proofs of the results above show that classes such as $\mathcal{AC}^0[\mathtt{Sym}]$ of fixed polynomial size, or $\mathcal{NC}^1$ of fixed logarithmic depth, are in fact *sub-classes* of $\mathcal{AC}^0$ of large sub-exponential size; that is, the proofs show that every problem decidable by the former is also decidable by the latter. Therefore, the results above actually *delineate several interesting special cases* of the problem of proving lower bounds for $\mathcal{AC}^0$ circuits of large sub-exponential size.

B.5.2 Detailed statements and proofs

The proofs of Theorems B.5.1 and B.5.2 amount to showing that polynomial-sized $\mathcal{AC}^0[\mathtt{Sym}]$ circuits and $\mathcal{NC}^1$ circuits can be simulated by sub-exponential sized $\mathcal{AC}^0$ circuits (and thus lower bounds for the latter imply lower bounds for the former). This simulation boils down to the well-known fact that any product of elements over a set with a binary associative operation (i.e., a semigroup) can be computed by an $\mathcal{AC}^0$ circuit of sub-exponential size. That is:

Proposition B.5.3 (solving the semigroup problem in $\mathcal{AC}^0$)**.** *For a semigroup Σ of size $\ell = \log(|\Sigma|)$ and $n \in \mathbb{N}$, let $f\colon \{0,1\}^{n\cdot\ell} \to \{0,1\}$ such that $f(x) = h\left(\prod_{i\in[n]} x_i\right)$ for some $h\colon \Sigma \to \{0,1\}$, where the x_i's are elements in Σ. Then, for any constant integer $d \geq 3$ it holds that f can be computed by a constant depth circuit of depth d and size $2^{O(n^{1/(d-1)}\cdot\ell)}$ with a bottom layer of OR gates, and by such a circuit with a bottom layer of AND gates.*

Proof. For any $m \in \mathbb{N}$, note that the function that takes as input m elements in Σ, represented in binary, and outputs their product, is computable by ℓ DNFs of size $2^{m\cdot\ell}$, and also by ℓ CNFs of such size.

Given $x \in \{0,1\}^{n\cdot\ell}$ we use a recursive construction to compute $f(x)$. For $i \in [d-2]$, in iteration i we start with a string of $n_i = n^{1-(i-1)/(d-1)}$ elements in Σ. We partition this string into $n_{i+1} = n^{1-i/(d-1)}$ substrings, each of length $m = n^{1/(d-1)}$, and compute the product of the elements in each substring using ℓ DNFs or CNFs of size $2^{n^{1/(d-1)}\cdot\ell}$. After completing $d-2$ iterations (i.e., when $i = d-1$), our string consists of a single block of m elements $r_1, \ldots, r_m \in \Sigma$, represented by $m \cdot \ell$ bits. We then directly compute $f(x) = h\left(\prod_{i\in[n]} x_i\right) = h\left(\prod_{i\in[m]} r_i\right)$ by a single depth-two formula of size $2^{m\cdot\ell}$.

If we alternate between using CNFs and using DNFs in each of the $d-2$ iterations, we can collapse one layer in each iteration after the first one. Thus, the mapping of x to the r_i's is computed in depth $d-1$ and size $2^{O(n^{1/(d-1)}\cdot\ell)}$. Using either a CNF or a DNF for the final computation of $f(\prod_i r_i)$ (to collapse a layer with the formula used in the $(d-2)^{th}$ iteration), the final circuit is of depth d and size $2^{O(n^{1/(d-1)}\cdot\ell)}$. ■

We now prove Theorem B.5.2, by simulating $\mathcal{NC}^1$ in sub-exponential $\mathcal{AC}^0$. To do so, we use Barrington's theorem [Bar89] to reduce any problem in $\mathcal{NC}^1$ to the problem of computing a product of elements over S_5 (i.e., over the symmetric group over five elements), and then use Proposition B.5.3 to solve the latter.

Theorem B.5.4 (parametrized version of Theorem B.5.2)**.** *For any $d_0\colon \mathbb{N}\to\mathbb{N}$ and any constant $d\geq 3$ the following holds. If a function $f\colon \{0,1\}^n\to\{0,1\}$ cannot be computed by n-bit $\mathcal{AC}^0$ circuits of depth d and size $2^{O(2^{2d_0(n)/(d-1)})}$, then f cannot be computed by circuits (with fan-in two) of depth $d_0(n)$. (Note that the size of such a circuit is at most $2^{d_0(n)}$.)*

Proof. Let $f\colon \{0,1\}^n\to\{0,1\}$ a function that is computable by a circuit of depth $d_0=d_0(n)$, and let S_5 be the symmetric group over five elements. For $n\in\mathbb{N}$ and $m=4^{d_0}$, we say that a mapping $\{0,1\}^n\to(S_5)^m$ is local if every output permutation depends on a single input bit. Recall that by Barrington's theorem [Bar89], there is a local mapping of any $x\in\{0,1\}^n$ to a sequence of m permutations in S_5 such that $f(x)=0$ if and only if the product of the m permutations is the identity element.[33]

Our $\mathcal{AC}^0$ circuit locally maps $x\in\{0,1\}^n$ to a sequence of m permutations in S_5, and then uses Proposition B.5.3 (instantiated with the monoid S_5 and with $h\colon S_5\to\{0,1\}$ such that $h(\sigma)=0 \iff \sigma=e$) to compute $f(x)$ in depth d and size $2^{O(m^{1/(d-1)})}$. By collapsing the first layer (that computes the local mapping) with the layer above it, we obtain a circuit of depth d and size $2^{O(m^{1/(d-1)})}=2^{O(2^{2d_0/(d-1)})}$. ■

Towards proving Theorem B.5.1, we first note that any symmetric function can be computed by an $\mathcal{AC}^0$ circuit of sub-exponential size; this fact follows as a special case of Proposition B.5.3:

[33] The locality of Barrington's reduction is well-known, but I didn't find an explicit proof of it, so let us verify now that it indeed holds. We map x to a sequence of permutations by iteratively constructing the sequence in d_0 layers, corresponding to the d_0 layers of the original circuit. Fix cyclic permutations $\sigma,\tau\in S_5$ such that $\sigma\tau\sigma^{-1}\tau^{-1}$ is cyclic. The bottom layer consists of at most 2^{d_0} permutations, where each permutation is of the form σ^{x_i} for some $i\in[n]$. We now construct the next layer, assuming that the mapping of x to the layer below us is local, and that each subsequence of permutations in the layer below us, which corresponds to a gate g, evaluates to $\sigma^{g(x)}$. For each AND gate g in the current layer, consider the two subsequences of permutations P_1,P_2 corresponding to the two gates feeding into g. For any two cyclic permutations $\rho_1,\rho_2\in S_5$, denote by $\mathtt{Cj}_{\rho_1\mapsto\rho_2}$ the element such that conjugating ρ_1 by $\mathtt{Cj}_{\rho_1\mapsto\rho_2}$ yields ρ_2 (i.e., $\mathtt{Cj}_{\rho_1\mapsto\rho_2}\rho_1\mathtt{Cj}^{-1}_{\rho_1\mapsto\rho_2}=\rho_2$). Let $P_2^{(\tau)}=\mathtt{Cj}_{\sigma\mapsto\tau}P_2\mathtt{Cj}^{-1}_{\sigma\mapsto\tau}$, let $P_1^{(\sigma^{-1})}=\mathtt{Cj}_{\sigma\mapsto\sigma^{-1}}P_1\mathtt{Cj}^{-1}_{\sigma\mapsto\sigma^{-1}}$, and let $P_2^{(\tau^{-1})}=\mathtt{Cj}_{\sigma\mapsto\tau^{-1}}P_2^{(\tau)}\mathtt{Cj}^{-1}_{\sigma\mapsto\tau^{-1}}$. Then, we replace g by the sequence $P_g=\mathtt{Cj}_{\rho\mapsto\sigma}P_1P_2^{(\tau)}P_1^{(\sigma^{-1})}P_2^{(\tau^{-1})}\mathtt{Cj}^{-1}_{\rho\mapsto\sigma}$, where $\rho=\sigma\tau\sigma^{-1}\tau^{-1}$. Note that the mapping of x to P_g is local, and that $P_g=\sigma^{g(x)}$. Finally, to simulate a negation of a gate g, let P_g be a sequence that simulates g, and let $P_{\neg g}=\sigma\mathtt{Cj}_{\sigma\mapsto\sigma^{-1}}P_g\mathtt{Cj}^{-1}_{\sigma\mapsto\sigma^{-1}}\sigma$.

Corollary B.5.5 (computing symmetric functions in $\mathcal{AC}^0$)**.** *For any constant integer $d \geq 3$, any symmetric function $\mathtt{Sym}\colon \{0,1\}^n \to \{0,1\}$ can be computed by a constant depth circuit of depth d and size $n^{O(n^{1/(d-1)})}$ with a bottom layer of OR gates, and by such a circuit with a bottom layer of AND gates.*

Proof. Denote $\mathtt{Sym}(x) = h\left(\sum_i x_i\right)$, for some composition function h. Consider the set $[n]$ with the action that maps any $x, y \in [n]$ to $x + y \pmod n$, and note that this yields a monoid. Given an input $x \in \{0,1\}^n$, we think of x as n elements in $[n]$, and instantiate Proposition B.5.3 with the monoid $\Sigma = [n]$ and the function h. ■

Theorem B.5.1 follows from Corollary B.5.5, by simulating each symmetric gate using a sub-exponential $\mathcal{AC}^0$ circuit. The only thing to be careful with is the parameters.

Theorem B.5.6 (parametrized version of Theorem B.5.1)**.** *For any two constants $k, d_0 \in \mathbb{N}$ there exists infinitely-many constants $d \in \mathbb{N}$ such that the following holds: If $f\colon \{0,1\}^n \to \{0,1\}$ cannot be computed by $\mathcal{AC}^0$ circuits of depth d and size $2^{\tilde{O}(n^{c_{k,d_0}/(d-1)})}$, where $c_{k,d_0} = k \cdot (d_0 - 1) + 1$, then f cannot be computed by $\mathcal{AC}^0[\mathtt{Sym}]$ circuits of depth d_0 and size m^k. In addition, if f cannot be computed by $\mathcal{AC}^0$ circuits of depth 6 and size $2^{\tilde{O}(n^{2/3})}$, then f cannot be computed by $\mathcal{AC}^0[\mathtt{Sym}]$ circuits of depth two and size $\tilde{O}(n^2)$.*

Proof. Let $C\colon \{0,1\}^n \to \{0,1\}$ be an $\mathcal{AC}^0[\mathtt{Sym}]$ circuit of depth d_0 and size $m = n^k$ that computes f, for some constant $k \in \mathbb{N}$. Our goal is to compute f by an $\mathcal{AC}^0$ circuit of depth d and size $2^{\tilde{O}(n^{c_{k,d_0}/(d-1)})}$, where d can be arbitrarily large.

To do so, fix an integer parameter $\alpha \geq 2$ (this parameter will determine the final depth d), and let $d_1 = \alpha \cdot k + 1$. We will use Corollary B.5.5 to compute any symmetric function by an $\mathcal{AC}^0$ circuit of depth d_1 and size $n^{O(n^{1/(d_1-1)})}$. For each of the top $d_0 - 1$ layers of C, we replace each gate in the layer by an $\mathcal{AC}^0$ circuit of depth d_1 that computes the corresponding symmetric function. Since each such gate has fan-in at most $m = n^k$, the size of the corresponding $\mathcal{AC}^0$ circuit is $2^{\tilde{O}(n^{k/(d_1-1)})}$. We also replace each of the gates in the bottom layer of C, which have fan-in at most n, by an $\mathcal{AC}^0$ circuit of depth $d_2 = \frac{d_1-1}{k} + 1$ and size $2^{\tilde{O}(n^{1/(d_2-1)})} = 2^{\tilde{O}(n^{k/(d_1-1)})}$.

To optimize this construction we use the fact (stated in Corollary B.5.5) that we can compute any symmetric function both with $\mathcal{AC}^0$ circuits whose bottom layer consists of AND gates, and with $\mathcal{AC}^0$ circuits as above whose bottom layer consists of OR gates. In each layer we use $\mathcal{AC}^0$ circuits with a bottom layer of either OR or AND gates, according to what would allow us to collapse a layer with the $\mathcal{AC}^0$ circuits in the layer beneath the current layer.[34] Thus, the final construction is of depth $d = d_2 + (d_0 - 1) \cdot (d_1 - 1) = \alpha \cdot c_{k,d_0} + 1$ and size $2^{\tilde{O}(n^{k/(d_1-1)})} = 2^{\tilde{O}(n^{c_{k,d_0}/(d-1)})}$.

For the "in addition" part, assume that f is computable by an $\mathcal{AC}^0[\mathtt{Sym}]$ circuit C of depth two and size $m = \tilde{O}(n^2)$. Using Corollary B.5.5, we replace the top gate of C by

[34] In this text I assume that all $\mathcal{AC}^0$ circuits are *layered*, in the sense that all gates of a fixed distance from the inputs are of the same type. Thus, if two consecutive layers consist of gates of the same type we can collapse both layers to one layer and reduce the depth of the circuit.

an $\mathcal{AC}^0$ circuit of depth $d_1 = 4$ and size $2^{\tilde{O}(m^{1/3})} = 2^{\tilde{O}(n^{2/3})}$, and we replace each of the gates in the bottom layer of C by an $\mathcal{AC}^0$ circuit of depth 3 and size $2^{\tilde{O}(\sqrt{n})}$. Collapsing a single layer between the two levels of $\mathcal{AC}^0$ circuits as in the proof of Theorem B.5.6, we obtain a circuit of depth 6 and size $2^{\tilde{O}(n^{2/3})}$. ■

For completeness, let us also show that significantly better sub-exponential lower bounds for $\mathcal{AC}^0$ would imply lower bounds for LTF circuits. To do so, we simulate LTF circuits by $\mathcal{AC}^0$ circuits, by first using the result of [GK98] to simulate any LTF circuit by a circuit with unweighted majority gates, and then relying on Theorem B.5.6 (since circuits with unweighted majority gates are a subclass of $\mathcal{AC}^0[\mathtt{Sym}]$).

Corollary B.5.7 (lower bounds for LTF circuits would follow from better $\mathcal{AC}^0$ lower bounds)**.** *For any two constants $k, d_0 \in \mathbb{N}$ there exists $c_{k,d_0} \in \mathbb{N}$ and infinitely-many constants $d \in \mathbb{N}$ such that the following holds. If a function $f\colon \{0,1\}^n \to \{0,1\}$ cannot be computed by $\mathcal{AC}^0$ circuits of depth d and size $2^{\tilde{O}(n^{c_{k,d_0}/(d-1)})}$, then f cannot be computed by* LTF *circuits of depth d_0 and size m^k.*

Proof. Recall that for every $k, d_0 \in \mathbb{N}$ there exists $k' \in \mathbb{N}$ such that any LTF circuit $C\colon \{0,1\}^n \to \{0,1\}$ of depth d_0 and with $m = n^k$ gates can be simulated by a circuit $C'\colon \{0,1\}^n \to \{0,1\}$ of depth $d_0 + 1$ with $n^{k'}$ unweighted majority gates (see [GK98]). In particular, the circuit C' is an $\mathcal{AC}^0[\mathtt{Sym}]$ circuit. By Theorem B.5.6, there exist infinitely-many $d \in \mathbb{N}$ such that C' can be simulated by an $\mathcal{AC}^0$ circuit of depth d and size $2^{\tilde{O}(n^{c_{k,d_0}/(d-1)})}$, for some c_{k,d_0} that depends only on k and d_0. ■

B.5.3 An analogous result: Implications of better lower bounds for LTF circuits

A similar (and even tighter) phenomenon happens when considering the implications of better lower bounds for LTF circuits (recall that these are constant-depth circuits with gates that compute arbitrary linear threshold functions).

As mentioned in Section B.5.1, we currently know lower bounds for LTF circuits of depth d with $n^{1+c^{-d}}$ wires, where $c \approx 2.41$ (see [IPS97], and also [CSS16; CT19]). Note that lower bounds for LTF circuits of *arbitrarily large polynomial size* and constant depth would imply lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$: In fact, when considering circuits of an arbitrarily large polynomial size and constant depth, then LTF circuits, $\mathcal{TC}^0$ circuits (with unweighted majority gates) and $\mathcal{AC}^0[\mathtt{Sym}]$ circuits can all compute exactly the same set of functions (i.e., each class of circuits can simulate the others with polynomial size overhead and constant depth overhead; see, e.g., [BBL92; GK98]). Nevertheless, as shown next, even much milder improvements in our lower bounds for LTF circuits – specifically, lower bounds for circuits with $n^{1+c^{-d}}$ wires where $c > 1$ is sufficiently small – would imply lower bounds for $\mathcal{AC}^0[\mathtt{Sym}]$ circuits with a super-linear number of wires, which are currently unknown (as far as I am aware).

Theorem B.5.8 ($\mathcal{AC}^0[\mathtt{Sym}]$ lower bounds would follow from better lower bounds for $\mathtt{LTF}$ circuits)**.** *For any constant $d_0 \in \mathbb{N}$ there exists a constant $c > 1$ and infinitely-many constants $d \in \mathbb{N}$ such that the following holds. If a function $f\colon \{0,1\}^n \to \{0,1\}$ cannot be computed by $\mathtt{LTF}$ circuits of depth d and with $n^{1+c^{-d}}$ wires, then f cannot be computed by $\mathcal{AC}^0[\mathtt{Sym}]$ circuits of depth d_0 and $O(n)$ wires.*

Proof. Let $C\colon \{0,1\}^n \to \{0,1\}$ be a function that is computable by an $\mathcal{AC}^0[\mathtt{Sym}]$ circuit of depth d_0 and with $m = O(n)$ wires. For $d_1 \geq 2$, recall that any symmetric function can be computed by an $\mathtt{LTF}$ circuit of depth d_1 and with $n^{1+1.41^{-d_1}}$ wires (see [BBL92] for details and for a finer bound that replaces 1.41 by $2^{1-1/d_1}$).

We replace each layer of gates in C by a layer of $\mathtt{LTF}$ circuits such that each gate g with fan-in n_g is replaced by an $\mathtt{LTF}$ circuit of depth d_1 with $(n_g)^{1+1.41^{-d_1}}$ wires. The overall depth of the resulting circuit is $d = d_0 \cdot d_1$, and its number of wires is

$$\sum_{g \text{ gate}} (n_g)^{1+1.41^{-d_1}} \leq \left(\sum_{g \text{ gate}} n_g \right)^{1+1.41^{-d_1}} = O\left(n^{1+c^{-d}}\right) ,$$

where $c = (1.41)^{1/d_0} > 1$. ∎

Bibliography

[Aar06] Scott Aaronson. "Oracles Are Subtle But Not Malicious". In: *Proc. 21st Annual IEEE Conference on Computational Complexity (CCC)*. 2006, pp. 340–354.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[ABN+92] N. Alon, J. Bruck, J. Naor, M. Naor, and R. M. Roth. "Construction of Asymptotically Good Low-rate Error-correcting Codes Through Pseudo-random Graphs". In: *IEEE Transactions on Information Theory* 38.2 (1992), pp. 509–516.

[Adl78] Leonard Adleman. "Two theorems on random polynomial time". In: *Proc. 19th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1978, pp. 75–83.

[Ajt83] M. Ajtai. "Σ_1^1-formulae on finite structures". In: *Annals of Pure and Applied Logic* 24.1 (1983), pp. 1–48.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Annals of Mathematics. Second Series* 160.2 (2004), pp. 781–793.

[AS15] Kazuyuki Amano and Atsushi Saito. "A nonuniform circuit class with multilayer of threshold gates having super quasi polynomial size lower bounds against NEXP". In: *Proc. 9th International Conference on Language and Automata Theory and Applications (LATA)*. 2015, pp. 461–472.

[ASW15] Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. "Reed-Muller codes for random erasures and errors". In: *IEEE Transactions on Information Theory* 61.10 (2015), pp. 5229–5252.

[AW85] Miklos Ajtai and Avi Wigderson. "Deterministic simulation of probabilistic constant depth circuits". In: *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1985.

[Bar89] David A. Mix Barrington. "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1". In: *Journal of Computer and System Sciences* 38.1 (1989), pp. 150–164.

[Baz09] Louay M. J. Bazzi. "Polylogarithmic Independence Can Fool DNF Formulas". In: *SIAM Journal of Computing* 38.6 (2009), pp. 2220–2272.

[BBG16] Arnab Bhattacharyya, Abhishek Bhowmick, and Chetan Gupta. "On higher-order Fourier analysis over non-prime fields". In: *Proc. 20th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2016, 23:1–23:29.

[BBL92] Paul Beame, Erik Brisson, and Richard Ladner. "The complexity of computing symmetric functions using threshold circuits". In: *Theoretical Computer Science* 100.1 (1992), pp. 253–265.

[BCG+96] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. "Oracles and queries that are sufficient for exact learning". In: *Journal of Computer and System Sciences* 52.3, part 1 (1996), pp. 421–433.

[BD18] Peter Beelen and Mrinmoy Datta. "Generalized Hamming weights of affine Cartesian codes". In: *Finite Fields and their Applications* 51 (2018), pp. 130–145.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. "Nondeterministic exponential time has two-prover interactive protocols". In: *Computational Complexity* 1.1 (1991), pp. 3–40.

[BFN+93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. "BPP has subexponential time simulations unless EXPTIME has publishable proofs". In: *Computational Complexity* 3.4 (1993), pp. 307–318.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. "Nonrelativizing separations". In: *Proc. 13th Annual IEEE Conference on Computational Complexity (CCC)*. 1998, pp. 8–12.

[BG81] Charles H. Bennett and John Gill. "Relative to a random oracle A, $\mathbf{P}^A \neq \mathbf{NP}^A \neq \mathrm{co}-\mathbf{NP}^A$ with probability 1". In: *SIAM Journal of Computing* 10.1 (1981), pp. 96–113.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Short PCPs Verifiable in Polylogarithmic Time". In: *Proc. 20th Annual IEEE Conference on Computational Complexity (CCC)*. 2005, pp. 120–134.

[BGY18] Paul Beame, Shayan Oveis Gharan, and Xin Yang. *On the Bias of Reed-Muller Codes over Odd Prime Fields*. 2018. eprint: arXiv:1806.06973.

[Bha14] Arnab Bhattacharyya. "Polynomial decompositions in polynomial time". In: *Proc. 22nd European Symposium on Algorithms (ESA)*. 2014, pp. 125–136.

[BHL12] Ido Ben-Eliezer, Rani Hod, and Shachar Lovett. "Random low-degree polynomials are hard to approximate". In: *Computational Complexity* 21.1 (2012), pp. 63–81.

[BHS08] Markus Bläser, Moritz Hardt, and David Steurer. "Asymptotically Optimal Hitting Sets Against Polynomials". In: *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part I*. Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP). 2008, pp. 345–356.

[BHT15] Arnab Bhattacharyya, Pooya Hatami, and Madhur Tulsiani. "Algorithmic regularity for polynomials and applications". In: *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2015, pp. 1870–1889.

[BIS12] Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. "Approximating AC^0 by small height decision trees and a deterministic algorithm for $\#AC^0SAT$". In: *Proc. 27th Annual IEEE Conference on Computational Complexity (CCC)*. 2012, pp. 117–125.

[BM84] Manuel Blum and Silvio Micali. "How to Generate Cryptographically Strong Sequences of Pseudo-random Bits". In: *SIAM Journal of Computing* 13.4 (1984), pp. 850–864.

[Bog05] Andrej Bogdanov. "Pseudorandom generators for low degree polynomials". In: *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*. 2005, pp. 21–30.

[Bop97] Ravi B. Boppana. "The average sensitivity of bounded-depth circuits". In: *Information Processing Letters* 63.5 (1997), pp. 257–261.

[BR94] M. Bellare and J. Rompel. "Randomness-efficient Oblivious Sampling". In: *Proc. 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1994, pp. 276–287.

[Bra10] Mark Braverman. "Polylogarithmic independence fools AC^0 circuits". In: *Journal of the ACM* 57.5 (2010).

[BRS91] Richard Beigel, Nick Reingold, and Daniel A. Spielman. "The perceptron strikes back". In: *Proc. 6th Annual IEEE Conference on Structure in Complexity Theory*. 1991, pp. 286–291.

[BS69] E. R. Berlekamp and N. J. A. Sloane. "Restrictions on weight distribution of Reed-Muller codes". In: *Information and Control* 14 (1969), pp. 442–456.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. "On the concrete efficiency of probabilistically-checkable proofs". In: *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC)*. 2013, pp. 585–594.

[BSGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Robust PCPs of proximity, shorter PCPs and applications to coding". In: *SIAM Journal of Computing* 36.4 (2006), pp. 889–974.

[BSV14] Eli Ben-Sasson and Emanuele Viola. "Short PCPs with projection queries". In: *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*. 2014, pp. 163–173.

[BV10] Andrej Bogdanov and Emanuele Viola. "Pseudorandom bits for polynomials". In: *SIAM Journal of Computing* 39.6 (2010), pp. 2464–2486.

[Cai07] Jin-Yi Cai. "$S_2^p \subseteq ZPP^{NP}$". In: *Journal of Computer and System Sciences* 73.1 (2007), pp. 25–35.

[Che19] Lijie Chen. "Non-deterministic Quasi-Polynomial Time is Average-case Hard for ACC Circuits". In: *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2019.

[CIS18] Marco L. Carmosino, Russell Impagliazzo, and Manuel Sabin. "Fine-grained derandomization: from problem-centric to resource-centric complexity". In: *Proc. 45th International Colloquium on Automata, Languages and Programming (ICALP)*. 2018, Art. No. 27, 16.

[CKK+15] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. "Mining circuit lower bound proofs for meta-algorithms". In: *Computational Complexity* 24.2 (2015), pp. 333–392.

[CL16] Kuan Cheng and Xin Li. "Randomness Extraction in AC0 and with Small Locality". In: *Electronic Colloquium on Computational Complexity: ECCC* 23 (2016), p. 18.

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Fourth. Undergraduate Texts in Mathematics. Springer, Cham, 2015.

[CMM+19] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. "Relations and Equivalences Between Circuit Lower Bounds and Karp-Lipton Theorems". In: *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 2019, 30:1–30:21.

[CNS99] Jin-Yi Cai, Ajay Nerurkar, and D. Sivakumar. "Hardness and hierarchy theorems for probabilistic quasi-polynomial time". In: *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC))*. 1999, pp. 726–735.

[Coo72] Stephen A. Cook. "A Hierarchy for Nondeterministic Time Complexity". In: *Proc. 4th Annual ACM Symposium on Theory of Computing (STOC)*. 1972, pp. 187–192.

[CR11] Venkatesan T. Chakaravarthy and Sambuddha Roy. "Arthur and Merlin as oracles". In: *Computational Complexity* 20.3 (2011), pp. 505–558.

[CR20] Lijie Chen and Hanlin Ren. "Strong Average-Case Circuit Lower Bounds from Non-trivial Derandomization". In: *Proc. 52th Annual ACM Symposium on Theory of Computing (STOC)*. 2020.

[CR96] Shiva Chaudhuri and Jaikumar Radhakrishnan. "Deterministic Restrictions in Circuit Complexity". In: *Proc. 28th Annual ACM Symposium on Theory of Computing (STOC)*. 1996, pp. 30–36.

[CRT+19] Lijie Chen, Ron Rothblum, Roei Tell, and Eylon Yogev. "On Exponential-Time Hypotheses, Derandomization, and Circuit Lower Bounds". In: *Electronic Colloquium on Computational Complexity: ECCC* 26 (2019), p. 169.

[CRV+02] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. "Randomness conductors and constant-degree lossless expanders". In: *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC)*. 2002, pp. 659–668.

[CSS16] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. "Average-case lower bounds and satisfiability algorithms for small threshold circuits". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. 2016, 1:1–1:35.

[CT19] Lijie Chen and Roei Tell. "Bootstrapping results for threshold circuits "just beyond" known lower bounds". In: *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 34–41.

[CTS13] Gil Cohen and Amnon Ta-Shma. "Pseudorandom Generators for Low Degree Polynomials from Algebraic Geometry Codes". In: *Electronic Colloquium on Computational Complexity: ECCC* 20 (2013), p. 155.

[DET+10] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. "Improved Pseudorandom Generators for Depth 2 Circuits". In: *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2010, pp. 504–517.

[DGJ+10] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. "Bounded independence fools halfspaces". In: *SIAM Journal of Computing* 39.8 (2010), pp. 3441–3462.

[DP09] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[DTST19] Dean Doron, Amnon Ta-Shma, and Roei Tell. "On Hitting-Set Generators for Polynomials that Vanish Rarely". In: *Electronic Colloquium on Computational Complexity: ECCC* 26 (2019), p. 119.

[Dvi09] Zeev Dvir. "On the size of Kakeya sets in finite fields". In: *Journal of the American Mathematical Society* 22.4 (2009), pp. 1093–1097.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. "On Completeness and Soundness in Interactive Proof Systems". In: *Advances in Computing Research* 5 (1989), pp. 429–442.

[FIK+08] Lance Fortnow, Russell Impagliazzo, Valentine Kabanets, and Christopher Umans. "On the complexity of succinct zero-sum games". In: *Computational Complexity* 17.3 (2008), pp. 353–376.

[FS16] Lance Fortnow and Rahul Santhanam. "New non-uniform lower bounds for uniform classes". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. 2016, Art. No. 19, 14.

[FS17] Lance Fortnow and Rahul Santhanam. "Robust simulations and significant separations". In: *Information and Computation* 256 (2017), pp. 149–159.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. "Parity, circuits, and the polynomial-time hierarchy". In: *Mathematical Systems Theory* 17.1 (1984), pp. 13–27.

[FSW09] Lance Fortnow, Rahul Santhanam, and Ryan Williams. "Fixed-polynomial size circuit bounds". In: *Proc. 24th Annual IEEE Conference on Computational Complexity (CCC)*. 2009, pp. 19–26.

[GHK+13] Anna Gál, Kristoffer Arnsfelt Hansen, Michal Koucký, Pavel Pudlák, and Emanuele Viola. "Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates". In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6611–6627.

[GHR92] Mikael Goldmann, Johan Håstad, and Alexander Razborov. "Majority gates vs. general weighted threshold gates". In: *Proc. 7th Annual Structure in Complexity Theory Conference*. 1992, pp. 2–13.

[Gil77] John Gill. "Computational complexity of probabilistic Turing machines". In: *SIAM Journal of Computing* 6.4 (1977), pp. 675–695.

[GK98] Mikael Goldmann and Marek Karpinski. "Simulating Threshold Circuits by Majority Circuits". In: *SIAM Journal of Computing* 27.1 (1998), pp. 230–246.

[GKM15] Parikshit Gopalan, Daniel Kane, and Raghu Meka. "Pseudorandomness via the discrete Fourier transform". In: *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2015, pp. 903–922.

[GL89] Oded Goldreich and Leonid A. Levin. "A Hard-core Predicate for All One-way Functions". In: *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*. 1989, pp. 25–32.

[GM15] Oded Goldreich and Or Meir. "Input-oblivious proof systems and a uniform complexity perspective on P/poly". In: *ACM Transactions on Computation Theory* 7.4 (2015), Art. 16, 13.

[GMR13] Parikshit Gopalan, Raghu Meka, and Omer Reingold. "DNF sparsification and a faster deterministic counting algorithm". In: *Computational Complexity* 22.2 (2013), pp. 275–310.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. New York, NY, USA: Cambridge University Press, 2008.

[Gol11] Oded Goldreich. "In a World of P=BPP". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*. 2011, pp. 191–232.

[Gol16] Oded Goldreich. "Deconstructing 1-local expanders". In: *Electronic Colloquium on Computational Complexity: ECCC* 23 (2016), p. 152.

[Gol17] Oded Goldreich. *Introduction to property testing*. Cambridge University Press, Cambridge, 2017, pp. xxv+445.

[GOW+10] Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. "Fooling functions of halfspaces under product distributions". In: *Proc. 25th Annual IEEE Conference on Computational Complexity (CCC)*. 2010, pp. 223–234.

[GR17] Oded Goldreich and Guy N. Rothblum. "Worst-case to Average-case reductions for subclasses of P". In: *Electronic Colloquium on Computational Complexity: ECCC* 26 (2017), p. 130.

[GRS19] Venkatesan Guruswami, Atri Rudra1, and Madhu Sudan. *Essential Coding Theory*. Accessed at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf. 2019.

[GS01] Venkatesan Guruswami and Madhu Sudan. "Extensions to the Johnson Bound". Manuscript. 2001.

[GS89] Yuri Gurevich and Saharon Shelah. "Nearly linear time". In: *Logic at Botik, Symposium on Logical Foundations of Computer Science*. Lecture Notes in Computer Science. 1989, pp. 108–118.

[GSTS03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. "Uniform hardness versus randomness tradeoffs for Arthur-Merlin games". In: *Computational Complexity* 12.3-4 (2003), pp. 85–130.

[GT09] Ben Green and Terence Tao. "The distribution of polynomials over finite fields, with applications to the Gowers norms". In: *Contributions to Discrete Mathematics* 4.2 (2009), pp. 1–36.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. "Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes". In: *Journal of the ACM* 56.4 (2009), Art. 20, 34.

[GV08] Dan Gutfreund and Salil Vadhan. "Limitations of hardness vs. randomness under uniform reductions". In: *Proc. 12th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2008, pp. 469–482.

[GVW15] Oded Goldreich, Emanuele Viola, and Avi Wigderson. "On Randomness Extraction in AC0". In: *Proc. 30th Annual IEEE Conference on Computational Complexity (CCC)*. 2015, pp. 601–668.

[GW13] Oded Goldreich and Avi Widgerson. "On derandomizing algorithms that err extremely rarely". In: *Electronic Colloquium on Computational Complexity: ECCC* 20 (2013), p. 152.

[GW14] Oded Goldreich and Avi Widgerson. "On derandomizing algorithms that err extremely rarely". In: *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*. Full version available online at *Electronic Colloquium on Computational Complexity: ECCC*, 20:152 (Rev. 2), 2013. 2014, pp. 109–118.

[Hås87] Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, 1987.

[Hås94] Johan Håstad. "On the size of weights for threshold gates". In: *SIAM Journal on Discrete Mathematics* 7.3 (1994), pp. 484–492.

[HR03] Tzvika Hartman and Ran Raz. "On the distribution of the number of roots of polynomials and explicit weak designs". In: *Random Structures & Algorithms* 23.3 (2003), pp. 235–263.

[HS10] Elad Haramaty and Amir Shpilka. "On the structure of cubic and quartic polynomials". In: *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*. 2010, pp. 331–340.

[HS16] Prahladh Harsha and Srikanth Srinivasan. "On polynomial approximations to AC^0". In: *Proc. 20th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2016.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "In search of an easy witness: exponential time vs. probabilistic polynomial time". In: *Journal of Computer and System Sciences* 65.4 (2002), pp. 672–694.

[IM83] Oscar H. Ibarra and Shlomo Moran. "Probabilistic algorithms for deciding equivalence of straight-line programs". In: *Journal of the Association for Computing Machinery* 30.1 (1983), pp. 217–228.

[IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. "A satisfiability algorithm for AC^0". In: *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2012, pp. 961–972.

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. "Pseudorandomness from shrinkage". In: *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2012, pp. 111–119.

[IPS13] Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. "A satisfiability algorithm for sparse depth two threshold circuits". In: *Proc. 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2013, pp. 479–488.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. "Size-depth tradeoffs for threshold circuits". In: *SIAM Journal of Computing* 26.3 (1997), pp. 693–707.

[IW98] R. Impagliazzo and A. Wigderson. "Randomness vs. Time: De-Randomization Under a Uniform Assumption". In: *Proc. 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1998, pp. 734–.

[IW99] Russell Impagliazzo and Avi Wigderson. "P = BPP if E requires exponential circuits: derandomizing the XOR lemma". In: *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*. 1999, pp. 220–229.

[JS12] Maurice Jansen and Rahul Santhanam. "Stronger lower bounds and randomness-hardness trade-offs using associated algebraic complexity classes". In: *Proc. 29th Symposium on Theoretical Aspects of Computer Science (STACS)*. Vol. 14. 2012, pp. 519–530.

[Juk12] Stasys Jukna. *Boolean function complexity*. Springer, Heidelberg, 2012.

[JVV86] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. "Random generation of combinatorial structures from a uniform distribution". In: *Theoretical Computer Science* 43.2-3 (1986), pp. 169–188.

[Kab01] Valentine Kabanets. "Easiness assumptions and hardness tests: trading time for zero error". In: vol. 63. 2. 2001, pp. 236–252.

[Kan11] Daniel M. Kane. "A small PRG for polynomial threshold functions of Gaussians". In: *Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2011, pp. 257–266.

[Kan14] Daniel M. Kane. "A pseudorandom generator for polynomial threshold functions of Gaussian with subpolynomial seed length". In: *Proc. 29th Annual IEEE Conference on Computational Complexity (CCC)*. 2014, pp. 217–228.

[Kan82] R. Kannan. "Circuit-size lower bounds and non-reducibility to sparse sets". In: *Information and Control* 55.1-3 (1982), pp. 40–56.

[KI04] Valentine Kabanets and Russell Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds". In: *Computational Complexity* 13.1-2 (2004), pp. 1–46.

[KL08] Tali Kaufman and Shachar Lovett. "Worst Case to Average Case Reductions for Polynomials". In: *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2008, pp. 166–175.

[KL80] Richard M. Karp and Richard J. Lipton. "Some connections between nonuniform and uniform complexity classes". In: *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC)*. 1980.

[KLP12] Tali Kaufman, Shachar Lovett, and Ely Porat. "Weight distribution and list-decoding size of Reed-Muller codes". In: *IEEE Transactions on Information Theory* 58.5 (2012), pp. 2689–2696.

[KM15] Pravesh K. Kothari and Raghu Meka. "Almost optimal pseudorandom generators for spherical caps". In: *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC)*. 2015, pp. 247–256.

[KMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. "Pseudorandom generators, typically-correct derandomization, and dircuit lower bounds". In: *Computational Complexity* 21.1 (2012), pp. 3–61.

[KRS12] Zohar S. Karnin, Yuval Rabani, and Amir Shpilka. "Explicit dimension reduction and its applications". In: *SIAM Journal of Computing* 41.1 (2012), pp. 219–249.

[KS01] Adam R. Klivans and Daniel Spielman. "Randomness efficient identity testing of multivariate polynomials". In: *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*. 2001, pp. 216–223.

[KS12] Swastik Kopparty and Srikanth Srinivasan. "Certifying polynomials for $AC^0[\oplus]$ circuits, with applications". In: *Proc. 32nd Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 2012, pp. 36–47.

[KT70] Tadao Kasami and Nobuki Tokura. "On the weight structure of Reed-Muller codes". In: *IEEE Transactions on Information Theory* IT-16 (1970), pp. 752–759.

[KUW88] Richard M. Karp, Eli Upfal, and Avi Wigderson. "The complexity of parallel search". In: *Journal of Computer and System Sciences* 36.2 (1988), pp. 225–253.

[KW16] Daniel M. Kane and Ryan Williams. "Super-linear Gate and Super-quadratic Wire Lower Bounds for Depth-two and Depth-three Threshold Circuits". In: *Proc. 48th Annual ACM Symposium on Theory of Computing (STOC)*. 2016, pp. 633–643.

[KW99] Johannes Köbler and Osamu Watanabe. "New collapse consequences of NP having small circuits". In: *SIAM Journal of Computing* 28.1 (1999), pp. 311–324.

[Lau83] Clemens Lautemann. "BPP and the polynomial hierarchy". In: *Information Processing Letters* 17.4 (1983), pp. 215–217.

[LFK+92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic methods for interactive proof systems". In: *Journal of the Association for Computing Machinery* 39.4 (1992), pp. 859–868.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. "Constant depth circuits, Fourier transform, and learnability". In: *Journal of the Association for Computing Machinery* 40.3 (1993), pp. 607–620.

[LN90] Nathan Linial and Noam Nisan. "Approximate Inclusion-Exclusion". In: *Combinatorica* 10.4 (1990), pp. 349–365.

[Lov09] Shachar Lovett. "Unconditional pseudorandom generators for low-degree polynomials". In: *Theory of Computing* 5 (2009), pp. 69–82.

[LRR17] Yuan Li, Alexander Razborov, and Benjamin Rossman. "On the AC^0 complexity of subgraph isomorphism". In: *SIAM Journal of Computing* 46.3 (2017), pp. 936–971.

[LRT+09] Shachar Lovett, Omer Reingold, Luca Trevisan, and Salil Vadhan. "Pseudorandom bit generators that fool modular sums". In: *Proc. 13th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2009, pp. 615–630.

[Lu01] Chi-Jen Lu. "Derandomizing Arthur-Merlin games under uniform assumptions". In: *Computational Complexity* 10.3 (2001), pp. 247–259.

[Lu12] Chi-Jen Lu. "Hitting set generators for sparse polynomials over any finite fields". In: *Proc. 27th Annual IEEE Conference on Computational Complexity (CCC)*. 2012, pp. 280–286.

[LV96] M. Luby and B. Veličković. "On deterministic approximation of DNF". In: *Algorithmica* 16.4-5 (1996), pp. 415–433.

[LV98] Daniel Lewin and Salil Vadhan. "Checking polynomial identities over any field: towards a derandomization?" In: *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*. 1998, pp. 438–447.

[LVW93] M. Luby, B. Velickovic, and A. Wigderson. "Deterministic approximate counting of depth-2 circuits". In: *Proc. 2nd Israel Symposium on Theory and Computing Systems*. 1993, pp. 18–24.

[LW13] Richard J. Lipton and Ryan Williams. "Amplifying circuit lower bounds against polynomial time, with applications". In: *Computational Complexity* 22.2 (2013), pp. 311–343.

[LY94] Richard J. Lipton and Neal E. Young. "Simple Strategies for Large Zero-sum Games with Applications to Complexity Theory". In: *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*. 1994, pp. 734–740.

[Man95] Yishay Mansour. "An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution". In: *Journal of Computer and System Sciences* 50.3, part 3 (1995), pp. 543–550.

[McE69] R. J. McEliece. "Quadratic forms over finite fields and second- order Reed-Muller codes". In: *Space Program Summary* 3.37–58 (1969), pp. 28–33.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes. II*. North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977.

[MW17] Cody Murray and Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP". In: *Electronic Colloquium on Computational Complexity: ECCC* 24 (2017), p. 188.

[MW18] Cody Murray and Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP". In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018.

[MZ13] Raghu Meka and David Zuckerman. "Pseudorandom generators for polynomial threshold functions". In: *SIAM Journal of Computing* 42.3 (2013), pp. 1275–1301.

[Nis91] Noam Nisan. "Pseudorandom bits for constant depth circuits". In: *Combinatorica* 11.1 (1991), pp. 63–70.

[NN93] Joseph Naor and Moni Naor. "Small-bias probability spaces: efficient constructions and applications". In: *SIAM Journal of Computing* 22.4 (1993), pp. 838–856.

[NW15] Zipei Nie and Anthony Y. Wang. "Hilbert functions and the finite degree Zariski closure in finite field combinatorial geometry". In: *Journal of Combinatorial Theory. Series A* 134 (2015), pp. 196–220.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[NZ96] Noam Nisan and David Zuckerman. "Randomness is Linear in Space". In: *Journal of Computer and System Sciences* 52.1 (1996), pp. 43–52.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[Oli13] Igor C. Oliveira. "Algorithms versus Circuit Lower Bounds". In: *Electronic Colloquium on Computational Complexity: ECCC* 20 (2013), p. 117.

[OS17] Igor C. Oliveira and Rahul Santhanam. "Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness". In: *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 79. 2017, Art. No. 18, 49.

[OST19] Igor C. Oliveira, Rahul Santhanam, and Roei Tell. "Expander-based cryptography meets natural proofs". In: *Proc. 10th Conference on Innovations in Theoretical Computer Science (ITCS)*. 2019, Art. No. 18, 14.

[PF79] Nicholas Pippenger and Michael J. Fischer. "Relations among complexity measures". In: *Journal of the ACM* 26.2 (1979), pp. 361–381.

[PS94] Ramamohan Paturi and Michael E. Saks. "Approximating threshold circuits by rational functions". In: *Information and Computation* 112.2 (1994), pp. 257–272.

[Raz09] Alexander Razborov. "A Simple Proof of Bazzi's Theorem". In: *ACM Transactions on Computation Theory* 1 (2009), p. 3.

[Raz87] Alexander A. Razborov. "Lower bounds on the size of constant-depth networks over a complete basis with logical addition". In: *Mathematical Notes of the Academy of Science of the USSR* 41.4 (1987), pp. 333–338.

[Ros08] Benjamin Rossman. "On the constant-depth complexity of k-clique". In: *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*. 2008, pp. 721–730.

[Ros14] Benjamin Rossman. "The monotone complexity of k-clique on random graphs". In: 43.1 (2014), pp. 256–279.

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. "Extracting all the randomness and reducing the error in Trevisan's extractors". In: *Journal of Computer and System Sciences* 65.1 (2002), pp. 97–128.

[RS10] Yuval Rabani and Amir Shpilka. "Explicit Construction of a Small epsilon-Net for Linear Threshold Functions". In: *SIAM Journal of Computing* 39.8 (2010), pp. 3501–3520.

[San09] Rahul Santhanam. "Circuit lower bounds for Merlin-Arthur classes". In: *SIAM Journal of Computing* 39.3 (2009), pp. 1038–1061.

[San10] Rahul Santhanam. "Fighting perebor: new and improved algorithms for formula and QBF satisfiability". In: *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 183–192.

[SB70] Neil J. A. Sloane and Elwyn R. Berlekamp. "Weight enumerator for second-order Reed-Muller codes". In: *IEEE Transactions on Information Theory* IT-16 (1970), pp. 745–751.

[Ser07] Rocco A. Servedio. "Every linear threshold function has a low-weight approximator". In: *Computational Complexity* 16.2 (2007), pp. 180–209.

[Sha92] Adi Shamir. "IP = PSPACE". In: *Journal of the ACM* 39.4 (1992), pp. 869–877.

[Sho90] Victor Shoup. "New algorithms for finding irreducible polynomials over finite fields". In: *Mathematics of Computation* 54.189 (1990), pp. 435–447.

[Smo87] Roman Smolensky. "Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity". In: *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*. 1987, pp. 77–82.

[SST+16] Takayuki Sakai, Kazuhisa Seto, Suguru Tamaki, and Junichi Teruyama. "Bounded depth circuits with weighted symmetric gates: satisfiability, lower bounds and compression". In: *Proc. 41st International Symposium on Mathematical Foundations of Computer Science*. 2016.

[ST12] K. Seto and S. Tamaki. "A Satisfiability Algorithm and Average-Case Hardness for Formulas over the Full Binary Basis". In: *Proc. 27th Annual IEEE Conference on Computational Complexity (CCC)*. 2012, pp. 107–116.

[ST17a] Rocco Servedio and Li-Yang Tan. "Deterministic search for CNF satisfying assignments in almost polynomial time". In: *Proc. 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2017.

[ST17b] Rocco Servedio and Li-Yang Tan. "Learning and fooling depth-two threshold circuits". Unpublished manuscript. 2017.

[ST18] Rocco A. Servedio and Li-Yang Tan. "Luby-Veličković-Wigderson revisited: improved correlation bounds and pseudorandom generators for depth-two circuits". In: *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. Vol. 116. 2018, Art. No. 56, 20.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. "Pseudorandom generators without the XOR lemma". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 236–266.

[SU05] Ronen Shaltiel and Christopher Umans. "Simple extractors for all min-entropies and a new pseudorandom generator". In: *Journal of the ACM* 52.2 (2005), pp. 172–216.

[SU07] Ronen Shaltiel and Christopher Umans. "Low-end uniform hardness vs. randomness tradeoffs for AM". In: *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*. 2007, pp. 430–439.

[SW13] Rahul Santhanam and Ryan Williams. "On medium-uniformity and circuit lower bounds". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 2013, pp. 15–23.

[Tal17] Avishay Tal. "Tight Bounds on the Fourier Spectrum of AC0". In: *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. 2017, 15:1–15:31.

[Tam16] Suguru Tamaki. "A Satisfiability Algorithm for Depth Two Circuits with a Sub-Quadratic Number of Symmetric and Threshold Gates". In: *Electronic Colloquium on Computational Complexity: ECCC* 23 (2016), p. 100.

[Tar93] Jun Tarui. "Probabilistic Polynomials, AC^0 Functions and the Polynomial-time Hierarchy". In: *Theoretical Computer Science* 113.1 (1993), pp. 167–183.

[Tel17a] Roei Tell. "A Note on the Limitations of Two Black-Box Techniques in Quantified Derandomization". In: *Electronic Colloquium on Computational Complexity: ECCC* 24 (2017), p. 187.

[Tel17b] Roei Tell. "Improved Bounds for Quantified Derandomization of Constant-Depth Circuits and Polynomials". In: *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. 2017, 18:1 –18:49.

[Tel18a] Roei Tell. "Lower bounds on black-box reductions of hitting to density estimation". In: *Proc. 35th Symposium on Theoretical Aspects of Computer Science (STACS)*. 2018, Art. No. 58, 13.

[Tel18b] Roei Tell. "Quantified Derandomization of Linear Threshold Circuits". In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 855–865.

[Tel19a] Roei Tell. "Improved bounds for quantified derandomization of constant-depth circuits and polynomials". In: *Computational Complexity* 28.2 (2019), pp. 259–343.

[Tel19b] Roei Tell. "Proving that $prBPP = prP$ is as hard as proving that "almost NP" is not contained in P/poly". In: *Information Processing Letters* 152 (2019), p. 105841.

[Tel20] Roei Tell. "A Note on Tolerant Testing with One-Sided Error". In: *Computational Complexity and Property Testing*. Ed. by Oded Goldreich. Springer, 2020, pp. 173–178.

[Tod91] Seinosuke Toda. "PP is as hard as the polynomial-time hierarchy". In: *SIAM Journal of Computing* 20.5 (1991), pp. 865–877.

[Tre01] Luca Trevisan. "Extractors and Pseudorandom Generators". In: *Journal of the ACM* 48.4 (2001), pp. 860–879.

[TSZ04] Amnon Ta-Shma and David Zuckerman. "Extractor codes". In: *IEEE Transactions on Information Theory* 50.12 (2004), pp. 3015–3025.

[TSZS06] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. "Extractors from Reed-Muller codes". In: *Journal of Computer and System Sciences* 72.5 (2006), pp. 786–812.

[TV07] Luca Trevisan and Salil P. Vadhan. "Pseudorandomness and Average-Case Complexity Via Uniform Reductions". In: *Computational Complexity* 16.4 (2007), pp. 331–364.

[TX13] Luca Trevisan and TongKe Xue. "A derandomized switching lemma and an improved derandomization of AC0". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 2013, pp. 242–247.

[Uma03] Christopher Umans. "Pseudo-random generators for all hardnesses". In: *Journal of Computer and System Sciences* 67.2 (2003), pp. 419–440.

[Vad12] Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

[Val77] Leslie G. Valiant. "Graph-theoretic arguments in low-level complexity". In: *Proc. 6th International Symposium on Mathematical Foundations of Computer Science*. 1977, pp. 162–176.

[Vin05] N. V. Vinodchandran. "A note on the circuit complexity of PP". In: *Theoretical Computer Science* 347.1-2 (2005), pp. 415–418.

[Vio05] Emanuele Viola. "The complexity of constructing pseudorandom generators from hard functions". In: *Computational Complexity* 13.3-4 (2005), pp. 147–188.

[Vio09a] Emanuele Viola. "Guest Column: correlation bounds for polynomials over 0 1." In: *SIGACT News* 40 (Feb. 2009), pp. 27–44.

[Vio09b] Emanuele Viola. "The sum of d small-bias generators fools polynomials of degree d". In: *Computational Complexity* 18.2 (2009), pp. 209–217.

[Vio17] Emanuele Viola. "Selected challenges in computational lower bounds". In: *ACM SIGACT News* 48.1 (2017), pp. 39–45.

[Vio19] Emanuele Viola. *Why do lower bounds stop "just before" proving major results?* Accessed at http://www.ccs.neu.edu/home/viola/talks/lower-bounds-201909.pdf, April 6, 2020. 2019.

[VW17] Emanuele Viola and Avi Wigderson. "Local Expanders". In: *Computational Complexity* (2017).

[War35] Ewald Warning. "Bemerkung zur vorstehenden Arbeit von Herrn Chevalley". In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 11 (1935), pp. 76–83.

[Wil11] Ryan Williams. "Non-uniform ACC circuit lower bounds". In: *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*. 2011, pp. 115–125.

[Wil13] Ryan Williams. "Improving Exhaustive Search Implies Superpolynomial Lower Bounds". In: *SIAM Journal of Computing* 42.3 (2013), pp. 1218–1244.

[Wil14a] Ryan Williams. "Algorithms for circuits and circuits for algorithms: Connecting the tractable and intractable". In: *Proc. International Congress of Mathematicians (ICM)*. 2014, pp. 659–682.

[Wil14b] Ryan Williams. "New algorithms and lower bounds for circuits with linear threshold gates". In: *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*. 2014, pp. 194–202.

[Wil16] Richard Ryan Williams. "Strong ETH breaks with Merlin and Arthur: short non-interactive proofs of batch evaluation". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 50. 2016, Art. No. 2, 17.

[Yao82] Andrew C. Yao. "Theory and Application of Trapdoor Functions". In: *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1982, pp. 80–91.

[Yao85] Andrew C-C. Yao. "Separating the Polynomial-time Hierarchy by Oracles". In: *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1985, pp. 1–10.