Better Hardness via Algorithms, and New Forms of Hardness versus Randomness

by

Lijie Chen

B.S., Tsinghua University (2017)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author .....
Department of Electrical Engineering and Computer Science
August 26, 2022

Certified by .....
Ryan Williams
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Better Hardness via Algorithms, and New Forms of Hardness versus Randomness

by

Lijie Chen

Submitted to the Department of Electrical Engineering and Computer Science
on August 26, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

One central theme of complexity theory is the rich interplay between hardness (the existence of functions that are hard to compute) and pseudorandomness (the procedure that converts randomized algorithms into equivalent deterministic algorithms). In one direction, from the classic works of Nisan-Widgerson and Impagliazzo-Widgerson, we know certain hardness hypothesis (circuit lower bounds) implies that all randomized algorithms can be derandomized with a polynomial overhead. In another direction, A decade ago, Williams have proved that certain circuit lower bounds follows from non-trivial derandomization.

In this thesis we establish many new connections between hardness and pseudorandomness, strengthening and refining the classic works mentioned above.

- **New circuit lower bounds from non-trivial derandomization.** Following Williams' algorithmic method, we prove several new circuit lower bounds using various non-trivial derandomization algorithms, including almost-everywhere and strongly average-case lower bound against ACC^0 circuits and a new construction of rigid matrices.
- **Superfast and non-black-box derandomization from plausible hardness assumptions.** Under plausible hardness hypotheses, we obtain almost optimal worst-case derandomization of both randomized algorithms and constant-round Arthur-Merlin protocols. We also propose a new framework for non-black-box derandomization and demonstrate its usefulness by showing (1) it connects derandomization to a new type of hardness assumptions that is against *uniform algorithms* and (2) (from plausible assumptions) it gives derandomization of both randomized algorithms and constant-round doubly efficient proof systems with almost no overhead such that no polynomial-time adversary can find a mistake.

Thesis Supervisor: Ryan Williams
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First, of course, I would like to express my deepest gratitude to my Ph.D. advisor Ryan Williams for all the guidance and support in my graduate career. Ryan is a fantastic advisor who is always very kind and supportive and often offers precious advice. Sometimes Ryan looks like an omnipotent oracle in all his insightful observations. I still remember that many years ago, when I was very young, I learned a little bit of complexity theory and tried to read several papers of Ryan's (including the famous work showing $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$). The elegance of his papers fascinated me, and I wanted to prove something like that. It is a miracle that I am working with this man many years later and have improved some of his theorems!

I wish to thank my other thesis committee members, Michael Sipser and Anand Natarajan.

This thesis is based on seven papers [CW19, CR20, CLW20, CL21, CT21b, CT21a, CT22] by my coauthors and me. I want to thank all my coauthors on these projects: without them, these works would be impossible, including Xin Lyu, Hanlin Ren, Roei Tell, and Ryan Williams.

I want to thank my undergrad mentor, Scott Aaronson. Scott hosted me while I was visiting MIT during the spring of 2016 and introduced me to the world of complexity theory. We had a lot of fun together thinking about quantum complexity. During that visit, I often thought very hard about problems until 7 am, and woke up sometime in the afternoon and repeated. I probably cannot (and should not) do that now, but that is an excellent memory. Scott's book, Quantum Computing Since Democritus, has a massive influence on me, which is another reason why I wanted to study TCS in the first place.

I also want to thank my supervisor at Tsinghua University, Jian Li, for introducing me to the beautiful world of theoretical computer science. I still remember that when I was a second-year undergraduate with no knowledge of TCS. I took his advanced TCS course and was overwhelmed by the intense workload of 20 hours a week. But it turned out to be a super rewarding experience, and I found myself working on Multi-Armed Bandit with him for the following years, with which I had a lot of fun.

During my Ph.D., I have visited many places and done two research internships, and I want to thank all my mentors and hosts for their kindness and support. In the fall of 2018, I visited the Simons Institute for the Theory of Computing (one of my favorite places on earth). In early 2019, I visited Scott Aaronson in Austin. It's certainly a much better place to be in winter compared to Boston, and I was happy to work with him again. In the summer of 2019, I went to Oxford and had

great fun with Rahul Santhanam, Igor Carboni Oliveira, and Ján Pich. I would come back to revisit them in March 2020. In early 2020, I went to Israel for the first time and visited Guy Rothblum. Guy is such a fun guy to work with, and we had a great time together. I wish I could make more visits, but sadly I could not due to the pandemic. Still, I managed to do two rewarding remote internships at Google and IBM. At Google, I worked on differential privacy with Ravi Kumar, Pasin Manurangsi, and Badih Ghazi; at IBM, I worked on quantum complexity theory with Ramis Movassagh.

After the pandemic was finally over, I visited Avi Wigderson at IAS in the spring of 2022. Avi is a giant in complexity theory and also one of my idols. All the works covered in this thesis can trace back to his early foundational works, and I was super happy to be able to visit him after the pandemic.

I want to thank all my visiting students (most of them are co-mentored with Ryan Williams) over the years: Kaifeng Lyu, Ce Jin, Hanlin Ren, Xin Lyu, Hongxun Wu, and Tianqi Yang. I am incredibly fortunate to work with all these super-talented students. In my career, I have benefited tremendously from all my amazing supervisors, so I hope to be a good mentor as well.

I want to thank the MIT theory group. I had a lot of fun with the people there, especially during all those fantastic theory retreats. It is a bit unfortunate that my Ph.D. was split into two by the pandemic, and there were not many activities during the pandemic. Still, MIT is a friendly and inclusive environment, and I am sad I have to leave.

I want to thank my former and current roommates, Toru Lin, Sophie the cat, and Sofia the cat, for all the fun we had together. Sophie and Sofia often want to play with me and do not allow me to work on my thesis (they are so adorable that I cannot refuse them). I guess without them, this thesis can be finished a month ago.

I also want to thank all the senior researchers who have helped me and are not mentioned before; a very partial list includes Oded Goldreich, Virginia Vassilevska Williams, Huacheng Yu, Zhao Song, Justin Thaler, Gillat Kol, Shafi Goldwasser, Aviad Rubinstein, Shuichi Hirahara, and Ron Rothblum.

I would love to thank all my friends who have supported me over the last five years. This acknowledgment is way too short to list all the names here, but you know who you are.

Finally, I want to thank my family for their continuing and unconditional support.

Contents

List of Figures

Part I

Introduction

Organization

This thesis consists of two lines of work that complement each other. In Chapter 1, we will describe our works on proving new circuit lower bounds using non-trivial derandomization. Next, in Chapter 2, we will describe our works on getting superfast and non-black-box derandomization from plausible hardness assumptions.

Chapter 1

Algorithmic Method: Circuit Lower Bounds from Non-trivial Derandomization

1.1 Background on Circuit Lower Bounds and the Algorithmic Method

1.1.1 Circuit Lower Bounds: Complexity Theory's Waterloo

Proving circuit lower bounds for explicit functions (with the flagship problem of $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$) is one of the central problems in theoretical computer science.[1] In the 1980s, considerable progress was made in proving lower bounds for constant-depth circuits, as first steps towards lower bounds for general circuits. The classical works [Ajt83, FSS84, Yao85, Hås89] and [Raz87, Smo87] culminated in exponential lower bounds against AC^0 (constant depth circuits consisting of unbounded fan-in AND/OR gates and fan-in 1 NOT gates) and against $\mathsf{AC}^0[q]$ (AC^0 circuits extended with unbounded fan-in MOD_q gates) where q is a prime power, respectively.[2]

Unfortunately, the progress in the 1980s did not go much further: lower bounds against $\mathsf{AC}^0[m]$ have been extremely difficult to establish for composite m, despite the conjecture that $\mathsf{AC}^0[m]$ cannot compute the majority function. In fact, it was a notorious open question whether NEXP (nondeterministic exponential time) has polynomial-size ACC^0 circuits.[3]

[1] The title of this subsection is taken from the title of [AB09, Section 14].

[2] For an integer m, the function $\mathsf{MOD}_m : \{0,1\}^* \to \{0,1\}$ is one if and only if the number of ones in the input is not divisible by m. We often also use $\mathsf{AC}^0[\oplus]$ to denote $\mathsf{AC}^0[2]$.

[3] This had been stressed several times as one of the most embarrassing open questions in complexity theory,

1.1.2 Williams' Algorithmic Method for Proving Circuit Lower Bounds

A decade ago, Williams [Wil11] finally proved $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$, via an *algorithmic* approach (which is often called Williams' algorithmic method) to circuit lower bounds [Wil10].[4] Combining many results from classical complexity theory, such as the nondeterministic time hierarchy theorem [SFM78, Žák83], hardness vs randomness [NW94], and the PCP Theorem [ALM+98, AS98], Williams' work shows how non-trivial circuit-analysis algorithms can be generically applied to prove circuit lower bounds.

In more details, Williams' approach was not limited to the circuit class ACC^0 and indeed gave a general connection between *circuit-analysis algorithms* and lower bounds. To discuss his approach more formally, we recall the definitions of the following standard circuit-analysis problems.

1. SAT **(Satisfiability):** Given a circuit C, determine if there is an input a such that $C(a) = 1$.
2. #SAT **(Counting Satisfiable Assignment):** Given a circuit C on n inputs, compute the number of assignments $x \in \{0,1\}^n$ such that $C(x) = 1$.
3. CAPP **(Circuit Acceptance Probability Problem) with error δ (denoted CAPP_δ):** Given a circuit C on n inputs, estimate $\Pr_{x \in_R \{0,1\}^n}[C(x) = 1]$ within an *additive error* of δ. When not explicitly stated, δ is set to be $1/3$ by default.

Williams [Wil10, Wil11] proved the following generic connection.

Theorem 1.1.1 ([Wil10, Wil11], Informal)**.** *Let $\mathscr{C}$ be a reasonable circuit class.*[5] *If there is a $2^n/n^{\omega(1)}$-time algorithm for* SAT *(or* CAPP*) of $\mathrm{poly}(n)$-size n-input $\mathscr{C}$ circuits*[6]*, then $\mathsf{NEXP} \not\subseteq \mathscr{C}$.*

To prove $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$, Williams [Wil11] designed the required non-trivial SAT algorithm for ACC^0 circuits and applied the above general connection.[7] This generic connection made Williams' approach very appealing: to prove new circuit lower bounds against a circuit class $\mathscr{C}$, we only have to get a *barely-faster-than-brute-force* (*i.e.*, non-trivial) SAT or CAPP algorithm for $\mathscr{C}$ (the brute-force algorithm that enumerates all 2^n inputs take $2^n \cdot \mathrm{poly}(n)$ time for both tasks). Moreover, since it is widely believed that $\mathsf{prP} = \mathsf{prBPP}$ [NW94, IW97], CAPP for even general circuits[8] is

see [AB09]. Note that ACC^0 denotes the union of $\mathsf{AC}^0[m]$ for all constants m.

[4] The journal versions of [Wil10] and [Wil11] are [Wil13a] and [Wil14b], respectively.

[5] *e.g.*, ACC^0, TC^0, De-Morgan formulas, or fan-in 2 De-Morgan circuits. More formally, it is required that $\mathscr{C}$ to satisfy certain "closure properties". The original paper of [Wil11] requires that $\mathscr{C} \circ \mathscr{C} \subseteq \mathscr{C}$ and $\mathsf{AC}^0 \circ \mathscr{C} \subseteq \mathscr{C}$ (see [Wil14b, Remark 3.5]). As we will see shortly, this condition has been weaken by subsequent work; see Theorem 4.1.3 for a formal statement with an improved condition on the circuit class.

[6] We stress that the SAT (CAPP) algorithm needs to work for n^k-size $\mathscr{C}$ circuits *for all* k.

[7] Indeed, [Wil11] proved a much stronger result: for every $d, m \in \mathbb{N}_{\geq 1}$, there is an $\varepsilon \in (0,1)$ such that SAT for 2^{n^ε}-size $\mathsf{AC}^0_d[m]$ circuits can be solved in deterministic 2^{n-n^ε} time.

[8] Throughout this chapter, general circuits refer to fan-in 2 De-Morgan circuits unless otherwise specified.

believed to admit a polynomial-time algorithm!

We remark that preceding Williams' work [Wil10, Wil11], a seminal work by Impagliazzo, Kabanets and Wigderson [IKW02] already established that *subexponential-time derandomization* of polynomial-size general circuits implies that $\mathsf{NEXP} \not\subseteq \mathsf{P}_{/\mathsf{poly}}$.[9] Theorem 1.1.1 is directly inspired by the method of [IKW02] and in particular, its proof crucially used the *easy witness lemma* from [IKW02].

1.1.3 Recent Developments in Algorithmic Method

Williams' algorithmic method has attracted a significant amount of research interest in the last decade, and many follow-up works have been devoted to further refining this method and proving stronger circuit lower bounds. In the following, we summarize the main developments since [Wil10, Wil11].[10]

Notation for standard concepts. We first recall some notation for standard concepts. We use $\mathscr{C}_1 \circ \mathscr{C}_2$ to refer to circuit families consisting of a top circuit from $\mathscr{C}_1$ composed with bottom circuits from $\mathscr{C}_2$.[11] We also use $\mathsf{AND}_m \circ \mathscr{C}$ to denote an AND of m $\mathscr{C}$ subcircuits ($\mathsf{OR}_m \circ \mathscr{C}$ and $\oplus_m \circ \mathscr{C}$ are similarly defined). A $\mathsf{MAJ}\colon \{0,1\}^m \to \{0,1\}$ gate $\mathsf{MAJ}(y_1, \ldots, y_m)$ outputs 1 if and only if $\sum_i y_i \geq m/2$. TC^0 is the class of circuit families of constant depth and polynomial size, with unbounded fan-in MAJ gates. We also use TC^0_d to denote the sub-class of TC^0 of depth d. A function $f\colon \{0,1\}^n \to \{0,1\}$ is a linear threshold function (THR) if there are weights $w_1, \ldots, w_n, t \in \mathbb{R}$ such that, for $x \in \{0,1\}^n$, we have $f(x) = 1$ if and only if $\sum_{i=1}^n w_i \cdot x_i \geq t$.

Tightening the connection. The original proof of Theorem 1.1.1 only works for circuit classes $\mathscr{C}$ that are closed under compositions (the composition of two circuit families from $\mathscr{C}$ is also a family in $\mathscr{C}$, *e.g.*, AC^0, formulas, or general circuits). This becomes problematic when working with fixed-depth circuit classes such as $\mathsf{THR} \circ \mathsf{THR}$ (it is still open that whether $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$). Several follow-up works [SW13, Wil14a, JMV15, BV14] have considerably weakened the condition on circuit classes $\mathscr{C}$ in Theorem 1.1.1. In particular, following the work by Jahanjou, Miles, and Viola [JMV15] (its Arxiv version appeared in 2013), Ben-Sasson and Viola [BV14] proved that for *any* circuit class $\mathscr{C}$ that is closed under negation and projection, $\mathsf{NEXP} \not\subseteq \mathscr{C}$ follows from a non-

[9] They in fact obtained an equivalence by adding more technical conditions to the derandomization algorithm. A follow-up work to [IKW02] by Kabanets and Impagliazzo [KI04] obtained circuit lower bound consequence assuming subexponential-time derandomization of the polynomial identity testing problem.

[10] In this subsection we only summarize works that are prior to the author's works. Works that are concurrent to the author's works will be discussed in later sections.

[11] As usual, in the case of $\mathscr{C}_2 = \mathsf{ACC}^0$, where $\mathsf{ACC}^0 = \bigcup_{m \in \mathbb{N}} \mathsf{AC}^0[m]$ with m here representing the modulo, we require that each $\mathcal{C}_2$ subcircuit of a circuit D from $\mathcal{C}_1 \circ \mathcal{C}_2$ uses the same fixed m.

trivial SAT algorithm for $\mathsf{AND}_3 \circ \mathscr{C}$ or a non-trivial CAPP algorithm for $\mathsf{OR}_{\mathrm{poly}(n)} \circ \mathsf{AND}_3 \circ \mathscr{C}$ (*i.e.*, a 3-DNF of $\mathscr{C}$ subcircuits).

Proving lower bounds for complexity classes weaker than NEXP. The most significant drawback of the separation $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ was that NEXP is a much larger class than our ultimate goal NP (previous lower bounds for AC^0 or $\mathsf{AC}^0[p]$ usually work for functions in P). In 2013, by refining the connection between circuit analysis algorithms and circuit lower bounds, Williams [Wil13b] proved that $(\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$ does not have polynomial-size ACC^0 circuits. In 2018, in an exciting new breakthrough, Murray and Williams [MW18] proved that $\mathsf{NQP} := \mathsf{NTIME}[2^{\mathrm{polylog}(n)}]$ does not have polynomial-size ACC^0 circuits.[12]

Indeed, similar to Theorem 1.1.1, [Wil13b] proved that $(\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1} \not\subseteq \mathscr{C}$ follows from the same algorithms stated in the hypothesis of Theorem 1.1.1, and [MW18] generalized Theorem 1.1.1 to NQP or NP lower bounds.

Theorem 1.1.2 ([MW18], Informal). *Let $\mathscr{C}$ be a reasonable circuit class.*[13] *For any constant $\varepsilon \in (0,1)$, the following hold:*

- *If there is a $2^{n-n^{\varepsilon}}$-time algorithm for* SAT *(or* CAPP*) of $2^{n^{\varepsilon}}$-size n-input $\mathscr{C}$ circuits, then* $\mathsf{NQP} \not\subseteq \mathscr{C}$.
- *If there is a $2^{n-\varepsilon n}$-time algorithm for* SAT *(or* CAPP*) of $2^{\varepsilon n}$-size n-input $\mathscr{C}$ circuits, then* NP *does not admit n^k-size $\mathscr{C}$ circuits for every $k \in \mathbb{N}$.*

Proving stronger circuit lower bounds. Given the generic connection established in [Wil10, Wil11], another line of work aimed to prove stronger lower bounds following this paradigm. In 2014, by designing a non-trivial algorithm analyzing the circuit class $\mathsf{ACC}^0 \circ \mathsf{THR}$ and applying Theorem 1.1.1, Williams [Wil14a] proved that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0 \circ \mathsf{THR}$. In 2016, by refining the circuit-analysis algorithm from [Wil11], Chen and Papakonstantinou [CP16] improved the dependence on depth by showing that NEXP does not have ACC^0 circuits of $o(\log n / \log\log n)$ depth.

Also in 2016, Alman, Chan, Williams, and independently Tamaki [ACW16, Tam16] proved that E^{NP} (class of languages computable by a $2^{O(n)}$-time algorithm with a SAT oracle) does not have $n^{2-o(1)}$ size $\mathsf{THR} \circ \mathsf{THR}$ circuits by designing non-trivial circuit-analysis algorithm for $\mathsf{THR} \circ \mathsf{THR}$ circuits.[14] In 2018, by giving a new circuit-analysis algorithm for $\mathsf{AC}^0[\oplus]$, Rajgopal, Santhanam,

[12] The journal versions of [Wil13b] and [MW18] are [Wil16a] and [MW20], respectively. Note that there is another notion of NQP [ADH97] (Nondeterministic Quantum Polynomial-Time) in the literature based on quantum complexity, but it turns out to equal $\mathsf{coC}_{=}\mathsf{P}$ [FGHP99].

[13] Using [BV14], we only require $\mathsf{AND}_3 \circ \mathscr{C} \subseteq \mathscr{C}$ for SAT and $\mathsf{OR}_{\mathrm{poly}(n)} \circ \mathsf{AND}_3 \circ \mathscr{C} \subseteq \mathscr{C}$ for CAPP. See Theorem 4.1.6 for a formal statement.

[14] [ACW16] indeed proved a stronger lower bound that E^{NP} does not have $2^{n^{o(1)}}$-size $\mathsf{ACC}^0 \circ \mathsf{THR} \circ \mathsf{THR}$ circuits whose bottom layer has at most $n^{2-\Omega(1)}$ THR gates.

and Srinivasan [RSS18] proved that E^{NP} does not have $2^{o(n^{1/(d+1)})}$-size depth-d $\mathsf{AC}^0[\oplus]$ circuits for every $d \in \mathbb{N}$, improving quantitatively on the $2^{\Omega(n^{1/(2d-1)})}$-size lower bound by Razborov and Smolensky [Raz87, Smo87] for large d. In 2017, building on [Wil13b], Chen, Oliveira, and Santhanam [COS18] proved that NEXP cannot be $1/2 + 1/\mathrm{polylog}(n)$-approximated by polynomial-size ACC^0 circuits.

1.2 Motivation

1.2.1 Further Tightening the Connection

Recall that [BV14] proved that circuit lower bounds for $\mathscr{C}$ follows from non-trivial derandomization of 3-DNF $\circ \mathscr{C}$. This is still not ideal for proving lower bounds against fixed-constant-depth circuit classes such as THR $\circ$ THR: ideally, we would like to show derandomization of THR $\circ$ THR itself implies circuit lower bounds for THR $\circ$ THR (solving CAPP for 3-DNF $\circ$ THR $\circ$ THR might be much harder than solving CAPP for THR $\circ$ THR). This motivates the following question.

Open Problem 1. *Can we further tighten the connection between circuit-analysis algorithms and circuit lower bounds? For example, can we prove* NEXP $\not\subseteq \mathscr{C}$ *(or* NQP $\not\subseteq \mathscr{C}$*) follow from non-trivial derandomization of* $\mathsf{AND}_3 \circ \mathscr{C}$ *(or even weaker classes)? In particular, does* NEXP $\not\subset$ THR $\circ$ THR *follow from non-trivial derandomization of* THR $\circ$ THR*?*

1.2.2 Proving Almost-Everywhere Lower Bounds via Algorithmic Method

Another subtle but important shortcoming of the algorithmic method is that it only achieves infinitely-often separations. For example, [MW18] proved there is an NQP function f such that, for every polynomial-size ACC^0 circuit family $\{C_n\}$, *there are infinitely many input lengths n such that C_n fails to compute f on n-bit inputs*. This certainly implies the separation NQP $\not\subset \mathsf{ACC}^0$, but it could be the case that *for nearly every input length* NQP is easy for ACC^0, and NQP is only hard on extremely rare input lengths n, *e.g.*, $n = 2^{2^{2^k}}$ for $k \in \mathbb{N}$. In a case where the hard input lengths are so far apart, *practically* the situation is not very different from NQP $\subset \mathsf{ACC}^0$. Hence, we would like to have *almost-everywhere separations*: we want a function $f = \{f_n : \{0,1\}^n \to \{0,1\}\}$ so that for all sufficiently large input lengths n, f_n cannot be computed by any ACC^0 circuit (in notation, we would say $f \notin$ i.o.-ACC^0). Unfortunately, it has remained open whether E^{NP} is contained in ACC^0 *infinitely often*.

Open Problem 2. *Can we prove* $\mathsf{E}^{\mathsf{NP}} \not\subset$ *i.o.-*ACC^0*? More generally, can we prove that* $\mathsf{E}^{\mathsf{NP}} \not\subset$ *i.o.-*$\mathscr{C}$ *follows from non-trivial circuit analysis algorithms for* $\mathscr{C}$ *circuits?*

The infinitely-often barrier in complexity theory. Most previous lower bounds for AC^0 and $\mathsf{AC}^0[p]$ are almost-everywhere: they show $f \notin$ i.o.-AC^0 or $f \notin$ i.o.-$\mathsf{AC}^0[p]$ for some f. Indeed, most combinatorial/algebraic lower bound approaches argue hardness for each input length separately, so they naturally give lower bounds for all input lengths. However, in structural complexity theory, arguments often involve different input lengths simultaneously, and it is common that in some settings almost-everywhere separations are much harder to achieve than corresponding infinitely-often separations. Two classical examples include:

- (An almost-everywhere NTIME hierarchy theorem is Open.) It is known that $\mathsf{NTIME}[2^n] \not\subset \mathsf{NTIME}[2^n/n]$ [SFM78, Žák83], but it is open whether $\mathsf{NTIME}[2^n] \subset$ i.o.-$\mathsf{NTIME}[n \log n]$. (Indeed, there is an oracle O such that $\mathsf{NEXP}^O \subset$ i.o.-NP^O [BFS09].)
- (An almost-everywhere super-linear circuit lower bound for $\mathsf{MATIME}[2^n]$ is open.) It is known that $\mathsf{MA}_{/1} \not\subset \mathsf{SIZE}(n^k)$ for all k [San09] and $\mathsf{MATIME}[2^n] \not\subset \mathsf{P}_{/\mathsf{poly}}$ [BFT98], but it is open whether $\mathsf{MATIME}[2^n] \subset$ i.o.-$\mathsf{SIZE}(O(n))$. (Indeed, it is even open whether $\Sigma_2\mathsf{TIME}[2^n] \subset$ i.o.-$\mathsf{SIZE}(O(n))$).

1.2.3 Proving Strongly Average-Case Lower Bounds via Algorithmic Method

Williams' lower bound $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$ and its later improvement to $\mathsf{NQP} \not\subseteq \mathsf{ACC}^0$ only yield worst-case lower bounds, while prior lower bounds against AC^0 or $\mathsf{AC}^0[p]$ can often be adapted to hold in the average-case setting (*e.g.*, [HRST17]). Average-case lower bounds are more interesting since they tend to have more applications than worst-case lower bounds, such as constructing unconditional PRGs. [COS18] proved that NEXP cannot be $1/2 + 1/\mathrm{polylog}(n)$-approximated by ACC^0 circuits.[15] But the $(1/2 + 1/\mathrm{polylog}(n))$-inapproximability result is not enough to get us a non-trivial (say, with $n^{o(1)}$ seed length) PRG construction for ACC^0, which requires at least a $(1/2 + n^{-\omega(1)})$-inapproximability bound. This raised the following open question.

Open Problem 3. *Can we prove that* NEXP *(or even* NQP*) cannot be* $(1/2 + n^{-\omega(1)})$*-approximated by polynomial-size* ACC^0 *circuits? Or more generally, can we strengthen the lower bound consequences from Theorem 1.1.1 and Theorem 1.1.2 to strongly average-case lower bounds?*

[15] We say that a function f cannot be $(1/2+\varepsilon)$-approximated by a circuit C, if $f(x) = C(x)$ for at most a $(1/2+\varepsilon)$ fraction of inputs x from $\{0,1\}^n$. See Section 3.1.1 for a formal definition.

The $(1/2+1/\sqrt{n})$ Razborov-Smolensky barrier. Indeed, proving a $(1/2+n^{-\omega(1)})$-inapproximability result is even open for $\mathsf{AC}^0[\oplus]$ circuits (AC^0 circuits extended with parity gates). [Raz87, Smo87, Smo93] showed that the majority function cannot be $(1/2+n^{1/2-\varepsilon})$-approximated by $\mathsf{AC}^0[\oplus]$ using the renowned polynomial approximation method. However, it is even open whether E^{NP} can be $(1/2+1/\sqrt{n})$-approximated by $\log n$-degree $\mathbb{F}_2$-polynomials.[16] Improving the $(1/2+1/\sqrt{n})$-bound (and constructing the corresponding PRGs) is recognized as a significant open question in circuit complexity [Vio09a, Vad12, FSUV13, CHLT19].

Explicit construction of rigid matrices. A matrix is rigid if it has a high hamming distance from all low-rank matrices. It is a long-standing open question [Val77, Lok09] to find explicit constructions of rigid matrices with strong enough parameters. Note that the question of constructing rigid matrices can indeed be interpreted as proving average-case lower bounds against low-rank matrices. Hence, another question is whether algorithmic method can be applied to this problem as well.

Open Problem 4. *Can we apply algorithmic method to obtain interesting construction of rigid matrices?*

1.3 Our Contributions: Bird's Eye View

In this thesis, motivated by the above open questions, we further develop Williams' algorithmic method and prove many new circuit lower bounds. In the rest of this chapter, we first give an overview of our contributions and then discuss the details of our results.

1. (**Tighter connections between circuit-analysis algorithms and lower bounds**) We prove that lower bounds against $\mathscr{C}$ follow from non-trivial derandomization of $\mathsf{AND}_2 \circ \mathscr{C}$ or $\oplus_2 \circ \mathscr{C}$, which in particular shows non-trivial derandomization of $\mathsf{THR} \circ \mathsf{THR}$ would imply lower bounds for $\mathsf{THR} \circ \mathsf{THR}$; see Section 1.4 for details.

   Using structural insight into $\mathsf{THR} \circ \mathsf{THR}$ circuits and ideas from our strongly average-case lower bounds, we further prove that lower bounds for $\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ (which is stronger than $\mathsf{THR} \circ \mathsf{THR}$) would follow from non-trivial derandomization of $\mathsf{MAJ} \circ \mathsf{MAJ}$ (Theorem 1.6.6).
2. (**Almost-everywhere lower bounds and quasi-polynomial-time derandomization via algorithmic method**) We prove that $\mathsf{E}^{\mathsf{NP}} \not\subset$ i.o.-$\mathscr{C}$ follow from non-trivial circuit analysis algorithms for $\mathscr{C}$ circuits (and in particular, $\mathsf{E}^{\mathsf{NP}} \not\subset$ i.o.-ACC^0), answering Open Problem 2 in the

[16] Note that $\log n$-degree $\mathbb{F}_2$-polynomials is a special case of depth-2 $n^{\log n}$-size $\mathsf{AC}^0[\oplus]$ circuits.

affirmative. We also prove that strongly average-case almost-everywhere lower bounds for E^{NP} follow from non-trivial circuit analysis algorithm, which in particular implies that there is a function $f \in \mathsf{E}^{\mathsf{NP}}$ that cannot be $(1/2 + 1/2^{n^{o(1)}})$-approximated by $2^{n^{o(1)}}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, on all sufficiently large input lengths.

Utilizing the connection between strongly average-case lower bounds and pseudorandom generators (PRGs), we also give a *nondeterministic* PRG with $\mathrm{polylog}(n)$ seed length fooling polynomial-size ACC^0 circuits. This, in particular, gives a quasi-polynomial time derandomization of Merlin Arthur protocols with an ACC^0 verifier. See Section 1.5 for details.

3. (**New average-case lower bounds for nondeterministic time classes**) We generalize the non-trivial-algorithm-to-circuit-lower-bound connections of Theorem 1.1.1 and Theorem 1.1.2 from worst-case lower bounds to average-case lower bounds, thereby answering Open Problem 3 in the affirmative. In particular, we prove that NQP cannot be $(1/2 + n^{-\omega(1)})$-approximated by polynomial-size ACC^0. See Section 1.6 for details.
4. (**New construction of rigid matrices**) Utilizing non-trivial algorithms for analyzing low-rank matrices and following the algorithmic method, we give a "semi-explicit" (P^{NP}) construction of rigid matrices in a particular regime of parameters, for which no non-trivial constructions were known before. This can be seen as an affirmative answer to Open Problem 4. We also prove strong correlation bounds against $\mathbb{F}_2$-polynomials. See Section 1.7 for details.

Notation. We use $\mathbb{N}$ to denote all non-negative integers and $\mathbb{N}_{\geq 1}$ to denote all positive integers. We say a circuit class $\mathscr{C}$ is **concrete** if we can talk about the $\mathscr{C}$ complexity of a single function $f\colon \{0,1\}^n \to \{0,1\}$ (as opposed to a family of functions $\{f_n\}_{n \in \mathbb{N}_{\geq 1}}$). For example, for fixed $d, m \in \mathbb{N}_{\geq 1}$, $\mathsf{AC}^0_d[m]$ is a concrete circuit class, but AC^0 is not (because the depth can vary). We say a concrete circuit class $\mathscr{C}$ is *typical* if it is closed under taking the negation of the output and projections of the input. See Section 3.1.1 for more formal definitions.

1.4 Tighter Connections between Lower Bounds and Non-trivial Algorithms

We begin by describing our results on obtaining tighter connections between circuit-analysis algorithms and lower bounds. To formally discuss our results, we introduce the Gap-UNSAT problem

and a variant of the CAPP problem below.

1. Gap-UNSAT **(GAP UNSATISFIABILITY) with gap δ (denoted as** Gap-UNSAT$_\delta$**):** Given a circuit C, distinguish between the (Yes) case that C is unsatisfiable and the (No) case that C has at least $\delta \cdot 2^n$ satisfying assignments.[17]
2. CAPP **with inverse-circuit-size error (denoted as** $\widetilde{\text{CAPP}}$**)**: Given a circuit C of size S on n input bits, estimate $\Pr_{x\in\{0,1\}^n}[C(x) = 1]$ within an additive error $1/S$.

We give tighter connections between non-trivial circuit-analysis algorithms for $\mathscr{C}$ and circuit lower bounds against $\mathscr{C}$. We show that, lower bounds against $\mathscr{C}$ follows from non-trivial derandomization algorithms for either $\mathsf{AND}_3 \circ \mathscr{C}$, $\mathsf{OR}_2 \circ \mathscr{C}$, or $\oplus_2 \circ \mathscr{C}$ (a.k.a. $\mathsf{XOR}_2 \circ \mathscr{C}$).[18]

Theorem 1.4.1 (NEXP lower bounds from non-trivial Gap-UNSAT or CAPP algorithms)**.** *There is an absolute constant $\delta \in (0,1)$, such that for every typical and concrete $\mathscr{C}$, if either:*

- *there is a $2^n/n^{\omega(1)}$-time* Gap-UNSAT$_\delta$ *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{AND}_3 \circ \mathscr{C}$ *circuits, or*
- *there is a $2^n/n^{\omega(1)}$-time* CAPP$_\delta$ *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{OR}_2 \circ \mathscr{C}$, $\oplus_2 \circ \mathscr{C}$*, or* $\mathsf{AND}_2 \circ \mathscr{C}$ *circuits,*

then $\mathsf{NEXP} \not\subset \mathscr{C}$*. Moreover, in the second bullet, $\mathscr{C}$ does not need to be closed under negation.*

Comparison with [BV14]. As mentioned in Section 1.1.3, [BV14] showed that non-trivial Gap-UNSAT algorithms for $\mathsf{OR}_{\mathrm{poly}(n)} \circ \mathsf{AND}_3 \circ \mathscr{C}$, or non-trivial SAT algorithms for $\mathsf{AND}_3 \circ \mathscr{C}$ would imply $\mathsf{NEXP} \not\subset \mathscr{C}$. Theorem 1.4.1 strengthens these two connections, as Gap-UNSAT is an easier problem than SAT. In particular, we avoid the unbounded fan-in OR entirely.

Applying recent results of [MW18], we can naturally generalize to circuit lower bounds for NP if faster algorithms are assumed.

Theorem 1.4.2 (NP lower bounds from faster Gap-UNSAT or CAPP algorithms)**.** *There is an absolute constant $\delta \in (0,1)$, such that for any typical and concrete $\mathscr{C}$, if there is a constant $\varepsilon \in (0,1)$ such that one of the following holds:*

- Gap-UNSAT$_\delta$ *for $2^{\varepsilon n}$-size* $\mathsf{AND}_3 \circ \mathscr{C}$ *circuits can be solved in $2^{n-\varepsilon n}$ time, or*
- CAPP$_\delta$ *for $2^{\varepsilon n}$-size* $\mathsf{OR}_2 \circ \mathscr{C}$, $\oplus_2 \circ \mathscr{C}$*, or* $\mathsf{AND}_2 \circ \mathscr{C}$ *circuits can be solved in $2^{n-\varepsilon n}$ time,*

then for every k there is a function in NP *that does not have n^k-size $\mathscr{C}$ circuits. Moreover, in the second bullet, $\mathscr{C}$ does not need to be closed under negation.*

[17] Note that Gap-UNSAT$_\delta$ is easier than both SAT and CAPP$_\delta$. When not explicitly stated, δ is set to be $1/2$ by default.

[18] We use $\mathsf{AND}_k \circ \mathscr{C}$ to denote an AND of k $\mathscr{C}$ subcircuits (similarly for $\mathsf{OR}_k \circ \mathscr{C}$ or $\oplus_k \circ \mathscr{C}$). The size of such a circuit is defined to be the sum of the sizes of all $\mathscr{C}$ subcircuits plus 1.

Remark 1.4.3. *In Theorem 1.4.1 and Theorem 1.4.2, the algorithm can even be nondeterministic, as long as on all computation paths, the algorithm either outputs "don't know" or a correct answer (and a correct answer is on at least one path).*[19]

1.4.1 Our Techniques: PCPs of Proximity

We remark that in terms of techniques, our approach is very different from that of [BV14]. [BV14] constructed a highly efficient PCP for $\mathsf{NTIME}[T(n)]$, with the queries being *projections* of random bits, and the verifier being a 3-CNF. Then they obtained their results by directly plugging this PCP construction into the original approach of [Wil10].

Our approach is more indirect. The key insight in proving Theorem 1.4.1 (and Theorem 1.4.2 similarly) is to use *probabilistically checkable proofs of proximity* (PCPs of proximity) to reduce circuit evaluation tasks to derandomization tasks. Using efficient PCPs for $\mathsf{NTIME}[2^n]$ [BGH+05], [Wil10] showed $\mathsf{NEXP} \not\subset \mathsf{P/poly}$ follows from non-trivial Gap-UNSAT algorithms for $\mathrm{poly}(n)$-size general circuits. Applying a PCP of proximity to the *Circuit Evaluation Problem*, we design a Gap-UNSAT algorithm for general circuits, only assuming $\mathsf{NEXP} \subset \mathscr{C}$ and a Gap-UNSAT algorithm for $\mathsf{AND}_3 \circ \mathscr{C}$, which results in a contradiction when $\mathscr{C} \subset \mathsf{P/poly}$. Therefore our overall argument applies PCPs in two different ways: once on a nondeterministic 2^n-time computation, and once on a $\mathrm{poly}(n)$-size circuit evaluation. See Section 5.1 for an overview of the whole argument.

1.4.2 Applications: Potential Approaches to Lower Bounds against $\mathsf{THR} \circ \mathsf{THR}$

As a direct corollary of Theorem 1.4.1 and the folklore result that an XOR of two $\mathsf{THR} \circ \mathsf{THR}$ can be written as a polynomially larger $\mathsf{THR} \circ \mathsf{THR}$,[20] it is immediate that non-trivial CAPP algorithms for $\mathsf{THR} \circ \mathsf{THR}$ circuits with small constant error would imply $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$. With some additional work, the same can be shown for non-trivial SAT algorithms for $\mathsf{THR} \circ \mathsf{THR}$ circuits.

Theorem 1.4.4. *There is an absolute constant $\delta \in (0,1)$, such that if CAPP_δ for $\mathrm{poly}(n)$-size $\mathsf{THR} \circ \mathsf{THR}$ circuits can be solved in $2^n/n^{\omega(1)}$ time, then $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$. The same holds with SAT in place of CAPP_δ.*

It still looks like a tough challenge to obtain a non-trivial derandomization of $\mathsf{THR} \circ \mathsf{THR}$ circuits, as usually derandomization comes from circuit lower bounds (ironically, the goal here is to

[19] For the CAPP_δ algorithm, an answer is correct if it is within δ of the exact acceptance probability. The answers of different computation paths do not need to be the same: we only require all of them to be within δ of the exact acceptance probability.

[20] see, *e.g.*, Lemma 5.7.4 for a proof

prove circuit lower bounds for $\mathsf{THR} \circ \mathsf{THR}$!). Utilizing some structural insights into $\mathsf{THR} \circ \mathsf{THR}$, we show it indeed suffices to obtain non-trivial derandomization for $\mathsf{THR} \circ \mathsf{MAJ}$ or $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits, for which we already know $2^{\Omega(n)}$-size lower bounds [HMP+93, FKL+01].

Theorem 1.4.5. *There is an absolute constant $\delta \in (0,1)$ such that if one of the following holds:*

1. CAPP_δ *(or* SAT*) for* $\mathrm{poly}(n)$*-size* $\mathsf{THR} \circ \mathsf{MAJ}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time, or*
2. $\widetilde{\mathsf{CAPP}}$ *for* $\mathrm{poly}(n)$*-size* $\mathsf{MAJ} \circ \mathsf{MAJ}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time.*

Then $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$.

Therefore a proof of $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$ follows, if we can "mine" the known $2^{\Omega(n)}$-size lower bounds for $\mathsf{THR} \circ \mathsf{MAJ}$ or $\mathsf{MAJ} \circ \mathsf{MAJ}$ and design appropriate circuit-analysis algorithms![21]

1.4.3 Applications: Lower Bounds against Approximate Linear Sums

As another application, we initiate the study of lower bounds against approximate linear sums of various circuit classes $\mathscr{C}$. We first introduce the definition of $\mathsf{Sum} \circ \mathscr{C}$ and $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ circuits.

Definition 1.4.6. *Let $\mathscr{C}$ be concrete and typical.* $\mathsf{Sum} \circ \mathscr{C}$ *denotes the following set of circuits: every $L \in \mathsf{Sum} \circ \mathscr{C}$ can be described as $L(x) = \sum_{i \in [m]} \alpha_i \cdot C_i(x)$ where each $\alpha_i \in \mathbb{R}$ and each $C_i(x) \colon \{0,1\}^n \to \{0,1\}$ is a $\mathscr{C}$ circuit. The size of L, denoted as $\mathsf{SIZE}(L)$, is defined as the total size of each C_i. (i.e., $\mathsf{SIZE}(L) = \sum_i \mathsf{SIZE}(C_i)$.)*

Let $f \colon \{0,1\}^n \to \{0,1\}$ be a function and $L \colon \{0,1\}^n \to \mathbb{R}$ be a $\mathsf{Sum} \circ \mathscr{C}$ circuit. We say L is a $\mathsf{Sum} \circ \mathscr{C}$ circuit for f, if $L(x) = f(x)$ for every $x \in \{0,1\}^n$. And we say L is a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit for f, if $|f(x) - L(x)| \leq \delta$ for every $x \in \{0,1\}^n$.

In [Wil18], Williams initiated the study of $\mathsf{Sum} \circ \mathscr{C}$ circuits and proved several lower bounds against them. In particular, it was proved that:

Theorem 1.4.7 ([Wil18])**.** *For every $d, m \in \mathbb{N}_{\geq 1}$, there is a $\beta \in \mathbb{N}_{\geq 1}$ and an $f \in \mathsf{NTIME}[2^{\log^\beta n}]$ such that f does not have $\mathsf{Sum} \circ \mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuits of size n^a, for every $a \in \mathbb{N}_{\geq 1}$.*

Using PCP of proximity, we can also strengthen Theorem 1.4.7 to hold against approximate linear sums. Formally, we have:

Theorem 1.4.8. *For every $d, m \in \mathbb{N}_{\geq 1}$ and $\delta \in [0, 0.5)$, there is a $\beta \in \mathbb{N}_{\geq 1}$ and an $f \in \mathsf{NTIME}[2^{\log^\beta n}]$ such that f does not have $\mathsf{Sum}_\delta \circ \mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuits of size n^a, for every $a \in \mathbb{N}_{\geq 1}$.*

[21] Part of Theorem 1.4.5 is improved by our later theorem Theorem 1.6.6.

1.4.4 Notes and Bibliographic Details

This subsection's results are from [CW19], which appeared in CCC 2019. In retrospect, [CW19] made two significant conceptual contributions to the algorithmic method that most of the subsequent works build on. First, it introduced the techniques of PCP of proximity into the algorithmic method, which is proven to be very useful by many subsequent works [AC19, VW20, CR20, CLW20, CL21, RSW22]. Second, it initialized the study of approximate linear sums of circuits in the context of the algorithmic method, which turned out to be the crucial concept for proving average-case lower bounds in subsequent works [CR20, CLW20, CL21, HV21, CLLO21].

We remark that [CW19] indeed proved a generic connection between #SAT algorithms for $\mathscr{C}$ and lower bounds against $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ and obtained several other lower bounds against $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ for various $\mathscr{C}$. We do not discuss them in this thesis since they are not related to other subsequent circuit lower bounds covered by this thesis; we refer interested readers to [CW19] for details. We also note that Theorem 1.4.8 was later improved by a subsequent work [CLLO21] (see its Corollary 4), so we omit its proof in this thesis.

1.5 Almost-everywhere Circuit Lower Bounds and Average-case Witness Lower Bounds from Non-trivial Derandomization

Next, we discuss our results on proving almost-everywhere circuit lower bounds via the algorithmic method.

1.5.1 Almost-everywhere Circuit Lower Bounds

Our first result is that non-trivial derandomization for a circuit class $\mathscr{C}$ implies almost-everywhere $\mathscr{C}$-circuit lower bounds for E^{NP}. For a language $L\colon \{0,1\}^* \to \{0,1\}$, we use $L_n\colon \{0,1\}^n \to \{0,1\}$ to denote its restriction to n-bit inputs.

Theorem 1.5.1. *Let $\mathscr{C}$ be a typical concrete circuit class and $S(n) \leq 2^{o(n)}$ be a non-decreasing size parameter. There is a universal constant $K \in \mathbb{N}_{\geq 1}$ such that if CAPP for $(n^K \cdot S(n))$-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ time, then there is an $L \in \mathsf{E}^{\mathsf{NP}}$ such that L_n does not have $\mathscr{C}$ circuits of size $S(n/2)$ for all sufficiently large $n \in \mathbb{N}_{\geq 1}$.*

An extension to average-case lower bounds. Combining PCPs of Proximity and a new XOR Lemma (see Section 1.5.3), we can extend Theorem 1.5.1 to prove strongly average-case lower

bounds. We say that a function $f\colon \{0,1\}^n \to \{0,1\}$ *cannot be* $(1/2+\varepsilon)$*-approximated* by circuits of type $\mathscr{C}$, if every circuit from $\mathscr{C}$ computes f correctly on less than $(1/2+\varepsilon)2^n$ of the n-bit inputs.

Theorem 1.5.2. *Let $\mathscr{C}$ be a typical and concrete circuit class. Suppose there is an $\varepsilon \in (0,1)$ such that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a language $L \in \mathsf{E}^{\mathsf{NP}}$ and a constant $\delta \in (0,1)$ such that, for every sufficiently large $n \in \mathbb{N}_{\geq 1}$, L_n cannot be $(1/2+2^{-n^\delta})$-approximated by $\mathscr{C}$ circuits of size 2^{n^δ}.*

Almost-everywhere strongly average-case sub-exponential lower bounds for $\mathsf{ACC}^0 \circ \mathsf{THR}$. Recall that for a given $d, m \in \mathbb{N}$, $\mathsf{AC}^0_d[m]$ is the class of circuit families of depth d, with unbounded fan-in $\mathsf{AND}, \mathsf{OR}, \mathsf{MOD}_m$ gates and fan-in 1 NOT gates, and $\mathsf{ACC}^0 := \bigcup_{d,m} \mathsf{AC}^0_d[m]$. Combining Theorem 1.5.2 and the corresponding #SAT algorithm from [Wil14a] for $\mathsf{ACC}^0 \circ \mathsf{THR}$,[22] we immediately obtain the following corollary.

Corollary 1.5.3. *For every $d, m \in \mathbb{N}_{\geq 1}$, there is an $\varepsilon \in (0,1)$ and a language $L \in \mathsf{E}^{\mathsf{NP}}$ such that L_n cannot be $(1/2+2^{-n^\varepsilon})$-approximated by $\mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuits of 2^{n^ε} size, for every sufficiently large $n \in \mathbb{N}_{\geq 1}$.*[23]

Applying the known connection between average-case hardness and construction of PRGs [NW94], we obtain the following E^{NP}-computable PRG as a direct consequence of our average-case lower bound for ACC^0.

Theorem 1.5.4. *For every $d, m \in \mathbb{N}_{\geq 1}$, there is $\delta = \delta(d,m) \in (0,1)$ and an E^{NP}-computable PRG which takes an n-bit seed and outputs a 2^{n^δ}-bit string and fools $\mathsf{AC}^0_d[m]$ circuits of size 2^{n^δ}.*

Comparison with previous work. Corollary 1.5.3 compares favorably with prior circuit lower bounds for problems in E^{NP}. Williams [Wil11, Wil14a] proved that E^{NP} cannot be worst-case computed by $2^{n^{o(1)}}$ size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. Following the work of Rajgopal, Santhanam and Srinivasan [RSS18], Viola [Vio20] recently proved E^{NP} cannot be $(1/2+1/n^{1-\varepsilon})$-approximated by $2^{n^{o(1)}}$-size $\mathsf{AC}^0[\oplus]$ circuits. All of these lower bounds are only infinitely-often separations, and yield strictly weaker average-case lower bounds than Corollary 1.5.3.

We also remark that [FS17] devised a notion of "significant separation", which is stronger

[22] Formally, [Wil14a] proved that for every $d, m \in \mathbb{N}_{\geq 1}$, there is an $\varepsilon \in (0,1)$ such that #SAT for 2^{n^ε}-size $\mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuits can be solved in deterministic 2^{n-n^ε} time.

[23] Note that ε depends on the values of d and m.

than infinitely-often separation while weaker than almost-everywhere separation.[24] They showed a significant separation of NEXP and ACC^0. This is incomparable with an almost-everywhere separation of E^{NP} against ACC^0.

1.5.2 Strongly Average-case Witness Lower Bounds and Derandomization of $\mathsf{MA}_{\mathscr{C}}$

Our new proof techniques can also be used to prove witness lower bounds for NE. We say that NE *does not admit $s(n)$-size $\mathscr{C}$ witnesses*, if there exists a verifier $V(x,y)$ that takes input $|x| = n$, $|y| = 2^n$, and runs in $2^{O(n)}$ time, such that for infinitely many $x \in \{0,1\}^*$: (1) $V(x,\cdot)$ accepts some $y \in \{0,1\}^{2^{|x|}}$ (*i.e.*, $V(x,y) = 1$) and (2) for every $y \in \{0,1\}^{2^{|x|}}$ accepted by $V(x,\cdot)$, $\mathsf{func}(y)$ has no $s(n)$-size $\mathscr{C}$ circuit.[25]

Moreover, we say that *unary* NE *does not admit $s(n)$-size $\mathscr{C}$ witnesses*, if for some verifier V and for infinitely many $n \in \mathbb{N}_{\geq 1}$, the above two conditions hold for $x = 1^n$. Clearly, this implies that NE does not admit $s(n)$-size $\mathscr{C}$ witnesses.

Williams [Wil13b] proved that for every $d, m \in \mathbb{N}_{\geq 1}$, there exists an $\varepsilon \in (0,1)$ such that NE does not admit 2^{n^ε}-size $\mathsf{AC}^0_d[m]$ witnesses.[26]

We generalize the above ACC^0 witness lower bounds to average-case as well. We say that *unary* NE *does not admit $(1/2 + \varepsilon(n))$-approximate $s(n)$-size $\mathscr{C}$ witnesses*, if in Condition (2) above, $V(1^n, y) = 1$ implies that $\mathsf{func}(y)$ cannot be $(1/2 + \varepsilon(n))$-approximated by $s(n)$-size $\mathscr{C}$ circuits.

Theorem 1.5.5 (Average-case witness lower bounds from non-trivial $\widetilde{\mathsf{CAPP}}$ algorithms)**.** *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ such that unary* NE *does not admit $(1/2 + 2^{-n^\delta})$-approximate 2^{n^δ}-size $\mathscr{C}$ witnesses.*

As a direct corollary of Theorem 1.5.5 and the non-trivial #SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$ [Wil14a], it follows that for every $d, m \in \mathbb{N}_{\geq 1}$, there is a $\delta \in (0,1)$ such that unary NE does not admit $(1/2 + 2^{-n^\delta})$-approximate 2^{n^δ}-size $\mathscr{C}$ witnesses.

Derandomization of $\mathsf{MA}_{\mathscr{C}}$ via NPRGs. Nondeterministic pseudorandom generators (NPRGs) are a weaker variant of pseudorandom generators (PRGs), which still allow for derandomization

[24] Roughly speaking, "significant separation" means that when the separation holds for an input length n, there is another input length at most polynomially larger than n such that the separation also holds. That is, the hardness cannot be very "sparsely distributed".

[25] We use $\mathsf{func}(y)$ to denote the the $|x|$-bit function whose truth-table is exactly y. See also Section 3.1.1.

[26] It is not explicitly stated in [Wil13b], but it is not hard to verify that its proof indeed implies a witness lower bound for unary NE. See Theorem 4.4.3 for a general statement, which is proved in [Che19].

of Merlin-Arthur protocols. Roughly speaking, an NPRG allows one to *guess and verify* a hard truth-table y (instead of producing one from scratch) and then use y to construct a PRG; see Section 3.5. Let $\mathscr{C}$ be concrete and typical. We use $\mathsf{MA}_{\mathscr{C}}$ to denote the sub-class of Merlin-Arthur protocols whose verification can be simulated by polynomial-size $\mathscr{C}$ circuits for every possible witness; see Definition 3.4.3.[27]

Indeed, Theorem 1.5.5 immediately implies a natural NPRG construction: guess a witness y and verify that $V(1^n, y) = 1$; then we know f is strongly average-case hard against $\mathscr{C}$, and we can apply the well-known Nisan-Widgerson PRG construction to obtain a PRG for $\mathscr{C}$. Formally, we have (the extra AC^0_2 at the bottom comes from the overhead of the NW reconstruction):

Theorem 1.5.6. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ and an infinity often nondeterministic PRG for 2^{n^δ}-size $\mathscr{C}$ circuits with error 2^{-n^δ}, seed-length $\mathrm{poly}(n)$, and $2^{\mathrm{poly}(n)}$ running time. Consequently, $\mathsf{MA}_{\mathscr{C}} \subseteq$ i.o.-$\mathsf{NTIME}[2^{\log^\beta n}]$ for some $\beta \in \mathbb{N}_{\geq 1}$.*

The following is an immediate corollary of Theorem 1.5.6 and the non-trivial #SAT algorithms for ACC^0 [Wil11, Wil14a].

Corollary 1.5.7. *For every $d, m \in \mathbb{N}_{\geq 1}$, there exists a $\delta \in (0,1)$ such that there is an i.o. NPRG for 2^{n^δ}-size $\mathsf{AC}^0_d[m]$ circuits with error 2^{-n^δ}, seed-length $\mathrm{poly}(n)$, and $2^{\mathrm{poly}(n)}$ running time. Consequently, $\mathsf{MA}_{\mathsf{AC}^0_d[m]} \subseteq$ i.o.-$\mathsf{NTIME}[2^{\log^\beta n}]$ for some $\beta \in \mathbb{N}_{\geq 1}$.*

1.5.3 Two Technical Tools

To achieve our almost-everywhere strongly average-case lower bounds, we develop two new technical tools. The first is a "constructive" proof of the almost-everywhere sublinear witness NTIME hierarchy of Fortnow and Santhanam [FS16]. Our new proof builds a P^{NP} algorithm that can explicitly find inputs on which the weak algorithms make mistakes. The second is an XOR Lemma based on computations by approximate linear sums. We believe both results are interesting in their own right and will likely have other applications in computational complexity. In the following, we state both of them informally.

[27] See Lemma 3.5.1 for how to use NPRGs for $\mathscr{C}$ to derandomize a $\mathsf{MA}_{\mathscr{C}}$ protocol into a nondeterministic protocol. Roughly speaking, the new prover sends y in addition to the original proof, and the new verifier first verifies if $V(1^n, y) = 1$ and then uses the PRG constructed from y to derandomize the original randomized verification procedure, which can be simulated by a polynomial-size $\mathscr{C}$ circuit.

An almost-everywhere (sublinear witness) NTIME hierarchy with refuter. A critical piece of Williams' proof that $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ (and later work) is the NTIME hierarchy [SFM78, Žák83]. However, as mentioned earlier (Section 1.2.2), that hierarchy is only known to hold infinitely often; consequently, the resulting circuit lower bounds fail to be almost-everywhere, and extending the NTIME hierarchy to hold almost-everywhere is notoriously open.

Nevertheless, Fortnow and Santhanam [FS16] proved an almost-everywhere NTIME hierarchy for a restricted subclass of NTIME, where the "weak" nondeterministic machines (being diagonalized against) use witnesses of length less than n bits. Let $\mathsf{NTIMEGUESS}[t(n), g(n)]$ be the class of languages decided by nondeterministic algorithms running in $O(t(n))$ steps and guessing at most $g(n)$ bits. Fortnow and Santhanam proved there is a language L in nondeterministic $O(T(n))$ time that is not decidable, even infinitely-often, by nondeterministic $o(T(n))$-time $n/10$-guess machines:

Theorem 1.5.8 ([FS16])**.** *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, $\mathsf{NTIME}[T(n)] \not\subset$ i.o.-$\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.*

Our crucial new ingredient is the construction of a "refuter" for the hierarchy of Theorem 1.5.8: an algorithm with an NP oracle that can efficiently find bad inputs for any $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ machine.

Theorem 1.5.9 (Refuter with an NP oracle)**.** *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, there is a language $L \in \mathsf{NTIME}[T(n)]$ and an algorithm $\mathcal{R}$ such that:*

1. ***Input.*** *The input to $\mathcal{R}$ is a pair $(M, 1^n)$, with the promise that M describes a nondeterministic Turing machine running in $o(T(n))$ time and guessing at most $n/10$ bits.*
2. ***Output.*** *For every fixed M and every sufficiently large n, $\mathcal{R}(M, 1^n)$ outputs a string $x \in \{0,1\}^n$ such that $M(x) \neq L(x)$.*
3. ***Complexity.*** *$\mathcal{R}$ runs in $\mathrm{poly}(T(n))$ time with adaptive access to an SAT oracle.*

Since $\mathcal{R}$ can find counterexamples to any faster algorithm attempting to decide L, we call $\mathcal{R}$ a refuter.

Applying the above refuter construction instead of the general NTIME hierarchy in the original proof of [Wil11], we can achieve almost-everywhere circuit lower bounds. See Section 4.5 for more intuitions and a formal proof.

An XOR lemma based on approximate linear sums of circuits. Our second important technical ingredient—critical to our average-case lower bounds—is a new XOR Lemma based on approximate linear sums of circuits. The XOR Lemma (originally due to Yao [GNW11]) says that if an n-bit

Boolean function f cannot be *weakly approximated* (*e.g.*, 0.99-approximated) by small circuits, then the kn-bit Boolean function $f^{\oplus k}$ cannot be *strongly approximated* (*e.g.*, $(1/2+2^{-\Omega(k)})$-approximated) by smaller and simpler circuits.[28]

It is tempting to apply the XOR Lemma directly to prove strongly average-case lower bounds against ACC^0 (or $\mathsf{AC}^0[2]$) given that weak bounds are known. However, when we apply the XOR Lemma to a restricted circuit class $\mathscr{C}$, the most refined analysis of the XOR Lemma [Imp95, Kli01] still only shows that 0.99-inapproximability for $\mathsf{MAJ}_{t^2} \circ \mathscr{C}$ computing f implies $(1/2+1/t)$-inapproximability for $\mathscr{C}$ computing $f^{\oplus k}$, where $k = O(\log t)$. That is, applying the XOR Lemma to show strongly average-case lower bounds against ACC^0, evidently requires proving weakly average-case lower bounds against $\mathsf{MAJ} \circ \mathsf{ACC}^0$. But this task seems just as hard as proving strongly average-case lower bounds in the first place!

We avoid this issue by proving a new kind of XOR Lemma, based on Levin's proof of the XOR Lemma [Lev87]. Recall that (Definition 1.4.6) a $\mathsf{Sum} \circ \mathscr{C}$ circuit $L\colon \{0,1\}^n \to \mathbb{R}$ is defined by $L(x) := \sum_{i\in[m]} \alpha_i \cdot C_i(x)$, where each $\alpha_i \in \mathbb{R}$ and $C_i \in \mathscr{C}$. We call L a $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit, if $L(x) \in [0,1]$ for all $x \in \{0,1\}^n$. The *complexity* of L is defined to be the maximum of $\sum_i |\alpha_i|$ and the sum of the sizes of each C_i (*i.e.*, $\mathsf{SIZE}(L)$).

Our new XOR Lemma shows that if a Boolean function f cannot be well-approximated by linear combinations of $\mathscr{C}$ circuits *on average*, then $f^{\oplus k}$ is strongly average-case hard for $\mathscr{C}$ circuits. The flexibility afforded by linear combinations allows us to improve our results to strongly average-case lower bounds.

Lemma 1.5.10 (New XOR Lemma based on approximate linear sums)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function, $\delta \in (0, \frac{1}{2})$, and $k, s \in \mathbb{N}_{\geq 1}$. Let $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2}-\delta\right)$. If f cannot be $(1-\delta)$-approximated in ℓ_1 distance by $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity $O\left(\frac{n\cdot s}{(\delta\cdot\varepsilon_k)^2}\right)$, then $f^{\oplus k}$ cannot be $(\frac{1}{2}+\varepsilon_k)$-approximated by $\mathscr{C}$ circuits of size s.*

Notes and Bibliographic Details. Results from this subsection are from [CLW20], which appeared in FOCS 2020. In retrospect, [CLW20] is an important development in the algorithmic method for two reasons: (1) it introduced the ideas of refuters, which gives the first almost-everywhere separation in the algorithmic method, and (2) it identified an *algorithmic-method friendly* XOR lemma based on approximate linear sums, which not only gives stronger average-case lower bounds, but also can be used to greatly simplify the previous average-case lower bound in [Che19] and [CR20].

[28] The function $f^{\oplus k}$ partitions its kn-bit input into k blocks $x_1, x_2, \ldots, x_k$ of length n each, and outputs $\bigoplus_{i=1}^k f(x_i)$.

Both techniques above are utilized by several subsequent works [CL21, HV21, RSW22].

1.6 Strongly Average-case Lower Bounds for Nondeterministic Time Classes from Non-trivial Derandomization

Next, we discuss our results on generalizing the algorithmic method to prove (strongly) average-case lower bounds. We first show that a non-trivial improvement on the brute-force $(2^n \cdot \mathrm{poly}(S))$-time algorithm for $\widetilde{\mathsf{CAPP}}$ already implies strongly average-case lower bounds against $\mathscr{C}$.

Theorem 1.6.1. *Let $\mathscr{C}$ be a typical concrete circuit class[29] such that $\mathscr{C}$ circuits of size S can be implemented by (fan-in 2 De-Morgan) circuits of depth $O(\log S)$. For every constant $\varepsilon \in (0,1)$, the following hold:*

- *(*NP **Average-Case Lower Bound**$)$ *If $\widetilde{\mathsf{CAPP}}$ of $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $2^{\varepsilon n}$ can be solved in $2^{n-\varepsilon n}$ time, then for every constant $k \geq 1$, NP cannot be $(1/2 + n^{-k})$-approximated by $\mathscr{C}$ circuits of n^k size.*
- *(*NQP **Average-Case Lower Bound**$)$ *If $\widetilde{\mathsf{CAPP}}$ of $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size 2^{n^ε} can be solved in 2^{n-n^ε} time, then for every constant $k \geq 1$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathscr{C}$ circuits of $2^{\log^k n}$ size.*
- *(*NEXP **Average-Case Lower Bound**$)$ *If $\widetilde{\mathsf{CAPP}}$ of $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $\mathrm{poly}(n)$ can be solved in $2^n/n^{\omega(1)}$ time, then NE cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by $\mathscr{C}$ circuits of $\mathrm{poly}(n)$ size.*

By the standard discriminator lemma [HMP+93], we immediately obtain worst-case lower bounds for $\mathsf{MAJ} \circ \mathscr{C}$ circuits as well.

Corollary 1.6.2. *Under the algorithmic assumptions of Theorem 1.6.1, we obtain the following worst-case lower bounds for $\mathsf{MAJ} \circ \mathscr{C}$ circuits in the corresponding cases: (1) NP does not have n^k-size $\mathsf{MAJ} \circ \mathscr{C}$ circuits for all $k \geq 1$; (2) NQP does not have $2^{\log^k n}$-size $\mathsf{MAJ} \circ \mathscr{C}$ circuits for all $k \geq 1$; (3) NE does not have $\mathrm{poly}(n)$-size $\mathsf{MAJ} \circ \mathscr{C}$ circuits.*

Remark 1.6.3. *We remark that the conclusions of Theorem 1.6.1 still hold if the corresponding $\widetilde{\mathsf{CAPP}}$ algorithms are* nondeterministic*. That is, on any computational path, it either outputs a correct estimation[30] or rejects, and it does not reject on all branches.*

[29] Recall that we use $\mathsf{AND}_4 \circ \mathscr{C}$ to denote an AND of four $\mathscr{C}$ subcircuits, and the size of such a circuit is defined to be the sum of the sizes of all $\mathscr{C}$ subcircuits plus 1.

[30] It is allowed that on different paths it outputs different estimations as long as they are all within an additive error of $1/S$ to the correct answer.

We note that Theorem 1.6.1 only covers circuit classes that are weaker than formulas. (But arguably it covers most of the interesting circuit classes except for general circuits.) We complement that with the following theorem.

Theorem 1.6.4. *For every constant $\varepsilon \in (0,1)$, the following hold:*

- *(*NQP **Average-Case Lower Bound***) If* CAPP *for circuits of size 2^{n^ε} can be solved in 2^{n-n^ε} time, then for every constant $k \in \mathbb{N}_{\geq 1}$,* NQP *cannot be $(1/2 + 2^{-\log^k n})$-approximated by circuits of $2^{\log^k n}$ size.*[31]

$\widetilde{\text{CAPP}}$ vs CAPP. We also remark that $\widetilde{\text{CAPP}}$ is a harder problem than CAPP: $\widetilde{\text{CAPP}}$ requires a $1/S$ additive error, while CAPP only requires an additive error of $1/3$. Nonetheless, a $\widetilde{\text{CAPP}}$ algorithm is *much weaker* than a full-power #SAT algorithm, and is widely believed to exist even for general circuits under the common belief that $\mathsf{prBPP} = \mathsf{prP}$.

Strongly average-case lower bounds for $\mathsf{ACC}^0 \circ \mathsf{THR}$. Applying the non-trivial #SAT algorithms for $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits in [Wil14a], it follows that NQP cannot be even weakly approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, and it is (worst-case) hard for $\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

Theorem 1.6.5. *For every $k \geq 1$,* NQP *cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$. Consequently,* NQP *cannot be computed by $\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$ (in the worst-case), for all $k \geq 1$.*

Very recently, building on [CW19] and the FGLSS reduction [FGL+91], Vyas and Williams [VW20] proved that NQP cannot be computed by $\mathsf{EMAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$[32] circuits of size $2^{\log^k n}$ for all $k \geq 1$.[33] Our result improves on theirs as $\mathsf{EMAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits is a subclass of $\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits (see, *e.g.*, [HP10]).

Theorem 1.6.5 implies that NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\log^k n$-degree $\mathbb{F}_2$-polynomials, and thus breaks the $1/2 + 1/\sqrt{n}$ Razborov-Smolensky barrier (albeit with a hard function in NQP).

Towards TC^0_3 lower bounds. Recall that in Theorem 1.4.5 we proved that non-trivial $\widetilde{\text{CAPP}}$ algorithms for $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits would already imply $\mathsf{THR} \circ \mathsf{THR}$ circuit lower bounds. Here, we

[31] It is possible to generalize Theorem 1.6.4 to cover NP lower bounds and NEXP lower bounds similar to Theorem 1.6.1. We do not discuss these generalizations in the thesis since it would be too repetitive.

[32] EMAJ is the "exact majority" function which outputs 1 on an n-bit input if and only if the number of ones in the input equals $\lceil \frac{n}{2} \rceil$.

[33] [VW20] indeed proved a stronger lower bound for NQP against $f \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ for every symmetric function f with a "sparse" companion function; see [VW20] for details.

significantly improved that connection by showing it would indeed imply TC^0_3 lower bounds!

Theorem 1.6.6. *If there is a* $2^n/n^{\omega(1)}$ *time* $\widetilde{\mathsf{CAPP}}$ *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{MAJ} \circ \mathsf{MAJ}$ *circuits, then* $\mathsf{NEXP} \not\subset \mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$.

We remark that $\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ is equivalent to $\mathsf{MAJ} \circ \mathsf{THR} \circ \mathsf{THR}$ (since $\mathsf{MAJ} \circ \mathsf{MAJ} = \mathsf{MAJ} \circ \mathsf{THR}$ [GHR92]). Since exponential-size (worst-case) lower bounds against $\mathsf{MAJ} \circ \mathsf{MAJ}$ are already known, if only we can "mine" a non-trivial CAPP algorithm (which is widely believed to exist) for $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits from these lower bounds, we would have worst-case lower bounds against TC^0_3.

1.6.1 Our Techniques: A Derandomization-centric View of the Algorithmic Method and a Win-Win Argument

To prove average-case lower bounds against NQP, we deviate from Williams' original approach, which crucially relies on easy witness lemmas [IKW02, MW18] (see Section 4.1 for an high-level overview of Williams' proof).[34]

Our new approach is centered around *derandomization of Merlin-Arthur protocols*. Here, for simplicity of presentation, we will be focusing on proving that NQP is strongly average-case hard against ACC^0. Recall that Corollary 1.5.7 derandomizes $\mathsf{MA}_{\mathsf{ACC}^0}$ into NQP. Therefore, intuitively speaking, if we can prove a circuit lower bound for $\mathsf{MA}_{\mathsf{ACC}^0}$, then the same lower bound would holds for NQP as well. Indeed, unconditional lower bounds for Merlin-Arthur protocols have been known for a while: [San09] proved that $\mathsf{MA}_{/1}$ cannot be $(1/2 + n^{-k})$-approximated by n^k-size circuits for every $k \in \mathbb{N}_{\geq 1}$. Hence, if we can prove similar average-case lower bounds for $\mathsf{MA}_{\mathsf{ACC}^0}$ against ACC^0, then we would be able to derive average-case lower bounds for NQP against ACC^0.

Formally implementing the plan above requires solving some significant challenges. First, the derandomization from Corollary 1.5.7 only gives *infinitely often* derandomization, and the lower bound from [San09] is also only infinitely often. Combining them together does not guarantee that the hardness is *retained* (it could be the case that derandomization only works when the MA hard language is not hard). Second, we are not able to prove strongly average-case lower bounds (*i.e.*, $(1/2 + 1/\mathrm{poly}(n))$-hardness) for $\mathsf{MA}_{\mathsf{ACC}^0}$ against ACC^0, but only a weakly average-case hard one (*i.e.*, $(1 - 1/\mathrm{poly}(n))$-hardness). Both issues above require substantially new ideas. Using ideas

[34] The main reason is that we do not know how to prove an average-case analogue of the easy witness lemma. We also think our new approach is more direct and thus easier to understand.

from [MW18], we resolve the first issue by proving a stronger lower bound for $\mathsf{MA}_{\mathsf{ACC}^0}$ so that its hardness is retained after applying the derandomization from Corollary 1.5.7; see Section 4.3.1 for the details.

To resolve the second issue, we would like to perform a mild-to-strong hardness amplification to convert $(1 - 1/\mathrm{poly}(n))$-hardness to $(1/2 + 1/\mathrm{poly}(n))$-hardness. However, we do not have such hardness amplification results for weak circuit classes such as ACC^0, and there are black-box barriers against this possibility [SV10]. We manage to circumvent this issue by proving an interesting win-win theorem using techniques from *randomized encodings* [IK02, AIK06, GGH+07]; see Section 4.3.2 for details.

1.6.2 Notes and Bibliographic Details

Results from this subsection are mostly from [Che19] and [CR20], which appeared in FOCS 2019 and STOC 2020, respectively.[35] [CR20] builds heavily on [Che19], and most results from [Che19] is improved by [CR20].

In this thesis, we will give a simplified and streamlined proof of the results from [CR20], which uses a result from a subsequent work [CLW20] (Theorem 1.5.6).[36] [CR20] also proved some strongly average-case lower bounds against $\mathsf{Sum} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$, which we omit in this thesis for a more focused presentation (these lower bounds are also improved by a subsequent work [CLLO21]).

1.7 Construction of Rigid Matrices and Correlation Bounds against $\mathbb{F}_2$-Polynomials

Finally, we discuss our results on (semi-)explicit construction of rigid matrices and proving correlation bounds against $\mathbb{F}_2$-polynomials.

1.7.1 Almost-Everywhere Construction of Extremely Rigid Matrices

The problem of efficiently constructing *rigid matrices* is a longstanding open problem in complexity theory [Val77, Lok09]. We first recall its definition formally.

[35] The journal version of [CR20] is published in SIAM Journal of Computing [CR21].
[36] This is why we break the chronological order of the presentation and present Section 1.5 before Section 1.6.

Definition 1.7.1. *Let $\mathbb{F}$ be a field. For $r, n \in \mathbb{N}$ and a matrix $M \in \mathbb{F}^{n \times n}$, the r-rigidity of M, denoted as $\mathcal{R}_M(r)$, is the minimum Hamming distance between M and a matrix of rank at most r.*

In 2019, Alman and the author [AC19] showed that rigid matrices over the field $\mathbb{F}_2$ with interesting parameters (considered by [Raz89]) could be constructed *infinitely often* in P^{NP} via the algorithmic method. Their proof has been simplified and significantly improved by Bhangale *et al.* [BHPT20]. Formally, [BHPT20] construct a P^{NP} algorithm M which, for infinitely many n, $M(1^n)$ outputs a matrix H_n such that $\mathcal{R}_{H_n}(2^{\log n/\Omega(\log\log n)}) \geq \Omega(n^2)$ over $\mathbb{F}_2$.[37] Similar results hold for all finite fields of constant order.

Almost-everywhere construction of rigid matrices with an NP oracle. Applying similar ideas from the proof of Theorem 1.5.1 and Theorem 1.5.2, we can strengthen their construction to an almost-everywhere one.

Theorem 1.7.2. *There is a $\delta \in (0,1)$ and a rank parameter $r = 2^{\log n/\Omega(\log\log n)}$ such that there is a P^{NP} algorithm that on input 1^n outputs a matrix $H_n \in \mathbb{F}_2^{n \times n}$ satisfying $\mathcal{R}_H(r) \geq \delta n^2$ over $\mathbb{F}_2$, for all large enough $n \in \mathbb{N}_{\geq 1}$.*

Extremely rigid matrices. In addition, building on a new *derandomized XOR Lemma* (see Section 1.7.3), we are able to strengthen Theorem 1.7.2 into a construction of *extremely* rigid matrices. Formally:

Theorem 1.7.3. *For every constant $\varepsilon \in (0,1)$, there is a P^{NP} algorithm that on input 1^n outputs a matrix $H_n \in \mathbb{F}_2^{n \times n}$ satisfying $\mathcal{R}_{H_n}(2^{\log^{1-\varepsilon} n}) \geq (1/2 - \exp(-\log^{2/3\cdot\varepsilon} n)) \cdot n^2$, for every sufficiently large n.*[38]

We note that independently of our work ([CL21]), [HV21] proved a weaker trade-off of $\mathcal{R}_{H_n}(2^{\log^{1-\varepsilon} n}) \geq (1/2 - \exp(-\log^{1/2\cdot\varepsilon} n)) \cdot n^2$ (similar to our result, their ε can be set to sub-constant as well, to work for rank $2^{\log n/\Omega(\log\log n)}$).

[37] [BHPT20] (and also [AC19]) indeed achieved a *nondeterministic construction*. That is, there is a nondeterministic algorithm M such that for infinitely many n: (1) On a computational path, $M(1^n)$ either outputs a valid rigid matrix or "failed". (2) There exists at least one computational path that $M(1^n)$ outputs a valid rigid matrix. This thesis focuses on P^{NP} constructions since they give a single rigid matrix. We also note that independently of [BHPT20], Viola [Vio20] proved an $\Omega(n/\log^2 n)$ probabilistic degree lower bound for E^{NP} using the algorithmic method (which is implied by the construction of the rigid matrices from [BHPT20]).

[38] We only state the case for $2^{\log^{1-\varepsilon} n}$ rank for constant $\varepsilon \in (0,1)$ to simplify the statement here; our trade-off applies to the case when ε is sub-constant as well, and in particular, it applies to the rank $2^{\log n/\Omega(\log\log n)}$ as in Theorem 1.7.2; see Section 10.7 for details.

1.7.2 Strong Correlation Bounds against $\mathbb{F}_2$-Polynomials

We also give some strong correlation bounds against $\mathbb{F}_2$-polynomials.

Theorem 1.7.4. *There is a function $f \in \mathsf{E}^{\mathsf{NP}}$ such that for every sufficiently large $n \in \mathbb{N}_{\geq 1}$ and for any $d \leq o(n/\log n)^{1/2}$, it holds that $\mathrm{corr}(f_n, d) \leq 2^{-d}$.*

One may wonder whether Theorem 1.7.4 can be further improved to, say, $\mathrm{corr}(f_n, d) \leq 2^{-d}$ for $d = n^{1/2+\varepsilon}$. We observe that such improvement will imply new lower bounds against AC^0_3 circuits (*i.e.*, depth-3 AC^0 circuits).

Theorem 1.7.5. *For any function $d(n) : \mathbb{N} \to \mathbb{N}$, if there is a function f in E^{NP} such that $\mathrm{corr}(f_n, d(n)) \leq 2^{-d(n)}$ for infinitely many $n \in \mathbb{N}$, then there is a function g in E^{NP} that does not admit AC^0_3 circuits of size at most $2^{o(d(n))}$.*

Currently, the best known lower bound against AC^0_3 circuits is $2^{\Omega(\sqrt{n})}$ [Hås89]. It has been a notorious open question to prove better lower bounds against depth-3 AC^0 circuits, even for functions in complexity class as large as E^{NP}. Therefore, improving our result, even by an "epsilon amount" in the exponent, would imply a breakthrough in constant-depth circuit lower bounds.

1.7.3 Our Techniques: a New Derandomized XOR Lemma

Perhaps more interestingly, our new results are all proved by a new *derandomized* XOR lemma based on approximate linear sums, and as far as we know, this is the first application of hardness amplification in the context of constructing rigid matrices. Recall that Lemma 1.5.10 roughly says that if $f : \{0,1\}^n \to \{0,1\}$ is mildly hard against $\mathsf{Sum} \circ \mathscr{C}$ circuits, then $f^{\oplus k}$ is $(1/2 + 2^{-\Omega(k)})$-hard against $\mathscr{C}$ circuits. However, a major disadvantage of Lemma 1.5.10 is that it blows up the input length of the function: now $f^{\oplus k}$ takes $n \cdot k$ bits as input, which makes the average-case lower bounds weaker.[39] We manage to prove a new derandomized XOR Lemma, based on approximate linear sums. Roughly speaking, given a function $f : \{0,1\}^n \to \{0,1\}$ that is mildly hard against $\mathsf{Sum} \circ \mathscr{C}$ circuits, we show how to construct a new function $\mathsf{Amp}^f : \{0,1\}^{O(n)} \to \{0,1\}$ such that Amp^f is strongly average-case hard against $\mathscr{C}$ circuits. The crucial improvement here is that as opposed to the hard function $f^{\oplus k}$ in an XOR Lemma, Amp^f has essentially the same input length as f; see Chapter 10 for more details.

[39] For example, if $k = \sqrt{n}$. Then f takes $m = n^{1.5}$ bits as input, then f is only $(1/2 + m^{-1/3})$-hard instead of $(1/2 + m^{-1/2})$-hard.

1.7.4 Notes and Bibliographic Details

Results from this subsection are from [CLW20] and [CL21], which appeared in FOCS 2020 and STOC 2021, respectively. We remark that although this line of work on semi-explicit construction of rigid matrices was initiated by a work by Alman and the author [AC19], we do not cover that paper in this thesis, since it was subsumed by the subsequent work of [BHPT20].

Chapter 2

Hardness vs. Randomness Revised: Super-fast and Non-Black-box Derandomization

2.1 Background and Motivation

The power of randomness has been demonstrated throughout computer science. In particular, randomness is extremely useful for the design of algorithms (see, *e.g.*, [MR95, CLRS09]) and for the theory of proof systems (see, *e.g.*, [LFKN92, Sha92] for notable examples). Given its ubiquitous presence, a natural question is whether we can eliminate the usage of randomness while achieving similar performance (we call such a procedure *derandomization*). This question is fundamental for both theoretical and practical reasons. On the theoretical side, understanding the power of randomness in computation contributes tremendously to our study of complexity theory. On the practical side, deterministic algorithms have several advantages over their randomized counterparts and are preferred in many important situations.[1]

Given the wide applications of randomness in algorithm design, it came as a surprise that the long line of works typically referred to as "hardness-to-randomness", which was initiated by [Yao82, BM84, Nis91, NW94], showed that certain *lower bounds* against non-uniform circuits imply derandomization with bounded *runtime overhead*. The fastest conditional derandomization

[1]For example, deterministic algorithms never *err*, while randomized algorithms may make mistakes. This makes deterministic algorithms much more preferable in certain critical situations. Also, deterministic algorithms are *reproducible* in the sense that running the algorithms on the same input multiple times leads to the same output. In contrast, a randomized algorithm may behave quite differently due to its usage of randomness.

in the classical line-of-works was proved by Impagliazzo and Wigderson [IW97]. They showed that $\mathsf{prBPP} = \mathsf{prP}$ follows from plausible lower bound assumptions.[2] In particular, this conclusion implies that when solving decision problems, randomness can be deterministically simulated with a *polynomial runtime overhead*.

Besides algorithms, randomness is also crucial for *proof systems*. While we do not expect to be able to eliminate randomness from general interactive proof systems with polynomially many rounds of interactions (as that would imply $\mathsf{NP} = \mathsf{IP} = \mathsf{PSPACE}$, by [Sha92]), another classical line of works suggests that for proof systems that only use constantly many rounds of interaction, this might be doable. In particular, assuming sufficiently strong circuit lower bounds,[3] we have that $\mathsf{AM} = \mathsf{NP}$ (for a subset of prominent works in this area, see, *e.g.*, [KvM02, IKW02, MV05, SU05, GSTS03, SU07]).

The hardness-to-randomness framework has been extremely successful and influential in theoretical computer science. Still, there are two fundamental questions regarding derandomization that have remained open:

1. **What is the fastest derandomization?** The classical works showed that both randomized algorithms and constant-round Arthur-Merlin proof systems can be derandomized with a polynomial overhead under plausible assumptions. However, the polynomial overhead could be huge. For example, for randomized algorithms, one can only derandomize $T(n)$-time randomized algorithms into deterministic algorithms with at least $T(n)^c$ time for a constant $c \geq 8$, even starting from very strong assumptions like $\mathsf{TIME}[2^n] \not\subset \mathsf{SIZE}(2^{(1-\varepsilon)n})$. In an exciting recent work, under hardness assumptions against non-uniform Merlin-Arthur protocols, Doron, Moshkovitz, Oh, and Zuckerman [DMOZ20] proved that randomized algorithms can be derandomized with only a quadratic overhead in the running time. Still, it remains open whether the quadratic-time derandomization above is the fastest possible and whether we can establish similarly fast derandomization of Arthur-Merlin protocols under plausible hardness assumptions.
2. **What is the minimum hardness assumption for derandomization?** Classical works on derandomization proved that circuit lower bounds for $\mathsf{E} = \mathsf{TIME}[2^{O(n)}]$ is equivalent to the existence of pseudorandom generators (PRGs) for polynomial-size circuits (a.k.a. black-box

[2] formally, under the assumption that exponential-time does not have $2^{o(n)}$-size circuits, even infinitely often

[3] For example, it suffices to assume that for some $\varepsilon > 0$ it holds that $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ is hard for SVN circuits of size $2^{\varepsilon \cdot n}$ on all input lengths (see [MV05]).

derandomization)[4]. However, it is not clear whether those circuit lower bounds are also *necessary* for derandomization, since an approach for derandomization *could* heavily exploit the structure of the given input circuit (instead of being ignorant of it). In light of this gap, an important question is whether we can formulate a natural hardness assumption that is both necessary and sufficient for derandomization.

In the second part of our thesis, we made significant progress on both of the questions above by introducing a new framework for derandomization.

2.2 What is the Fastest Derandomization?

2.2.1 Fast Derandomization of Algorithms

Recall that [DMOZ20] demonstrated quadratic-overhead derandomization assuming hardness against non-uniform Merlin-Arthur protocols. In this thesis, we further improve the running time under an incomparable (but arguably more standard) assumption.

For a time function $T(n)$, consider the following assumption: Non-uniformly secure one-way functions exist, and for some $\delta = \delta(\varepsilon)$ and $k = k_T(\varepsilon)$ there is a problem in $\mathsf{DTIME}[2^{k \cdot n}]$ that is hard for algorithms that run in time $2^{(k-\delta)\cdot n}$ and use $2^{(1-\delta)\cdot n}$ bits of advice. Under this assumption, we show that:

1. (**Worst-case derandomization.**) Probabilistic algorithms that run in time $T(n)$ can be deterministically simulated in time $n \cdot T(n)^{1+\varepsilon}$.
2. (**Conditional optimality.**) For worst-case derandomization, the multiplicative time overhead of n is essentially optimal, conditioned on a counting version of the nondeterministic strong exponential-time hypothesis (*i.e.*, on #NSETH).[5]

We also present an alternative proof for the result of [DMOZ20] that is simpler and more versatile.

Derandomization with no overhead? It appears hard (or impossible) to further improve the overhead of n above (since it would refute #NSETH). One way to get around is to *weaken the correctness condition of derandomization slightly*. We show that under plausible hardness assumptions *against uniform algorithms*[6], a $T(n)$-time randomized algorithm A can be converted into a

[4] See Section 3.5 for a formal definition.
[5] See Assumption 13.3.19 for details.
[6] See Section 12.1.2 for details.

$T(n) \cdot n^{o(1)}$-time deterministic algorithm B, such that *no* $\mathrm{poly}(T(n))$*-time uniform adversary* S *can find a mistake (an input* x *such that* $A(x) \neq B(x)$*) with a non-negligible probability*.

In other words, this derandomization is **indistinguishable from being correct and has almost no overhead**. This already suffices for many (if not most) practical applications. We also prove that **no PRGs** (pseudorandom generators) can give us the almost-no-overhead derandomization mentioned above. In fact, we indeed rely on a new non-black-box approach based on *targeted PRGs* (see Section 2.3 for more details).

2.2.2 Fast Derandomization of Arthur-Merlin Protocols

Similar to the derandomization of algorithms, recall that classical works [KvM02, MV05] showed that $\mathsf{AM} = \mathsf{NP}$ follows from plausible circuit lower bounds, but again the polynomial overhead is huge. This brings us to the question of obtaining the fastest derandomization for Arthur-Merlin protocols.

In this thesis, we show conditional derandomization of MA and of AM with **optimal runtime overhead**, where optimality is again under the #NSETH assumption. Specifically, denote by $\mathsf{AMTIME}^{[\rightleftharpoons c]}[T]$ a protocol with c turns of interaction in which the verifier runs in polynomial time T.[7] For every $\varepsilon > 0$, we show:

$$\mathsf{MATIME}[T] \subseteq \mathsf{NTIME}[T^{2+\varepsilon}] \,,$$
$$\mathsf{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{NTIME}[n \cdot T^{\lceil c/2 \rceil + \varepsilon}] \,,$$

assuming the existence of properties of Boolean functions that can be recognized quickly from the function's truth-table such that functions with the property are hard for proof systems that receive a near-maximal amount of non-uniform advice.

To obtain faster derandomization, we introduce the notion of a **deterministic doubly efficient argument system**. This is a doubly efficient proof system in which the verifier is *deterministic*, and the soundness is relaxed to be computational, as follows: For every probabilistic polynomial-time adversary $\tilde{P}$, the probability that $\tilde{P}$ finds an input $x \notin L$ and misleading proof π such that $V(x, \pi) = 1$ is negligible.

Under strong hardness assumptions,[8] we prove that **any constant-round doubly efficient**

[7] In particular, $\mathsf{AMTIME}^{[\rightleftharpoons 2]}[T]$ denotes the usual AM protocol in which Arthur speaks first and Merlin then respond.
[8] See Section 13.1.3 for more details.

proof system can be compiled into a deterministic doubly efficient argument system, with essentially no time overhead. As one corollary, under strong hardness assumptions, for every $\varepsilon > 0$ there is a deterministic verifier V that gets an n-bit formula Φ of size $2^{o(n)}$, runs in time $2^{\varepsilon \cdot n}$, and satisfies the following: An honest prover running in time $2^{O(n)}$ can print, for every Φ, a proof π such that $V(\Phi, \pi)$ outputs the number of satisfying assignments for Φ; and for every adversary $\tilde{P}$ running in time $2^{O(n)}$, the probability that $\tilde{P}$ finds Φ and π such that $V(\Phi, \pi)$ outputs an incorrect count is $2^{-\omega(n)}$.

2.3 What is the Minimum Hardness Assumption for Derandomization?

In this thesis, we propose a new approach to the hardness-to-randomness framework and to the $\mathsf{prBPP} = \mathsf{prP}$ conjecture. Recall that classical results rely on non-uniform hardness assumptions to construct derandomization algorithms that work in the worst-case, or rely on uniform hardness assumptions to construct derandomization algorithms that work only in the average-case. In both types of results, the derandomization algorithm is "black-box" and uses the standard PRG approach. In this thesis, we present results that closely relate *new and natural uniform hardness assumptions* to *worst-case derandomization* of prBPP, where the algorithms underlying the latter derandomization are *non-black-box*.

In our main result, we show that $\mathsf{prBPP} = \mathsf{prP}$ if the following holds: There exists a multi-output function computable by logspace-uniform circuits of polynomial size and depth n^2 that cannot be computed by uniform probabilistic algorithms in time n^c, for some universal constant $c > 1$, on *almost all inputs*. The required failure on "almost all inputs" is stronger than the standard requirement of failing on one input of each length; however, the same assumption without the depth restriction on f is *necessary* for the conclusion. This suggests a potential equivalence between worst-case derandomization of prBPP of any form (*i.e.*, not necessarily by a black-box algorithm) and the existence of efficiently-computable functions that are hard for probabilistic algorithms on almost all inputs.

Technically, our approach is to design *targeted PRGs and HSGs*, as introduced by Goldreich [Gol11a]. The targeted PRGs/HSGs "produce randomness from the input", as suggested by Goldreich and Wigderson [GW02]; and their analysis relies on non-black-box versions of the reconstruction procedure of Impagliazzo and Wigderson [IW98]. Our main reconstruction procedure crucially relies on the ideas underlying the proof system of Goldwasser, Kalai, and Rothblum [GKR15].

2.4 Notes and Bibliographic Details

Results from this chapter are from [CT21b, CT21a, CT22]. The first two appeared in STOC 2021 and FOCS 2021, respectively. Our results on near-optimal (worst-case) derandomization of $\mathsf{prBPTIME}[T]$ (*i.e.*, $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[T^{1+\varepsilon} \cdot n]$) are presented in Chapter 11. Our results on almost no-overhead effective derandomization of $\mathsf{BPTIME}[T]$ and new hardness conditions for $\mathsf{prBPP} = \mathsf{prP}$ are presented in Chapter 12. Our results on fast derandomization of Arthur-Merlin protocols are presented in Chapter 13.

Omitted Work

For a more focused presentation, this thesis does not include a number of results that the author has published during his PhD.[9] Some notable such results (in the author's opinion), are listed below (in chronological order):

1. The line of works on the topic of *hardness magnification* [CT19, CMMW19, CJW19, CJW20, CHO+22, CJSW21, CLY22].
2. On The Hardness of Approximate and Exact (Bichromatic) Maximum Inner Product [Che20].
3. On Exponential-Time Hypotheses, Derandomization, and Circuit Lower Bounds [CRTY20].
4. Almost Optimal Super-Constant-Pass Streaming Lower Bounds for Reachability [CKP+21].
5. Majority vs. Approximate Linear Sum and Average-Case Complexity Below NC^1 [CLLO21].
6. Truly Low-Space Element Distinctness and Subset Sum via Pseudorandom Hash Functions [CJWW22].
7. Unstructured Hardness to Average-Case Randomness [CRT22].

[9] Some works such as [Che19, AC19] are not included because their results are improved by subsequent works.

Part II

Algorithmic Method: From Non-trivial Derandomization to Circuit Lower Bounds

Chapter 3

Preliminaries

We use $\mathbb{N}$ to denote all non-negative integers, and $\mathbb{N}_{\geq 1}$ to denote all positive integers. We use $\mathsf{GF}(p^r)$ to denote the finite field of size p^r, where p is a prime and r is an integer. For a set X, we often use $x \in_{\mathsf{R}} X$ to denote that we pick an element x from X uniformly at random. We also use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$.

For $r, m \in \mathbb{N}$, we use $\mathcal{F}_{r,m}$ to denote the set of all functions from $\{0,1\}^r$ to $\{0,1\}^m$. For a language $L\colon \{0,1\}^* \to \{0,1\}$, we use L_n to denote its restriction on n-bit inputs. For a function $f\colon \{0,1\}^n \to \{0,1\}$, we use $\mathsf{tt}(f)$ to denote the truth-table of f (*i.e.*, $\mathsf{tt}(f)$ is a string of length 2^n such that $\mathsf{tt}(f)_i$ is the output of f on the i-th string from $\{0,1\}^n$ in the lexicographical order). For a string $Z \in \{0,1\}^{2^n}$, we use $\mathsf{func}(Z)$ to denote the unique function from $\mathcal{F}_{n,1}$ with the truth-table being Z.

Let Σ be an alphabet set. For two strings $x, y \in \Sigma^*$, we use $x \circ y$ to denote their concatenation. We also use $f \circ g$ to denote the composition of two functions f and g. The meaning of the symbol $\circ$ (concatenation or composition) will always be clear from the context. We sometimes use $\vec{x}$ ($\vec{y}, \vec{z}$, etc.) to emphasize that $\vec{x}$ is a vector. For $\vec{x} \in \Sigma^n$ for some $n \in \mathbb{N}$, we use $\vec{x}_{<i}$ and $\vec{x}_{\leq i}$ to denote its prefix $(x_1, \ldots, x_{i-1})$ and $(x_1, \ldots, x_i)$, respectively. We also define $\vec{x}_{>i}$ and $\vec{x}_{\geq i}$ in the same way.

We assume knowledge of basic complexity theory (see [AB09, Gol08] for excellent references on this subject).

3.1 Complexity Classes and Basic Definitions

3.1.1 Basic Circuit Families

Unless otherwise specified, the circuits appear in Part II consist of fan-in 2 AND/OR gates and fan-in 1 NOT gates. We also always measure the size of a circuit by *number of gates* unless explicitly stated otherwise.

A *circuit family* is an infinite sequence of circuits $\{C_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$. A *circuit class* is a collection of circuit families. The *size* of a circuit is the number of *gates* in the circuit, and the size of a circuit family is a function of the input length that upper-bounds the size of circuits in the family. The *depth* of a circuit is the maximum number of wires on a path from an input gate to the output gate.

We will mainly consider classes in which the size of each circuit family is bounded by some polynomial; however, for a circuit class $\mathscr{C}$, we will sometimes also abuse notation by referring to $\mathscr{C}$ circuits with various other size or depth bounds.

AC^0 is the class of circuit families of constant depth and polynomial size, with AND, OR and NOT gates, where AND and OR gates have unbounded fan-in. For an integer m, the function $\mathsf{MOD}_m : \{0,1\}^* \to \{0,1\}$ is one if and only if the number of ones in the input is not divisible by m. The class $\mathsf{AC}^0[m]$ is the class of constant-depth polynomial-size circuit families consisting of unbounded fan-in AND, OR and MOD_m gates, along with unary NOT gates. We denote $\mathsf{ACC}^0 = \cup_{m\geq 2}\mathsf{AC}^0[m]$. We also use AC^0_d (resp. $\mathsf{AC}^0_d[m]$) to denote the subclass of AC^0 (resp. $\mathsf{AC}^0[m]$) with depth at most d.

The function majority, denoted as $\mathsf{MAJ} : \{0,1\}^* \to \{0,1\}$, is the function that outputs 1 if the number of ones in the input is no less than the number of zeros, and outputs 0 otherwise. TC^0 is the class of circuit families of constant depth and polynomial size, with unbounded fan-in MAJ gates. NC^k for a constant k is the class of $O(\log^k n)$-depth and poly-size circuit families consisting of fan-in two AND and OR gates and unary NOT gates.

For $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, we define $\mathsf{Approx\text{-}MAJ}_{n,\varepsilon}$ to be the function that outputs 1 (resp. 0) if at least a $(1-\varepsilon)$ fraction of the inputs are 1 (resp. 0), and is undefined otherwise. We also use $\mathsf{Approx\text{-}MAJ}_n$ to denote $\mathsf{Approx\text{-}MAJ}_{n,1/3}$ for simplicity.

The following standard construction for approximate-majority in AC^0 will be useful for the proofs in this thesis.

Lemma 3.1.1 ([ABO84, Ajt90, Vio09b]). $\mathsf{Approx\text{-}MAJ}_n$ *can be computed by uniform* AC^0_3.

We say that a circuit family $\{C_n\}_{n\in\mathbb{N}}$ is uniform, if there is a deterministic algorithm A, such that $A(1^n)$ runs in time polynomial of the size of C_n, and outputs C_n.[1]

For a circuit class $\mathscr{C}$, we say that a circuit C is a $\mathscr{C}$ oracle circuit, if C is also allowed to use a special oracle gate (which can occur multiple times in the circuit, but with the same fan-in), in addition to the usual gates allowed by $\mathscr{C}$ circuits. We say that an oracle circuit is *non-adaptive*, if on any path from an input gate to the output gate, there is at most one oracle gate.[2]

Typical concrete circuit class. We say a circuit class $\mathscr{C}$ is *concrete*, if we can talk about a single $\mathscr{C}$ circuit $C\colon \{0,1\}^n \to \{0,1\}$ for a fixed input length $n \in \mathbb{N}$. For example, AC^0 is not a concrete circuit class while AC^0_d for any fixed $d \in \mathbb{N}$ is. For two concrete circuit classes $\mathscr{C}$ and $\mathscr{D}$, we say $\mathscr{C}$ is weaker than $\mathscr{D}$, if there exists a constant $c \in \mathbb{N}_{\geq 1}$ such that for every $n, s \in \mathbb{N}_{\geq 1}$ satisfying $s \geq n$, an s-size $\mathscr{C}$ circuit on n-bit inputs has an equivalent s^c-size $\mathscr{D}$ circuits. We use $\mathsf{Formula}$ and $\mathsf{Circuit}$ to denote the concrete circuit classes of fan-in 2 De-Morgan formulas and fan-in 2 De-Morgan circuits (*i.e.*, general circuits).

For a concrete circuit class $\mathscr{C}$ and a function $f\colon \{0,1\}^n \to \{0,1\}$, we define $\mathscr{C}\text{-}\mathsf{SIZE}(f)$ to be the minimum size of a $\mathscr{C}$ circuit computing f exactly. We say a circuit $C : \{0,1\}^n \to \{0,1\}$ γ-approximates a function $f : \{0,1\}^n \to \{0,1\}$, if $C(x) = f(x)$ for at least γ fraction of inputs from $\{0,1\}^n$. For a parameter $\gamma \in (1/2, 1]$, we define $\mathsf{heur}_\gamma\text{-}\mathsf{SIZE}(f)$ to be the minimum size of a circuit that γ-approximating f. Note that $\mathsf{heur}_1\text{-}\mathsf{SIZE}(f) = \mathsf{SIZE}(f)$.

We say that a concrete circuit class $\mathscr{C}$ is typical, if given the description of a circuit C of size s, for indices $i, j \leq n$ and a bit b, the following functions

$$\neg C, C(x_1, \ldots, x_{i-1}, x_j \oplus b, x_{i+1}, \ldots, x_n), C(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$$

all have $\mathscr{C}$ circuits of size s, and their corresponding circuit descriptions can be constructed in $\mathrm{poly}(s)$ time. We also require that $\mathscr{C}$ is weaker than $\mathsf{Circuit}$. That is, $\mathscr{C}$ is typical if it is closed under both *negation* and *projection*, and it can be simulated by general circuits up to a polynomial blow-up in size.

[1] That is, we use the P uniformity by default.

[2] Note that the function computed by the oracle circuit C depends on the truth-table of the oracle.

3.2 Norms and Inner Products

For two functions $f, g\colon \{0,1\}^n \to \mathbb{R}$, we define their inner product as

$$\langle f, g\rangle \coloneqq \mathop{\mathbb{E}}_{x \leftarrow U_n}[f(x) \cdot g(x)].$$

For every real $p \geq 1$, recall that the ℓ_p-norm of f is defined as

$$\|f\|_p \coloneqq \left(\mathop{\mathbb{E}}_{x \leftarrow U_n}|f(x)|^p\right)^{1/p}.$$

We also define the ℓ_∞-norm of f as

$$\|f\|_\infty \coloneqq \max_{x \in \{0,1\}^n}|f(x)|.$$

We need the concept of duality between ℓ_p-norms for different choices of p. We first recall the definition of Hölder conjugates.

Definition 3.2.1. *Let $p, q \in \mathbb{R}_{\geq 1} \cup \{\infty\}$. We say that p and q are Hölder conjugates of each other, if it holds that $1/p + 1/q = 1$. (In particular, it can be the case that $p = 1$ and $q = \infty$ and vice versa.)*

The following inequality and its extensions will be useful for us.

Lemma 3.2.2 (Hölder's inequality)**.** *Let $p, q \in \mathbb{R}_{\geq 1} \cup \{\infty\}$ be such that p and q are Hölder conjugates of each other. Let $f, g\colon \{0,1\}^n \to \mathbb{R}$. Then it holds that*

$$\langle f, g\rangle \leq \|f\|_p \|g\|_q.$$

In particular, when $p = q = 2$, we get the Cauchy-Schwarz inequality $\langle f, g\rangle \leq \|f\|_2 \|g\|_2$.

Lemma 3.2.3. *For every function f and $p, q \in \mathbb{R} \cup \{\infty\}$ such that $1 \leq p \leq q$, it holds that $\|f\|_p \leq \|f\|_q$.*

Lemma 3.2.4 (Duality between ℓ_p spaces)**.** *Let $n \in \mathbb{N}_{\geq 1}$. Let $p, q \in \mathbb{R}_{\geq 1} \cup \{\infty\}$ be such that p and q are Hölder conjugates of each other. For every function $f\colon \{0,1\}^n \to \mathbb{R}$, it holds that*

$$\max_{h:\|h\|_q = 1}\{\langle f, h\rangle\} = \|f\|_p.$$

3.3 Hardness Amplification

We also need the standard worst-case to strongly average-case hardness amplification. We refer to [STV01] for an excellent exposition.

Theorem 3.3.1 ([STV01]). *There is a constant $c \geq 1$ such that, for any time-constructible function $S(n)$ and every $f\colon \{0,1\}^n \to \{-1,1\}$ that does not have (general) circuits of size $S(n)$. There is a function $g\colon \{0,1\}^{O(n)} \to \{-1,1\}$ that cannot be $(1/2 + S(n)^{-1/c})$-approximated by circuits of size $S(n)^{1/c}$. Furthermore, given the 2^n-length truth table of f, the truth table of g can be constructed in $2^{O(n)}$ time.*

3.4 NTIME and MATIME

We first recall the definitions of $\mathsf{NTIME}[T(n)]$ and $\mathsf{NTIMEGUESS}[T(n), G(n)]$.

Definition 3.4.1. *Let $T, G\colon \mathbb{N} \to \mathbb{N}$ be two time-constructible functions. A language $L \in \mathsf{NTIME}[T(n)]$ if there is an $O(T(n))$-time algorithm $V(x,y)$ such that $|x| = n$ and $|y| = T(n)$ and*

$$x \in L \Leftrightarrow \exists y \in \{0,1\}^{T(|x|)} V(x,y) = 1.$$

We call V an $\mathsf{NTIME}[T(n)]$ verifier *for L. Moreover, we say $L \in \mathsf{NTIMEGUESS}[T(n), G(n)]$ if V only takes $G(n)$ bits of witness (i.e., $|y| = G(n)$ instead of $|y| = T(n)$), and call V an* $\mathsf{NTIMEGUESS}[T(n), G(n)]$ *verifier.*

In particular, $\mathsf{NP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}[n^k]$, and $\mathsf{NQP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}[2^{\log^k n}]$.

MA is the randomized version of NP. We now recall the definition of $\mathsf{MATIME}[T(n)]$.

Definition 3.4.2. *For a time-constructible function $T(n)$, a language $L \in \mathsf{MATIME}[T(n)]$ if there is an $O(T(n))$-time algorithm $V(x,y,r)$ such that $|x| = n$ and $|y| = |r| = T(n)$ and*

$$x \in L \Rightarrow \exists y \in \{0,1\}^{T(|x|)} \Pr_{r\in\{0,1\}^{T(|x|)}}[V(x,y,r) = 1] \geq 2/3,$$

and

$$x \in L \Rightarrow \forall y \in \{0,1\}^{T(|x|)} \Pr_{r\in\{0,1\}^{T(|x|)}}[V(x,y,r) = 1] \leq 1/3.$$

We call V an $\mathsf{MATIME}[T(n)]$ verifier *for L.*

In particular, $\mathsf{MA} = \bigcup_{k\in\mathbb{N}} \mathsf{MATIME}[n^k]$.

We will also pay attention to the complexity of the verifiers in $\mathsf{MATIME}[T(n)]$. We define $\mathsf{MATIME}_{\mathscr{C}}[T(n)]$ as follows.

Definition 3.4.3. *Let $\mathscr{C}$ be a circuit class. For a time-constructible function $T(n)$, a language $L \in \mathsf{MATIME}_{\mathscr{C}}[T(n)]$ if there is an $\mathsf{MATIME}[T(n)]$ verifier V for L that also satisfies the following additional condition:*

- *For every $x \in \{0,1\}^n$ and $y \in \{0,1\}^{T(n)}$, $V(x,y,\cdot)$ (the restriction of V to the randomness part) has a $T(n)$-size $\mathscr{C}$ circuit.*

We call V an $\mathsf{MATIME}_{\mathscr{C}}[T(n)]$ verifier *for L.*

In particular, $\mathsf{MA}_{\mathscr{C}} = \bigcup_{k\in\mathbb{N}} \mathsf{MATIME}_{\mathscr{C}}[n^k]$. We also define

$$\mathsf{MATIME}[T(n)]_{\mathsf{AC}^0} = \bigcup_{d\in\mathbb{N}} \mathsf{MATIME}[T(n)]_{\mathsf{AC}^0_d} \text{ and } \mathsf{MATIME}[T(n)]_{\mathsf{ACC}^0} = \bigcup_{d,m\in\mathbb{N}} \mathsf{MATIME}[T(n)]_{\mathsf{AC}^0_d[m]}.$$

$\mathsf{MA}_{\mathsf{AC}^0}$ and $\mathsf{MA}_{\mathsf{ACC}^0}$ are defined similarly.

We note that the additional condition in Definition 3.4.3 is weaker than requiring that V itself has a $T(n)$-size $\mathscr{C}$ circuit, so it applies to more languages.

3.4.1 $\mathsf{MA} \cap \mathsf{coMA}$ and $\mathsf{NP} \cap \mathsf{coNP}$ Algorithms

We also introduce convenient definitions of $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T(n)]$ and $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$ algorithms, which simplifies the presentation.

Definition 3.4.4. *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A language L is in $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T(n)]$, if there is a deterministic algorithm $V(x,y,z)$ (V is called the predicate) such that:*

- *V takes three strings x, y, z such that $|x| = n$, $|y| = |z| = T(n)$ as inputs (y is the witness and z is the collection of random bits), runs in $O(T(n))$ time, and outputs an element from $\{0,1,\perp\}$.*
- *(Completeness) For every $x \in \{0,1\}^*$, there exists a y such that*

$$\Pr_z[A(x,y,z) = L(x)] = 1.$$

- *(Soundness) For every $x \in \{0,1\}^*$ and every y,*

$$\Pr_z[A(x,y,z) = 1 - L(x)] \leq 1/3.$$

Moreover, we say that $L \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}_{\mathscr{C}}[T(n)]$, if L further satisfies the following condition:

- *For every $x \in \{0,1\}^n$ and $y \in \{0,1\}^{T(n)}$, $V(x,y,\cdot)$ (the restriction of V to the randomness part) has a $T(n)$-size $\mathscr{C}$ circuit.*

Remark 3.4.5. $(\mathsf{MA} \cap \mathsf{coMA})$ *(resp.* $(\mathsf{MA} \cap \mathsf{coMA})_{\mathscr{C}}$*) languages with advice are defined similarly, with V being an algorithm with the corresponding advice.*

Definition 3.4.6. *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A language L is in* $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T(n)]$*, if there is an algorithm $V(x,y)$ (which is called the predicate) such that:*

- *A takes two inputs x, y such that $|x| = n$, $|y| = T(n)$ (y is the witness), runs in $O(T(n))$ time, and outputs an element from $\{0,1,\perp\}$.*
- *(Completeness) For all $x \in \{0,1\}^*$, there exists a y such that*

$$A(x,y) = L(x).$$

- *(Soundness) For all $x \in \{0,1\}^*$ and all y,*

$$A(x,y) \neq 1 - L(x).$$

Remark 3.4.7. $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T(n)]$ *languages with advice are defined similarly, with A being an algorithm with the corresponding advice.*

3.5 Pseudorandom Generators

We will deal with different types of pseudorandom generators (PRG) throughout Part II. In the following, we recall their definitions.

3.5.1 PRGs and NPRGs

Let $r, m \in \mathbb{N}$ and $\varepsilon \in (0,1)$, and let $\mathcal{H} \subseteq \mathcal{F}_{m,1}$ be a set of functions. We say $G \in \mathcal{F}_{r,m}$ is a PRG for $\mathcal{H}$ with error ε, if for every $D \in \mathcal{H}$

$$\left| \Pr_{z \in_R \{0,1\}^r}[D(G(z)) = 1] - \Pr_{z \in_R \{0,1\}^m}[D(z) = 1] \right| \leq \varepsilon.$$

We also call r the *seed length* of G.

We also need the notion of *non-deterministic PRGs*, which is defined as below.

Let $w \in \mathbb{N}$. We say a pair of function $G = (G^{\mathsf{P}}, G^{\mathsf{W}})$ such that $G^{\mathsf{P}} \in \{0,1\}^w \times \{0,1\}^r \to \{0,1\}^m$ and $G^{\mathsf{W}} \in \mathcal{F}_{w,1}$ is an NPRG for $\mathcal{H}$ with error ε, if the following hold:

1. For every $u \in \{0,1\}^w$, if $G^{\mathsf{W}}(u) = 1$, then $G^{\mathsf{P}}(u, \cdot)$ is a PRG for $\mathcal{H}$ with error ε.
2. There exists $u \in \{0,1\}^w$ such that $G^{\mathsf{W}}(u) = 1$.

Here, we call r the *seed length* of G and w the *witness length* of G.

Although NPRG in general does not compute *the same PRG* for different witness u (*i.e.*, $G^{\mathsf{P}}(u_1, \cdot)$ and $G^{\mathsf{P}}(u_2, \cdot)$ can be two different PRG for $\mathcal{H}$), it is still useful for many tasks such as the derandomization of MA. We remark that the concept of NPRG is already implicit in [IKW02]. Our definition is from the journal version of [Che19].[3]

3.5.2 Family of PRGs and NPRGs

Most of the time we will be interested in a *family of PRGs (NPRGs)* $G = \{G_n\}$ that fools a family of sets of functions $\mathcal{H} = \{\mathcal{H}_n\}$. In this case, for seed $r \colon \mathbb{N} \to \mathbb{N}$, error $\varepsilon \colon \mathbb{N} \to (0,1)$, output length $m \colon \mathbb{N} \to \mathbb{N}$ and witness length $w \colon \mathbb{N} \to \mathbb{N}$, we say $G = \{G_n\}$ is a PRG (resp. NPRG) family for $\mathcal{H} = \{\mathcal{H}_n\}$ if for every $n \in \mathbb{N}$, (1) $\mathcal{H}_n \subseteq \mathcal{F}_{m(n),1}$ (2) G_n is a PRG (resp. NPRG) for $\mathcal{H}_n$ with error $\varepsilon(n)$, seed length $r(n)$ (and witness length $w(n)$ for G being an NPRG). When the meaning is clear, sometimes we just say G is a PRG (resp. NPRG) instead of a PRG (resp. NPRG) family.

Let $\mathcal{I} \subseteq \mathbb{N}_{\geq 1}$. We also say G is an PRG with range $\mathcal{I}$ (resp. NPRG with range $\mathcal{I}$) for $\mathcal{H}$ if the above two conditions hold for every $n \in \mathcal{I}$. When $\mathcal{I}$ is an infinite set, we also say G is an i.o. PRG (resp. i.o. NPRG) family for $\mathcal{H}$.

We say that a PRG $G = \{G_n\}$ is computable in $T \colon \mathbb{N} \to \mathbb{N}$ time, if there is a uniform algorithm $A \colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}$ such that A_n (meaning the first input of A is fixed to n) computes G_n in $T(n)$ time. Similarly, we say an NPRG $G = \{G_n\}$ is computable in $T \colon \mathbb{N} \to \mathbb{N}$ time, if there are two uniform algorithms $A^{\mathsf{P}} \colon \mathbb{N} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ and $A^{\mathsf{W}} \colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}$ such that A^{P}_n computes G^{P}_n and A^{W}_n computes G^{W}_n, both in $T(n)$ time. Note that a $T(n)$-time computable NPRG G also has witness length at most $T(n)$. So if we do not specify the witness length parameter, it is by default the running time T.

We can similar define PRG and NPRG computable with $\alpha \colon \mathbb{N} \to \mathbb{N}$ bits of advice, by allowing algorithm A_n (resp. A^{P}_n and A^{W}_n) to use $\alpha(n)$ bits of advice. (Note the in the case of NPRG, the advice for A^{P}_n and A^{W}_n are the same.)

[3] See http://www.mit.edu/~lijieche/Che19-journal-version.pdf for the draft.

3.5.3 Derandomization of $\mathsf{MATIME}_{\mathscr{C}}[T(n)]$ from NPRGs for $\mathscr{C}$

Now we show that NPRGs are enough for the derandomization of $\mathsf{MA}_{\mathscr{C}}[T(n)]$.

Lemma 3.5.1. *Let $\mathcal{I} \subseteq \mathbb{N}_{\geq 1}$. Suppose there is an $s(n)$-seed-length, $w(n)$-witness-length NPRG G for $T(n)$-size $\mathscr{C}$ circuits with range $\mathcal{I}$, error $1/10$, and running time $T_G(n)$. Then, for every $L \in \mathsf{MATIME}_{\mathscr{C}}[T(n)]$, there is an $L' \in \mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$ such that L and L' agree on all n-bit inputs for every $n \in \mathcal{I}$.*

Proof. Let $L \in \mathsf{MA}_{\mathscr{C}}[T(n)]$ and $V(x, y, r)$ be the corresponding $\mathsf{MATIME}[T(n)]_{\mathscr{C}}$ verifier. We construct a new deterministic verifier V' as follows:

- V' takes both $y \in \{0,1\}^{T(n)}$ and $u \in \{0,1\}^{w(n)}$ as witness. (*i.e.*, V' takes $T(n) + w(n)$ bits as witness.)
- V' accepts if and only if $G^{\mathsf{W}}(u) = 1$ and $\Pr_{r \in \{0,1\}^s}[V(x, y, G^{\mathsf{P}}(u, r)) = 1] \geq 1/2$.

It is then easy to verify that V' is the desired $\mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$ verifier for L when the input length $n \in \mathcal{I}$, which completes the proof. ∎

There are two useful special cases of Lemma 3.5.1: (1) when $\mathcal{I} = \mathbb{N}_{\geq 1}$ (*i.e.*, G is an NPRG), then we have $\mathsf{MATIME}_{\mathscr{C}}[T(n)] \subseteq \mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$ and (2) when $\mathcal{I}$ is an infinite set, then we have $\mathsf{MATIME}_{\mathscr{C}}[T(n)] \subseteq \text{i.o.-}\mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$.

Remark 3.5.2. *If the NPRG mentioned in Lemma 3.5.1 requires $\alpha(n)$ bits of advice to compute (for G_n). Then the resulting* NTIMEGUESS *simulations also need $\alpha(n)$ bits of advice on n-bit inputs.*

Chapter 4

Overview and Organization of Part II

The goal of this chapter is to highlight the dependency of our results, and give a streamlined presentation of our new almost-everywhere lower bounds and average-case lower bounds.[1]

In Section 4.1, we will first provide an overview of Williams' original proofs from [Wil11] and [MW18]. We will give a presentation that is somewhat different than the original presentation of [Wil11]. Our presentation will be centered around the concept of *easy witness lemmas (EWL)*, which converts *witness lower bounds* into *circuit lower bounds for nondeterministic time classes*. Thus, the proof is naturally decomposed into two parts: first, we prove a witness lower bound; second, we apply an easy witness lemma to convert the obtained witness lower bound into a circuit lower bound for nondeterministic time classes.

In Section 4.2 (corresponding to Section 1.5), we give an overview of our results on witness and E^{NP} lower bounds, this part highlights average-case witness lower bounds and almost-everywhere E^{NP} lower bounds.

Then in Section 4.3 (corresponding to Section 1.6), we give an overview of our results on circuit lower bounds for nondeterministic time classes, which follows a *derandomization-centric perspective*. Roughly speaking, our proofs are centered around *derandomization of Merlin-Arthur classes*. Our lower bounds for non-deterministic time classes can also be naturally decomposed into two parts: first, a circuit lower bound for certain subclass of Merlin-Arthur protocols; second, a derandomization of the same subclass into a nondeterministic time class, which follows from average-case witness lower bounds mentioned above. To obtain average-case circuit lower bound for nonde-

[1] This is different from the purpose of the introduction (Chapter 1), which is to survey the known results and compare them properly with previous works. It suffices to read the introduction if you are only curious about *what we proved*. This chapter instead aims to help you to understand the overall proof structure, which spans several papers ([Che19, CR20, CLW20, CL21]).

terministic time classes, it amounts to start from average-case lower bounds for Merlin-Arthur classes, together with a careful win-win analysis.

Finally, we remark that our results from Section 1.4 (improved connection between non-trivial circuit-analysis algorithms and circuit lower bounds) are not covered by the streamlined overview above since they are somewhat independent. We will present these results in Chapter 5. In terms of techniques, we will introduce PCPs of proximity into the algorithmic method, which will be used very frequently in chapters after Chapter 5. Strictly speaking, later chapters do not depend on any results from Chapter 5 since they apply PCPs of proximity in different ways, but we recommend reading Section 5.1 to gain some intuitions on how PCPs of proximity enter the picture naturally while trying to improve the algorithmic method.

4.1 An Overview of Williams' EWL-centered Proofs

Notation. Let $s\colon \mathbb{N} \to \mathbb{N}$ and $\mathscr{C}$ be a concrete circuit class. We say that NE *does not admit* $s(n)$*-size* $\mathscr{C}$ *witnesses*, if there exists a verifier $V(x,y)$ that takes input $|x| = n$, $|y| = 2^n$ and runs in $2^{O(n)}$ time, such that for infinitely many $x \in \{0,1\}^*$, the following hold:

1. there exists $y \in \{0,1\}^{2^{|x|}}$ such that $V(x,y) = 1$;
2. for every $y \in \{0,1\}^{2^{|x|}}$ such that $V(x,y) = 1$, it follows that $\mathsf{func}(y)$ has no $s(n)$-size $\mathscr{C}$ circuit.

Moreover, we say *unary* NE *does not admit* $s(n)$*-size* $\mathscr{C}$ *witnesses*, if for some verifier V and for infinitely many $n \in \mathbb{N}_{\geq 1}$, the above two conditions hold for $x = 1^n$. Clearly, this is a stronger statement and it implies NE does not admit $s(n)$-size $\mathscr{C}$ witnesses.

4.1.1 Overview for The proof of Theorem 1.1.1

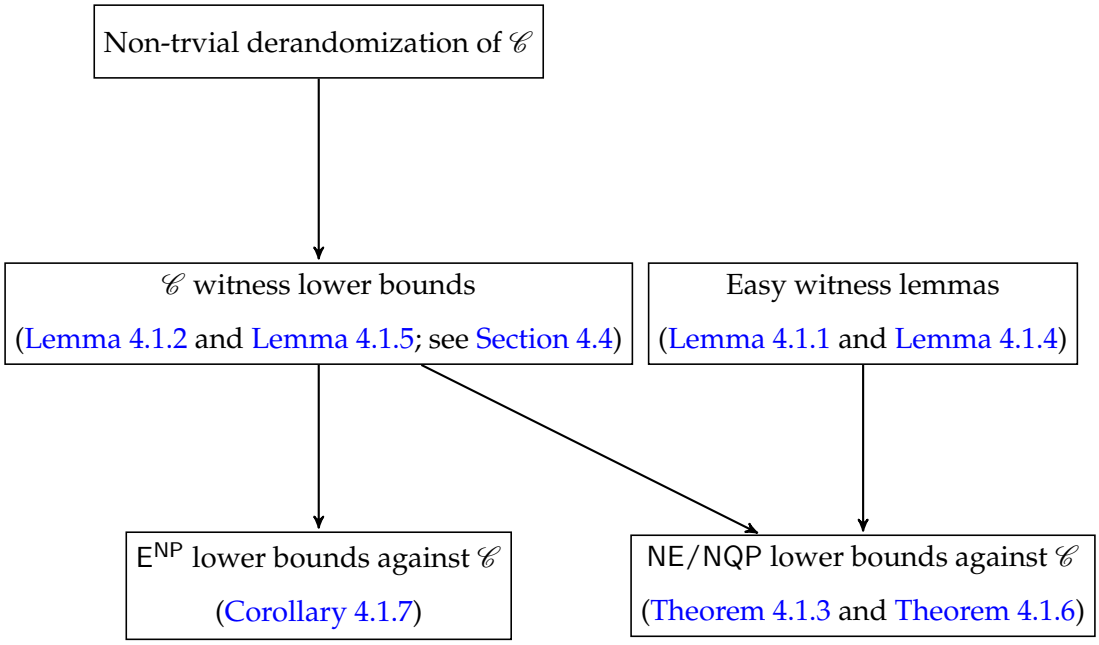


Figure 4-1: High-level structure of Williams' EWL-centered proofs

The easy witness lemma of [IKW02] says the following:

Lemma 4.1.1 (EWL for NE [IKW02])**.** *If* NE *does not admit* $\mathrm{poly}(n)$*-size* $\mathscr{C}$ *witnesses*[2]*, then* $\mathsf{NE} \not\subset \mathscr{C}$*.*

Indeed, the original statement says the contrapositive of Lemma 4.1.1: if $\mathsf{NE} \subseteq \mathscr{C}$, then NE admits polynomial-size $\mathscr{C}$ witnesses. Hence the name of easy witness lemma (*i.e.*, NE admits small circuit implies that NE admits *easy witnesses*).[3] We present it in this way since it makes very clear that witnesses lower bounds imply circuit lower bounds.

Williams gave a way to prove witness lower bounds from non-trivial derandomization. The below is in fact from [Wil13b], but the proof ideas are similar to that from [Wil10, Wil11].

Lemma 4.1.2 ([Wil13b])**.** *Let* $\mathscr{C}$ *be a typical concrete circuit classes. If* CAPP *for polynomial-size* $\mathsf{AC}^0_2 \circ \mathscr{C}$ *can be solved in* $2^n/n^{\omega(1)}$ *time, then unary* NE *does not admit* $\mathrm{poly}(n)$*-size* $\mathscr{C}$ *witnesses.*

Combining Lemma 4.1.1 and Lemma 4.1.2 immediately proves Theorem 1.1.1.

Theorem 4.1.3 (Formal version of Theorem 1.1.1)**.** *Let* $\mathscr{C}$ *be a typical concrete circuit class. If there is a* $2^n/n^{\omega(1)}$*-time algorithm for* CAPP *of* $\mathrm{poly}(n)$*-size* n*-input* $\mathsf{AC}^0_2 \circ \mathscr{C}$ *circuits, then* $\mathsf{NE} \not\subset \mathscr{C}$*.*

[2] More precisely, NE does not admit n^k-size $\mathscr{C}$ witnesses for every $k \in \mathbb{N}_{\geq 1}$.
[3] [IKW02] indeed talks about NEXP instead of NE; we choose to work with NE since it simplifies the discussions.

We remark that in both Lemma 4.1.2 and Theorem 4.1.3, CAPP can be replaced by Gap-UNSAT, which is weaker than both CAPP and SAT. We will only work with CAPP for simplicity.

4.1.2 Overview for the Proof of Theorem 1.1.2

To obtain lower bounds for NQP, [MW18] proved the following easy witness lemma. Again, we state their lemma in the contrapositive.

Lemma 4.1.4 (EWL for NQP [MW18])**.** *Let $\mathscr{C}$ be a typical concrete circuit classes and $\varepsilon \in (0,1)$. If* NE *does not admit 2^{n^ε}-size $\mathscr{C}$ witnesses, then* $\mathsf{NQP} \not\subset \mathscr{C}$.

We note that the above lemma is in fact weaker than the easy witness lemma for NQP in [MW18], but we observe that it still suffices for circuit lower bounds for NQP. To obtain $\mathsf{NQP} \not\subset \mathscr{C}$, we need the following adaption of Lemma 4.1.2.

Lemma 4.1.5 ([Wil13b])**.** *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. If* CAPP *for 2^{n^ε}-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ can be solved in $2^n/n^{\omega(1)}$ time, then unary* NE *does not admit $2^{n^\varepsilon/2}$-size $\mathscr{C}$ witnesses.*

We give a proof of Lemma 4.1.5 in Section 4.4.

Now, combining Lemma 4.1.4 and Lemma 4.1.5, we immediately prove the NQP part of Theorem 1.1.2.

Theorem 4.1.6. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. If* CAPP *for 2^{n^ε}-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ can be solved in $2^n/n^{\omega(1)}$ time, then* $\mathsf{NQP} \not\subset \mathscr{C}$.

We also remark that a direct corollary of Lemma 4.1.5 is that lower bounds for E^{NP} follows from non-trivial derandomization.

Corollary 4.1.7. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. If* CAPP *for 2^{n^ε}-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ can be solved in $2^n/n^{\omega(1)}$ time, then E^{NP} does not admit $2^{n^\varepsilon/2}$-size $\mathscr{C}$ circuits.*

Proof. Assuming that unary NE does not admit $s(n)$-size $\mathscr{C}$ witnesses and letting V be the corresponding verifier, we can construct a hard language $L \in \mathsf{E}^{\mathsf{NP}}$ as follows: on an input x with length n, we use a binary search and an NP oracle to find the lexicographically first string $y \in \{0,1\}^{2^n}$ such that $V(1^n, y) = 1$ (if no such y exists, we set $y = 0^{2^n}$); we then set $L(x) = \mathsf{func}(y)(x)$. It is straightforward to verify that for infinitely many $n \in \mathbb{N}_{\geq 1}$, L_n does not have $s(n)$-size $\mathscr{C}$ circuits. ∎

4.2 Witness Lower Bounds and E^{NP} Lower Bounds

In this section we will first discuss the proof structure of our general connection between non-trivial derandomization and lower bounds. We will use $\mathscr{C}$ to denote a typical concrete circuit class throughout this section. It may help to think of it as $\mathsf{AC}^0_d[6]$ for some large constant $d \in \mathbb{N}_{\geq 1}$.

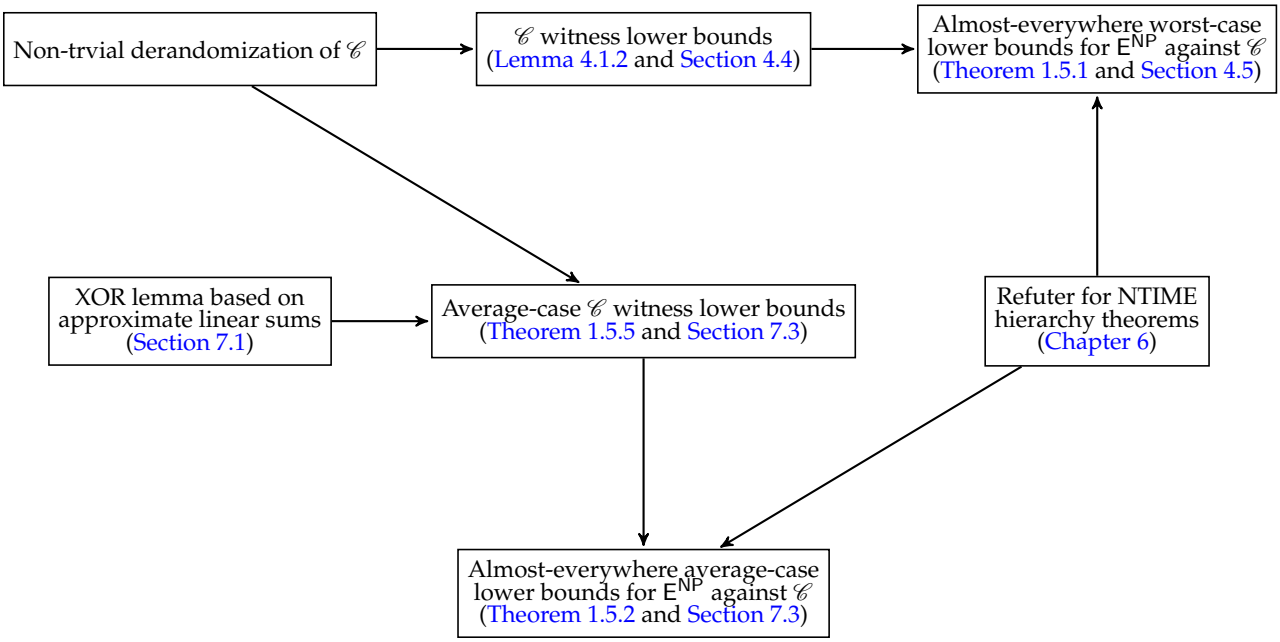


Figure 4-2: High-level structure of our results for witness lower bounds and E^{NP} lower bounds

Notation. Let $s\colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \to \mathbb{N}$ and $\mathscr{C}$ be a concrete circuit class. We say that *unary* NE *does not admit* $(1/2 + \varepsilon(n))$*-approximate* $s(n)$*-size* $\mathscr{C}$ *witnesses*, if there exists a verifier $V(x, y)$ that takes input $|x| = n$, $|y| = 2^n$ and runs in $2^{O(n)}$ time, such that for infinitely many $n \in \mathbb{N}_{\geq 1}$, the following hold:

1. there exists $y \in \{0,1\}^{2^n}$ such that $V(1^n, y) = 1$;
2. for every $y \in \{0,1\}^{2^n}$ such that $V(x, y) = 1$, it follows that $\mathsf{func}(y)$ cannot be $(1/2 + \varepsilon(n))$-approximated by $s(n)$-size $\mathscr{C}$ circuit.

Average-case witness lower bounds. We first strengthen Lemma 4.1.5 to the average case, by showing that non-trivial derandomization (with inverse-circuit-size error) indeed implies average-case witness lower bound.

Reminder of Theorem 1.5.5. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ such that unary NE does not admit $(1/2 + 2^{-n^\delta})$-approximate 2^{n^δ}-size $\mathscr{C}$ witnesses.*

The key technical ingredients behind the proof of Theorem 1.5.5 is a new XOR Lemma based on *approximate linear sums*, and a careful use of *PCP of proximity*. The actual proof is quite complicated and see Section 7.3 for details.

Almost-everywhere lower bounds. We next strengthen Corollary 4.1.7 to almost-everywhere hardness. Note that Corollary 4.1.7 only states that there is hard language $L \in \mathsf{E}^{\mathsf{NP}}$ such that L_n is hard for infinitely many input length $n \in \mathbb{N}_{\geq 1}$. This is inherent from the proof of Lemma 4.1.5, which crucially used NTIME hierarchy theorem. One can observe from the proof that one would actually need an almost-everywhere NTIME hierarchy theorem (*e.g.*, $\mathsf{NTIME}[2^n] \not\subset$ i.o.-$\mathsf{NTIME}[2^n/n]$) to obtain almost-everywhere hardness. But proving an a.e. NTIME hierarchy theorem is notoriously open.

We manage to overcome the issue above via a *refuter*, which is a very general framework that can convert various i.o. E^{NP} lower bounds into a.e. ones. Formally, we have:

Reminder of Theorem 1.5.1. *Let $\mathscr{C}$ be a typical concrete circuit class and $S(n) \leq 2^{o(n)}$ be a non-decreasing size parameter. There is a universal constant $K \in \mathbb{N}_{\geq 1}$ such that if CAPP for $(n^K \cdot S(n))$-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ time, then there is an $L \in \mathsf{E}^{\mathsf{NP}}$ such that L_n does not have $\mathscr{C}$ circuits of size $S(n/2)$ for all sufficiently large $n \in \mathbb{N}_{\geq 1}$.*

Next, combining proof ideas behind Theorem 1.5.5 and Theorem 1.5.1, we are able to show E^{NP} lower bounds that are *simultaneously* strongly average-case and almost-everywhere from non-trivial derandomization.

Reminder of Theorem 1.5.2. *Let $\mathscr{C}$ be a typical and concrete circuit class. Suppose there is an $\varepsilon \in (0,1)$ such that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a language $L \in \mathsf{E}^{\mathsf{NP}}$ and a constant $\delta \in (0,1)$ such that, for every sufficiently large $n \in \mathbb{N}_{\geq 1}$, L_n cannot be $(1/2 + 2^{-n^\delta})$-approximated by $\mathscr{C}$ circuits of size 2^{n^δ}.*

In Section 4.5, we first introduce our refuter construction and intuitively explain how it converts infinitely often lower bounds into almost-everywhere ones, and then provide a detailed proof of Theorem 1.5.1. The proof of Theorem 1.5.2 follows a similar structure, but is much more involved; see Section 7.3 for details.

$\mathsf{ACC}^0 \circ \mathsf{THR}$ **lower bounds.** Recall that [Wil14a] gave the following algorithms for analyzing $\mathsf{ACC}^0 \circ \mathsf{THR}$: for every constant $d, m \in \mathbb{N}_{\geq 1}$, there is a constant $\varepsilon \in (0,1)$ that only depends on d and m, such that #SAT of 2^{n^ε}-size $\mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuits can be solved in deterministic 2^{n-n^ε} time. Combining this algorithm with Theorem 1.5.2, we immediately have Corollary 1.5.3 (restated below).

Reminder of Corollary 1.5.3. *For every $d, m \in \mathbb{N}_{\geq 1}$, there is an $\varepsilon \in (0,1)$ and a language $L \in \mathsf{E}^{\mathsf{NP}}$ such that L_n cannot be $(1/2 + 2^{-n^\varepsilon})$-approximated by $\mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuits of 2^{n^ε} size, for every sufficiently large $n \in \mathbb{N}_{\geq 1}$.*

4.3 NQP Lower Bounds

In this section we will present the structure of our proofs for the NQP case of Theorem 1.6.1.[4]

Derandomization of $\mathsf{MA}_{\mathscr{C}}$. Recall that $\mathsf{MA}_{\mathscr{C}}$ denotes the sub-class of Merlin-Arthur protocols whose verification can be simulated by $\mathscr{C}$ circuits for every possible witnesses, and NPRG is a weaker version of PRG that suffices to derandomize Merlin-Arthur protocols; see Section 3.4 and Section 3.5 for formal definitions.

An acute reader may have already observed that an average-case witness lower bound immediately gives a construction of NPRG. For any typical circuit class $\mathscr{C}$, recall that the well-known Nisan-Widgerson PRG construction can convert a truth-table f that is average-case hard against $\mathscr{C} \circ \mathsf{AC}^0_2$ into a PRG fooling $\mathscr{C}$ circuits (see Lemma 7.4.1; the extra AC^0_2 at the bottom comes from the overhead of the NW reconstruction). Our NPRG can simply use the verifier V (that corresponds to the average-case witness lower bound) to verify an average-case hard truth-table y, and then apply NW to f. Formally, we have:

Reminder of Theorem 1.5.6. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ and an infinity often nondeterministic PRG for 2^{n^δ}-size $\mathscr{C}$ circuits with error 2^{-n^δ}, seed-length $\mathrm{poly}(n)$, and $2^{\mathrm{poly}(n)}$ running time. Consequently, $\mathsf{MA}_{\mathscr{C}} \subseteq$ i.o.-$\mathsf{NTIME}[2^{\log^\beta n}]$ for some $\beta \in \mathbb{N}_{\geq 1}$.*

[4] In this thesis, we aim to prove a weaker version with a proof that we believe is easier to understand, yet still implies strongly average-case lower bounds for NQP against ACC^0. For a full proof of Theorem 1.6.1, see [CR21].

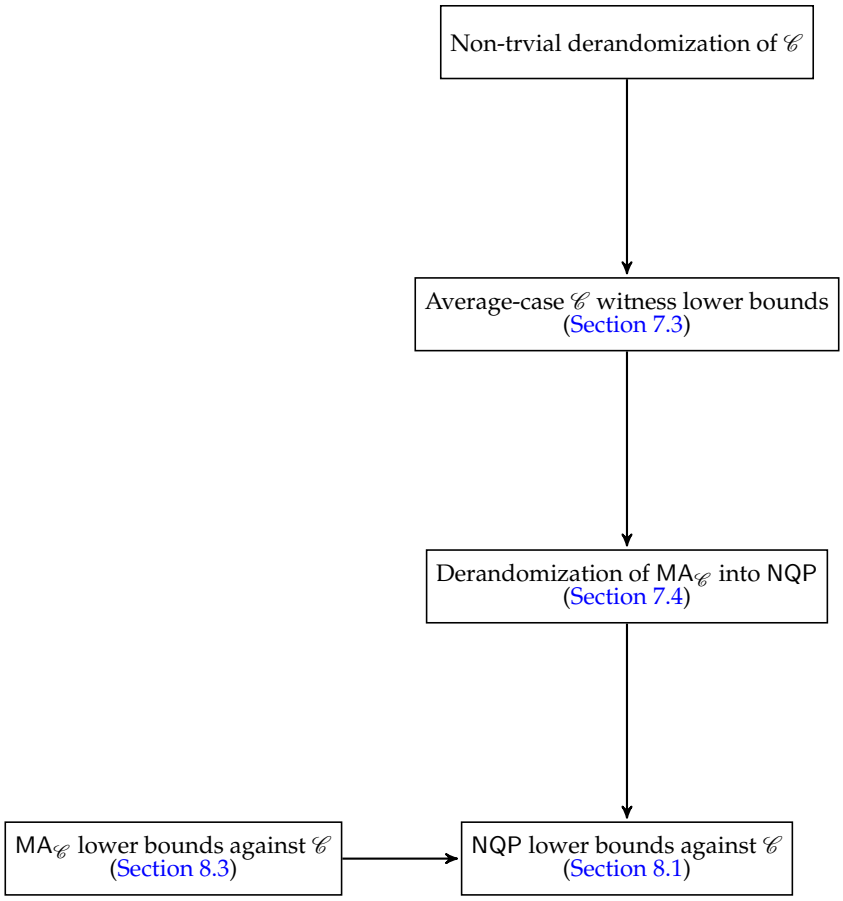


Figure 4-3: High-level structure of our results for NTIME lower bounds

4.3.1 NQP Lower Bounds via Derandomization

In order to apply Theorem 1.5.6 to get circuit lower bounds for NQP. We will (roughly speaking) prove an $\mathsf{MA}_{\mathscr{C}}$ lower bounds against $\mathscr{C}$. In more details, we will prove the following theorem.

Theorem 4.3.1. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a universal constant $d_{\mathsf{v}} \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, it holds that*

$$\left(\mathsf{MA}_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2]\circ\mathscr{C}}\right)_{/1} \not\subset \mathscr{C}\text{-}\mathsf{SIZE}[n^a].$$

It seems that assuming $\widetilde{\mathsf{CAPP}}$ for $\mathsf{AC}^0_{d_{\mathsf{v}}+1}[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$,[5] we will be able to derandomize the hard $\left(\mathsf{MA}_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2]\circ\mathscr{C}}\right)_{/1}$ language from Theorem 4.3.1 into $\mathsf{NTIME}[2^{\log^\beta n}]$, which *should* imply NQP has no polynomial-size $\mathscr{C}$ circuits. But however, there is a huge caveat that we explain below.

[5] We recommend reader to think of $\mathscr{C} = \mathsf{AC}^0_d[6]$ for some large constant $d \in \mathbb{N}_{\geq 1}$, then we only need non-trivial $\widetilde{\mathsf{CAPP}}$ for $\mathsf{AC}^0_{d+O(1)}[6]$, which follows from [Wil11].

Retaining the hardness after derandomization. The key issue here is that, the $\mathsf{MA}_{/1}$ hard language L from Theorem 4.3.1 is only *infinitely often* hard, meaning that we only know for infinitely many input lengths $n \in \mathbb{N}_{\geq 1}$, L_n is hard against $\mathscr{C}$ circuits. Also, the derandomization of Theorem 1.5.6 works *infinitely often* too, in the sense that our new NQP language L' only agrees with the hard language L on infinitely many input lengths n. Let I_{hard} and I_{derand} be the input lengths that L_n is hard and $L'_n = L_n$, respectively. We see that it is entirely possible that $I_{\mathsf{hard}} \cap I_{\mathsf{derand}} = \emptyset$, meaning that our new NQP language L' is not hard at all.

Following [MW18], the idea is to make both I_{hard} and I_{derand} larger so that they *must intersect at infinitely many input lengths*.

In more details, we first strength Theorem 1.5.6 by showing the following theorem.

Theorem 4.3.2. *Let $\mathscr{C}$ be a typical concrete circuit class. Suppose that there is a constant $\varepsilon \in (0,1)$ and an infinity often nondeterministic PRG for 2^{n^ε}-size $\mathscr{C}$ circuits with $\mathrm{poly}(n)$ seed-length and $2^{\mathrm{poly}(n)}$ running time.*

Then, there is a constant $\beta \in \mathbb{N}_{\geq 1}$ that only depends on ε such that for every $L \in (\mathsf{MA}_{\mathscr{C}})_{/1}$ and $c \in \mathbb{N}_{\geq 1}$, there is an $L' \in \mathsf{NTIME}[2^{\log^\beta n}]_{/O(\log\log n)}$ such that for infinitely many $n \in \mathbb{N}$, for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

Combining Theorem 1.5.6 and Theorem 4.3.2, we immediately have the following strengthening of Theorem 1.5.6.

Corollary 4.3.3. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Assuming the hypothesized $\widetilde{\mathsf{CAPP}}$ algorithm from Theorem 1.5.6, the conclusion of Theorem 4.3.2 holds.*

Roughly speaking, Corollary 4.3.3 says that by allowing $O(\log\log n)$ bits of advice, we can enlarge I_{derand} from a set of infinitely many integers to a union of infinitely many segments of the from $[n, n^c]$, where c is a constant *of our choice*.

Next, we have the following strengthening of Theorem 4.3.1, which fits perfectly with the larger I_{derand} above.

Theorem 4.3.4. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a universal constant $d_{\mathsf{v}} \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, there is a constant $c \in \mathbb{N}_{\geq 1}$ and a language $L \in \left(\mathsf{MA}_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2]\circ\mathscr{C}}\right)_{/1}$ such that, for all large enough $n \in \mathbb{N}_{\geq 1}$, there exists $m \in [n, n^c]$ such that L_m does not have m^a-size $\mathscr{C}$ circuits.*

Essentially, it says that we can enlarge I_{hard} to be a *hitting set* for all segment $[n, n^c]$: for every large enough $n \in \mathbb{N}_{\geq 1}$, $[n, n^c] \cap I_{\mathsf{hard}} \neq \emptyset$. This fits perfectly with the I_{derand} above, which consists of infinitely many segments of the form $[n, n^c]$. Hence, we have that $I_{\mathsf{hard}} \cap I_{\mathsf{derand}}$ is an infinite set.

Therefore, combining Corollary 4.3.3 and Theorem 4.3.4, we immediately have the following theorem.[6]

Theorem 4.3.5. *Let $\mathscr{C}$ be a typical concrete circuit class, $\varepsilon \in (0,1)$, and $d_{\mathsf{v}} \in \mathbb{N}_{\geq 1}$ be the constant from Theorem 4.3.4. Suppose that $\widetilde{\mathsf{CAPP}}$ of $2^{n^{\varepsilon}}$-size $\mathsf{AC}^0_{d_{\mathsf{v}}+1}[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in $2^{n-n^{\varepsilon}}$ time. Then NQP does not have polynomial-size $\mathscr{C}$ circuits.*

As a corollary from the theorem above and Williams' #SAT algorithm for ACC^0, we immediately have that

$$(\mathsf{L1}) \ : \ \mathsf{NQP} \not\subset \mathsf{AC}^0_{d_\star}[m_\star] \text{ for every } d_\star, m_\star \in \mathbb{N},$$

Getting $\mathsf{NQP} \not\subset \mathsf{ACC}^0$. Interestingly, (L1) above is different from

$$(\mathsf{L2}) \ : \ \mathsf{NQP} \not\subset \mathsf{ACC}^0.[7]$$

This issue occurs in [MW18] as well, as a direct application of Theorem 4.1.6 also only proves (L1). Nonetheless, [MW18] resolved this issue by proving that (L1) implies (L2) by a simple win-win analysis.[8]

4.3.2 Average-case Lower Bounds for NQP via Randomized Encodings

Finally, we strengthen Theorem 4.3.5 to also give average-case lower bounds. Given the discussions above, it seems we can simply strengthen the $\mathsf{MA}_{\mathscr{C}}$ lower bounds from Theorem 4.3.4 to the average case, and then our derandomization from Corollary 4.3.3 would immediately imply average-case lower bounds for NQP.

We are indeed able to strengthen to Theorem 4.3.4 to an average-case lower bound, but only with very weak inapproximability.

Theorem 4.3.6. *Let $\mathscr{C}$ be a typical concrete circuit class. There are universal constants $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, there is a constant $c \in \mathbb{N}_{\geq 1}$ and a language $L \in \left(\mathsf{MA}_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2]\circ\mathscr{C}}\right)_{/1}$ such that, for*

[6] A direct application of Corollary 4.3.3 yields a hard language in $\mathsf{NQP}_{/O(\log\log n)}$ instead of just NQP. Those advice can nonetheless be removed via a straightforward *enumeration trick* (from [COS18]); see Section 8.1.5 for details.

[7] (L2) implies that there is *fixed* language $L \in \mathsf{NQP}$ such that $L \notin \mathsf{AC}^0_{d_\star}[m_\star]$ for every $d_\star, m_\star \in \mathbb{N}$, which *a priori* looks stronger than (L1).

[8] If $\mathsf{P} \not\subset \mathsf{ACC}^0$, clearly (L2) is true. If $\mathsf{P} \subseteq \mathsf{ACC}^0$, then $\mathsf{P}_{/\mathsf{poly}}$ collapsed to $\mathsf{AC}^0_{d_\star}[m_\star]$ for some fixed $d_\star$ and $m_\star$, hence (L1) implies $\mathsf{NQP} \not\subset \mathsf{P}_{/\mathsf{poly}}$.

all large enough $n \in \mathbb{N}_{\geq 1}$, there exists $m \in [n, n^c]$ such that L_m cannot be $(1 - m^{-\tau})$-approximated by m^a-size $\mathscr{C}$ circuits.[9]

Combining with Corollary 4.3.3, we immediately have the following theorem.

Theorem 4.3.7. *Let $\mathscr{C}$ be a typical concrete circuit class, $\varepsilon \in (0,1)$, and $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ be the constants from Theorem 4.3.6. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AC}^0_{d_{\mathsf{v}}+1}[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in 2^{n-n^ε} time. Then NQP cannot be $(1 - n^{-\tau})$-approximated by polynomial-size $\mathscr{C}$ circuits.*

To improve the inapproximability of Theorem 4.3.7 from $1 - n^{-\tau}$ to $1/2 + 1/\mathrm{poly}(n)$, we wish to perform some *mild-to-strong average-case hardness amplification* (*e.g.*, an XOR Lemma; see Lemma 8.1.14). Sadly, we currently *do not* have such an amplification for weak circuit classes such as ACC^0.

We overcome the issue above with a clever win-win argument, based on *randomized encodings* [IK02, AIK06] and *approximate linear sums*. Recall that for a $\mathsf{Sum} \circ \mathscr{C}$ circuit $L = \sum_{i \in [m]} \alpha_i \cdot C_i$ (where each C_i is a $\mathscr{C}$ circuit), the complexity of L is defined as $\max(\sum_{i \in [m]} |\alpha_i|, \sum_{i \in [m]} \mathsf{SIZE}(C_i))$. For a function $F \colon \{0,1\}^n \to \{0,1\}$, we say that F admits a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity S, if there exists a $\mathsf{Sum} \circ \mathscr{C}$ circuit L with complexity at most S, such that $|F(x) - L(x)| \leq \delta$ for every $x \in \{0,1\}^n$.

Using the techniques from randomized encodings, we are able to prove the following win-win result.

Lemma 4.3.8. *Let $\mathscr{C}$ be a typical concrete circuit class. One of the following holds:*

1. *There is a language $L \in \mathsf{P}$ such that for every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*
2. *There is a constant $\gamma \in \mathbb{N}_{\geq 1}$ such that every S-size formula admits a $\widetilde{\mathsf{Sum}}_{0.01} \circ \mathscr{C}$ circuit of complexity S^γ.*

The key observation now is that, an NPRG that fools $\mathscr{C}$ circuits with *a small error* also fools functions admitting low-complexity $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ circuits. Hence, now we are able to perform the following win-win analysis:

- Suppose Item (1) of Lemma 4.3.8 holds. Then it immediately follows that NQP cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.
- Otherwise, Item (2) of Lemma 4.3.8 holds. Then under the condition of Theorem 1.5.6, we would have i.o. NPRG for *formulas* (note that the i.o. NPRG from Theorem 1.5.6 indeed has a

[9] We remark that in Chapter 8 we indeed prove a stronger version where the hard language L is in $\left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2] \circ \mathscr{C}}\right)_{/1}$; see Theorem 8.1.1. We will also discuss why this is needed at the end of this subsection.

small error). Applying Theorem 4.3.2, this implies that we can now derandomize $\mathsf{MA}_{\mathsf{Formula}}$ as follows:

There is a constant $\beta \in \mathbb{N}_{\geq 1}$ such that for every $L \in (\mathsf{MA}_{\mathsf{Formula}})_{/1}$ and every $c \in \mathbb{N}_{\geq 1}$, there is an $L' \in \mathsf{NTIME}[2^{\log^{\beta} n}]_{/O(\log\log n)}$ such that for infinitely many $n \in \mathbb{N}$, for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

Note that formulas are closed under taking an $\mathsf{AC}^0[2]$ circuit at the top, we now can use the derandomization above together with Theorem 4.3.6 to obtain a NQP language that is $(1 - n^{-\tau})$-hard against polynomial-size formulas, which can then be amplified to $(1/2 + 1/\mathrm{poly}(n))$-hardness against formulas, using mild-to-strong average-case hardness amplification for formulas.

If we further assume that $\mathscr{C}$ is weaker than Formula, then now we also have that NQP cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

To summarize, we have the following theorem.

Theorem 4.3.9 (strong average-case lower bound for NQP via an additional win-win argument)**.** *Let $\mathscr{C}$ be a typical concrete circuit class that is weaker than formulas. There are two universal constants $d, \tau \in \mathbb{N}_{\geq 1}$ such that the following holds. Suppose that for some $\eta \in (0,1)$, CAPP of $2^{n^{\eta}}$-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in $2^{n-n^{\eta}}$ time. Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that $\mathsf{NTIME}[2^{\log^{\beta} n}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.*

Finally, we give a technical remark below.

Mild-to-strong hardness amplification requires $(\mathsf{N}\cap\mathsf{coN})\mathsf{QP}$ lower bounds. Recall that we wish to apply an XOR Lemma (Lemma 8.1.14) to the mildly average-case hard NQP language L from Theorem 4.3.7. This causes a subtle issue: $L^{\oplus 2}(x,y) \coloneqq L(x) \oplus L(y)$ may not be in NQP, since to certify $L^{\oplus 2}(x,y) = 1$, one needs to prove exactly one of $L(x)$ and $L(y)$ is 1 and the other one is 0; we cannot prove (say) $L(y) = 0$ since this requires $L \in \mathsf{coNQP}$.

To resolve this issue, we wish to get a mildly average-case hard language L from $(\mathsf{N}\cap\mathsf{coN})\mathsf{QP}$ instead of NQP. It is easy to see that the derandomization from Corollary 4.3.3 also derandomize $\mathsf{MA}\cap\mathsf{coMA}_{/1}$ languages into $(\mathsf{N}\cap\mathsf{coN})\mathsf{QP}_{/O(\log\log n)}$ languages. Hence, we manage to strengthen Theorem 4.3.6 so that the hard languages belong to $\mathsf{MA}\cap\mathsf{coMA}_{/1}$; see Theorem 8.1.1 for details.

4.4 Witnesses Lower Bounds from Non-trivial Derandomization

Now we give a proof of Lemma 4.1.5. Our proof below is from the journal version of [Che19], which follows the ideas from [Wil13b] with a new PCP construction of [BV14].

To prove Lemma 4.1.5, we need the following PCP construction from [BV14].

Lemma 4.4.1 ([BV14]). *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where $|x| = n$. Given $x \in \{0,1\}^n$, one can output in $\mathrm{poly}(n, \log T)$ time circuits $Q\colon \{0,1\}^r \to \{0,1\}^{rt}$ for $t = \mathrm{poly}(r)$ and $R\colon \{0,1\}^t \to \{0,1\}$ such that:*

Proof length. *$2^r \leq T \cdot \mathrm{polylog} T$.*

Completeness. *If there is a $y \in \{0,1\}^{T(n)}$ such that $M(x, y)$ accepts then there is a map $\pi\colon \{0,1\}^r \to \{0,1\}$ such that for all $z \in \{0,1\}^r$, $R(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = Q(z)$.*

Soundness. *If no $y \in \{0,1\}^{T(n)}$ causes $M(x, y)$ to accept, then for every map $\pi\colon \{0,1\}^r \to \{0,1\}$, at most $\frac{2^r}{n^{10}}$ many $z \in \{0,1\}^r$ have $R(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = Q(z)$.*

Complexity. *Q is a projection, i.e., each output bit of Q is a bit of input, the negation of a bit, or a constant. R is a* 3-CNF.

Now we are ready to prove Lemma 4.1.5. The proof idea is to use the assumed non-trivial CAPP algorithm to contradict a certain NTIME hierarchy theorem. For this purpose we will need to construct a slightly faster nondeterministic algorithm for a language $L \in \mathsf{NTIME}[T]$, which is described in Algorithm 4.1.

We summarize the important properties of Algorithm 4.1 below.

Lemma 4.4.2. *Let $T, L, \ell, K, S, \mathsf{VPCP}_x$ be stated as in Algorithm 4.1. Under the assumption from Algorithm 4.1, the following holds:*

1. *APCP runs in $\mathrm{poly}(n, \log T) + T/(\log T)^{\omega(1)}$ time and guesses $\mathrm{poly}(S(\ell))$ bits.*
2. *For every $x \in \{0,1\}^*$ such that $L(x) = 0$, it holds that $\mathsf{APCP}(x) = 0$.*[10]
3. *For every $x \in \{0,1\}^*$ such that $L(x) = 1$ and $\mathsf{APCP}(x) = 0$, it holds that*
   (a) *There exists $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$ such that*

$$\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^{\mathcal{O}}(r) = 1] = 1.$$

   (b) *For every $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$ satisfying the above, $\mathcal{O}$ does not have $S(\ell)$-size $\mathscr{C}$ circuits.*

[10] Since APCP is a nondeterministic algorithm, we use $\mathsf{APCP}(x) = 0$ to denote that APCP reject *all* guesses on the input x, and $\mathsf{APCP}(x) = 1$ to denote that APCP accepts *some* guesses on the input x.

Algorithm 4.1: The algorithm APCP attempting to speed up L

Setting: Let $T(n)$ be a time-constructive function, and $L \in \mathsf{NTIME}[T(n)]$. Let $\ell(n) = \log T + O(\log\log T)$ be the number of random bits from Lemma 4.4.1 when the running time is set to $T(n)$. Let $K \in \mathbb{N}_{\geq 1}$ be a sufficiently large universal constant.

Parameters: Let $S(n)$ be a size parameter.

Assumption: CAPP for $n^K \cdot S(n)$-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ time.

Input: $x \in \{0,1\}^n$

1 Apply Lemma 4.4.1 to L to obtain a $\mathrm{poly}(\ell)$-size AC^0_2 oracle circuit $\mathsf{VPCP}_x \colon \{0,1\}^\ell \to \{0,1\}$ that queries an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$;

2 **Guess** an $S(\ell)$-size ℓ-input $\mathscr{C}$ circuit C, and run the assumed algorithm for CAPP on VPCP^C_x to obtain an estimate $p \in [0,1]$;

`// Note that` VPCP^C_x `is an` $\ell^K \cdot S(\ell)$`-size` $\mathsf{AC}^0_2 \circ \mathscr{C}$ `from the complexity part of Lemma 4.4.1`

3 **if** $p > 1/2$ **then accept**;

4 **else reject**;

Before proving Lemma 4.4.2, we show it immediately imply Lemma 4.1.5. In fact, we will prove a more general version of it.

Theorem 4.4.3 (General version of Lemma 4.1.2 and Lemma 4.1.5)**.** *Let $\mathscr{C}$ be a typical concrete circuit class, $K \in \mathbb{N}_{\geq 1}$ be a sufficiently large constant, and $S(n)$ be a size parameter. If* CAPP *for $n^K \cdot S(n)$-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ time, then unary* NE *does not admit $S(n)$-size $\mathscr{C}$ witnesses.*

Proof. Let $T(n) = 2^n$ and L be a unary language such that $L \in \mathsf{NTIME}[T(n)] \setminus \mathsf{NTIME}[T(n)/n]$, whose existence is guaranteed by the non-deterministic time hierarchy theorem [Žák83]. Now we consider APCP with T, L, and size parameter S, and set K to be the constant K in Algorithm 4.1. Note that the assumption of Algorithm 4.1 is satisfied with our choice of $S(n)$ and K.

By Item (1) of Lemma 4.4.2, APCP runs in $\mathrm{poly}(n) + 2^n/n^{\omega(1)} < T(n)/n$ time, meaning that APCP cannot solve L. Hence, from the fact that L is a unary language and Item (2) of Lemma 4.4.2, it follows that for infinitely many $n \in \mathbb{N}_{\geq 1}$, we have $L(1^n) = 1$ and yet $\mathsf{APCP}(1^n) = 0$. We call these n good.

Now we are ready to define our verifier $V(x, y)$. Without loss of generality we can assume $\ell(n) = n + O(\log n)$ is an increasing function. For every $\alpha \in \mathbb{N}_{\geq 1}$, $V(1^\alpha, y)$ rejects immediately if there is no $n \in \mathbb{N}_{\geq 1}$ such that $\ell(n) = \alpha$. Otherwise, there is a unique $n \in \mathbb{N}_{\geq 1}$ such that $\ell(n) = \alpha$,

and $V(1^\alpha, y)$ accepts if and only if

$$\Pr_{r \in_R \{0,1\}^{\ell(n)}}[\mathsf{VPCP}_{1^n}^{\mathsf{func}(y)}(r) = 1] = 1.$$

Note that $V(x, y)$ runs in $2^{O(|x|)}$ time. Also, by Item (3) of Lemma 4.4.2, for every good n, $V(1^{\ell(n)}, y)$ accepts some y, and all the accepted y have no $S(\ell(n))$-size $\mathscr{C}$ circuits. This implies that unary NE does not admit $S(n)$-size witnesses. ■

Finally we are ready to prove Lemma 4.4.2.

Proof of Lemma 4.4.2. Item (1) follows from the fact that the running time is dominated by the complexity of applying Lemma 4.4.1 (which is $\mathrm{poly}(n, \log T)$) and running the assumed CAPP algorithm (which is $2^\ell / \ell^{\omega(1)} = T/(\log T)^{\omega(1)}$).

Let $x \in \{0,1\}^n$. To prove Item (2) and Item (3), note that from the completeness and soundness part of Lemma 4.4.1, we have:

(Completeness) If $L(x) = 1$, then there is an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ such that

$$\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^{\mathcal{O}}(r) = 1] = 1.$$

(Soundness) If $L(x) = 0$, then for all oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$, it holds that

$$\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_n^{\mathcal{O}}(r) = 1] \leq 1/n^{10}.$$

To see Item (2), from the soundness condition above, we know that when $L(x) = 0$, for all guessed circuits C, the estimate of $\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^C(r) = 1]$ is at most $1/n^{10} + 1/3 < 1/2$ (recall that the error parameter of CAPP is set to $1/3$ by default). Hence $\mathsf{APCP}(x) = 0$ as well.

Now we turn to establish Item (3). Item (3.a) follows immediately form the completeness condition above. To see Item (3.b), suppose for the sake of contradiction that there exists an $S(\ell)$-size $\mathscr{C}$ circuit C such that $\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^C(r) = 1] = 1$. Then such C would be guessed by APCP on the input x and its estimate of $\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^C(r) = 1]$ is at least $1 - 1/3 > 1/2$, meaning that $\mathsf{APCP}(x) = 1$. This contradicts the assumption of Item (3). ■

4.5 Almost-everywhere Lower Bounds via Refuters

In this section we prove Theorem 1.5.1 by utilizing the concept of *refuters*.

We begin by discussing why the proof of Theorem 4.4.3 and Corollary 4.1.7 fails to give almost-everywhere lower bounds. The issue is that the E^{NP} language from Corollary 4.1.7 is hard on input length ℓ only when the verifier $V(1^\ell, \cdot)$ "certifies" hardness, which only happens when ℓ corresponds to an input length n such that $L(1^n) = 1$ and $\mathsf{APCP}(1^n) = 0$ in the proof of Theorem 4.4.3 (*i.e.*, a good n in the proof of Theorem 4.4.3). From the NTIME hierarchy theorem, we can only infer that there are infinitely many good n (these n correspond to input lengths that APCP fails to compute L). If we wish to argue that almost all n are good, we would need an *almost-everywhere* NTIME hierarchy theorem, which is open.

Fortunately, we do have such an almost-everywhere NTIME hierarchy theorem by Fortnow and Santhanam [FS16], if we limit the amount of non-determinism used by the nondeterministic algorithms.

Reminder of Theorem 1.5.8. *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, $\mathsf{NTIME}[T(n)] \not\subset$ i.o.-$\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.*

It is tempting to apply Theorem 1.5.8 together with the algorithm APCP (described in Algorithm 4.1): Let $T(n) = n^C$ for a large constant C, $\ell(n) = \log T + O(\log\log T)$ be stated in Algorithm 4.1, and $L \in \mathsf{NTIME}[T(n)]$ be the hard language from Theorem 1.5.8. We can set $S(\ell) = 2^{\ell^\varepsilon}$ so that under the assumption of Algorithm 4.1, the resulting APCP is an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm. Hence, from Theorem 1.5.8, we know that APCP fails to compute L *on every input length*.

However, there is a missing piece: in order to utilizing Item (3) of Lemma 4.4.2, one needs to *find* an $x \in \{0,1\}^n$ satisfying $\mathsf{APCP}(x) \neq L(x)$ (by Item (2) of Lemma 4.4.2, it must be the case that $\mathsf{APCP}(x) = 0$ and $L(x) = 1$) so that we can use VPCP_x to verify a hard truth-table.[11]

To fill in the missing piece above, our most important new technical ingredient is the construction of a "refuter" for Theorem 1.5.9: an algorithm with an NP oracle that can efficiently find bad inputs for any $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ machine that attempts to compute the hard language $L \in \mathsf{NTIME}[T(n)]$, *for all sufficiently large input length n*.

Reminder of Theorem 1.5.9. *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq$*

[11] This is not an issue in the proof of Theorem 4.4.3, since the hard language there is unary, so one only has to consider 1^n. Also, one may attempt to provide such x as an *advice*, but this does not work as well since the advice size n would be much larger than the circuit lower bound size $S(\ell(n)) = 2^{O(\log n)^\varepsilon}$; using such a long advice would not give us any non-trivial lower bound.

$2^{\mathrm{poly}(n)}$, *there is a language* $L \in \mathsf{NTIME}[T(n)]$ *and an algorithm* $\mathcal{R}$ *such that:*

1. ***Input.*** *The input to* $\mathcal{R}$ *is a pair* $(M, 1^n)$*, with the promise that* M *describes a nondeterministic Turing machine running in* $o(T(n))$ *time and guessing at most* $n/10$ *bits.*
2. ***Output.*** *For every fixed* M *and every sufficiently large* n*,* $\mathcal{R}(M, 1^n)$ *outputs a string* $x \in \{0,1\}^n$ *such that* $M(x) \neq L(x)$*.*
3. ***Complexity.*** $\mathcal{R}$ *runs in* $\mathrm{poly}(T(n))$ *time with adaptive access to an* SAT *oracle.*

Since $\mathcal{R}$ *can find counterexamples to any faster algorithm attempting to decide* L*, we call* $\mathcal{R}$ *a* refuter.

Now, armed with Theorem 1.5.9, we are able to complete the proof outline above: we apply the refuter $\mathcal{R}$ to find an input x such that $L(x) = 1$ and $\mathsf{APCP}(x) = 0$, and now that VPCP_x certifies hardness from Item (3) of Lemma 4.4.2, we can find the lexicographically first string that makes VPCP_x always accepts. See below for a detailed proof.

Reminder of Theorem 1.5.1. *Let* $\mathscr{C}$ *be a typical concrete circuit class,* $K \in \mathbb{N}_{\geq 1}$ *be a sufficiently large constant, and* $S(n) \leq 2^{o(n)}$ *be a non-decreasing size parameter. If* CAPP *for* $n^K \cdot S(n)$*-size* $\mathsf{AC}^0_2 \circ \mathscr{C}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time, then there is an* $L \in \mathsf{E}^{\mathsf{NP}}$ *such that* L_n *does not have* $\mathscr{C}$ *circuits of size* $S(n/2)$ *for all sufficiently large* $n \in \mathbb{N}_{\geq 1}$*.*

Proof. Let $C \in \mathbb{N}$ be a sufficiently large constant. We also set $\varepsilon = \frac{1}{10C}$.

Let $T(n) = n^C$. Let L and $\mathcal{R}$ be the $\mathsf{NTIME}[T]$ language and the corresponding refuter from Theorem 1.5.9. Now we consider APCP (described in Algorithm 4.1) with T, L and size parameter $S(n)$, and set K to be the constant K in Algorithm 4.1. Note that the assumption of Algorithm 4.1 is satisfied with our choice of $S(n)$ and K and recall that $\ell(n) = \log T(n) + O(\log\log T(n)) = C\log n + O(\log\log n)$.

From Item (1) of Algorithm 4.1, APCP runs $\mathrm{poly}(n, \log T) + T/(\log T)^{\omega(1)} \leq o(T)$ time and guesses $\mathrm{poly}(S(\ell)) \leq o(n)$ bits (since C is sufficiently large and $S(\ell) \leq 2^{o(\ell)}$). Hence, the refuter $\mathcal{R}$ from Theorem 1.5.9 can be applied to find an input $x \in \{0,1\}^n$ such that $\mathsf{APCP}(x) \neq L(x)$, for every sufficiently large $n \in \mathbb{N}_{\geq 1}$.

Now we are ready to define a hard language $\mathcal{A}_{\mathsf{HARD}} \in \mathsf{E}^{\mathsf{NP}}$ as follows: on an input y of length m, we first construct (in E^{NP}) a function $f_m \colon \{0,1\}^m \to \{0,1\}$ (which only depends on m) such that f_m is hard against $\mathscr{C}$ circuits of size $S(m)$. Then we output $f_m(y)$. The function f_m is constructed as follows:

1. Let $n = 2^{m/2C}$, and $z = \mathcal{R}(\mathsf{APCP}, 1^n)$. By Item (2) of Lemma 4.4.2, it follows that $L(z) = 1$ and $\mathsf{APCP}(z) = 0$.

2. Now, we find the lexicographically first $\mathcal{O}\colon \{0,1\}^{\ell(n)} \to \{0,1\}$ such that $\Pr_{r\in_R\{0,1\}^\ell}[\mathsf{VPCP}_x^{\mathcal{O}}(r) = 1] = 1$ in $2^{O(\ell(n))} = \mathrm{poly}(T(n))$ time: fix the oracle $\mathcal{O}$ bit by bit, each time we can check if there is a way to complete remaining bits of the oracle $\mathcal{O}$ such that $\mathsf{VPCP}_z^{\mathcal{O}}$ always accepts using a SAT oracle.

   Note that $\mathcal{O}$ has $\ell(n) = C\log n + O(\log\log n) \in [m/2, m]$ inputs. By Item (3) of Lemma 4.4.2, it follows that $\mathcal{O}$ does not have $S(m/2)$-size $\mathscr{C}$ circuits (since $\ell(n) \geq m/2$ and S is non-decreasing).
3. Finally, the hard function $f_m\colon \{0,1\}^m \to \{0,1\}$ is defined as $f_m(y) = \mathcal{O}(y_{\leq\ell})$, where $y_{\leq\ell}$ is the prefix of y of length ℓ. Note that f_m does not have $\mathscr{C}$ circuits of size $s(m/2)$ as well.

To summarize, $\mathcal{A}_{\mathsf{HARD}}$ runs in $\mathsf{DTIME}[\mathrm{poly}(T(n))]^{\mathsf{NP}} = \mathsf{DTIME}[2^{O(m)}]^{\mathsf{NP}}$, and for all sufficiently large $m \in \mathbb{N}_{\geq 1}$, $\mathcal{A}_{\mathsf{HARD}}$ restricted to m-length inputs does not have $s(m/2)$-size $\mathscr{C}$ circuits, which completes the proof. ■

Chapter 5

Tighter Connections between Lower Bounds and Non-trivial Algorithms

This chapter is based on [CW19]. Below is the organization of this chapter.

Section 5.1 We provide intuitions behind the proof of Theorem 1.4.1, and explain how does PCP of proximity rise naturally while trying to improve algorithmic method.

Section 5.2 We discuss necessary preliminaries needed for this chapter.

Section 5.3 We establish tighter connection between circuit-analysis algorithms and lower bounds, and prove Theorem 1.4.1 and Theorem 1.4.2.

Section 5.4 We give two structure lemmas for $\mathsf{THR} \circ \mathsf{THR}$, which will be useful for the next section.

Section 5.5 To prove Theorem 1.4.5, in this section we first prove equivalences between non-trivial circuit analysis tasks of $\mathsf{THR} \circ \mathsf{THR}$ and that of $\mathsf{THR} \circ \mathsf{MAJ}$ or $\mathsf{MAJ} \circ \mathsf{MAJ}$. See Theorem 5.5.1 and Theorem 5.5.3.

Section 5.6 We propose approaches toward proving $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$. In particular, we prove Theorem 1.4.4 and Theorem 1.4.5 by combining Theorem 1.4.1 and the equivalence theorems from Section 5.5.

Section 5.7 Finally we give some proofs that are missing from previous sections.

Notation. Throughout this chapter, we say an algorithm for a circuit-analysis task (*e.g.*, SAT, Gap-UNSAT, or CAPP) is *non-trivial*, if its running time is at most $2^n/n^{\omega(1)}$ on circuits with n-bit inputs. Also recall that we use general circuits (or simply circuits) to refer to fan-in 2 De-Morgan circuits.

5.1 Intuition: Solving Gap-UNSAT with Probabilistic Checkable Proofs of Proximity

Here we provide an overview of the ideas behind our new tightened connection between circuit lower bounds and circuit-analysis algorithms (we will focus on Theorem 1.4.1). In the following we always use $\mathscr{C}$ to denote a typical and concrete circuit class (see Section 3.1.1).

Starting point: designing Gap-UNSAT algorithms for general circuits, assuming NEXP $\subset \mathscr{C}$. Suppose $\mathscr{C} \subset \mathsf{P/poly}$. We want to show that a non-trivial Gap-UNSAT algorithm with a constant gap for $\mathrm{poly}(n)$-size $\mathsf{AND}_3 \circ \mathscr{C}$ circuits implies $\mathsf{NEXP} \not\subset \mathscr{C}$. We start with the following connection of Williams [Wil10]:

> If Gap-UNSAT with gap $1 - 1/n^{10}$ for (fan-in 2 De Morgan) circuits with n inputs and $\mathrm{poly}(n)$ size is solvable in $2^n/n^{\omega(1)}$ nondeterministic time, then NEXP doesn't have $\mathrm{poly}(n)$-size circuits.

Our strategy is to assume $\mathsf{NEXP} \subset \mathscr{C}$, and use our non-trivial Gap-UNSAT algorithm for $\mathsf{AND}_3 \circ \mathscr{C}$ to derive a non-trivial Gap-UNSAT algorithm for general circuits. This would imply a contradiction, since by the above connection, it follows that $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathrm{poly}}$ and therefore $\mathsf{NEXP} \not\subset \mathscr{C}$.

So suppose we are given a $\mathrm{poly}(n)$-size general circuit $C : \{0,1\}^n \to \{0,1\}$ with the promise that either C is unsatisfiable (the YES case) or C has at least $(1 - 1/n^{10}) \cdot 2^n$ satisfying assignments (the NO case), where our goal is to distinguish the two cases in $2^n/n^{\omega(1)}$ non-deterministic time.

To simplify the discussion, we negate the circuit C. Now we are promised C is a tautology, or C has at most $1/n^{10} \cdot 2^n$ satisfying assignments, and we must nondeterministically prove C is a tautology (when that is the case) in $2^n/n^{\omega(1)}$ time.

Review of the approach in Williams' ACC^0 lower bound. It will be useful to review the previous approach ([Wil11]) first, and see where we deviate from it.[1] Let the circuit C be given as above. First, assuming $\mathsf{NEXP} \subset \mathscr{C}$ (which implies $\mathsf{Circuit\text{-}Eval} \in \mathscr{C}$), there is an equivalent $\mathrm{poly}(n)$-size $\mathscr{C}$ circuit D equivalent to C. Since we are allowed to use *non-deterministic* algorithms, we might try to *guess* a $\mathscr{C}$ circuit D, and *verify* that D is equivalent to C. If this verification can be done in $2^n/n^{\omega(1)}$ time, then we could apply the Gap-UNSAT algorithm for $\mathscr{C}$ to the circuit D, and solve Gap-UNSAT for C. Indeed, this is the original approach of Wiliams [Wil11].

[1] Our presentation here is slightly different from the original proof.

Since the NAND gate ($\mathsf{NAND}(z_1, z_2) := \neg(z_1 \wedge z_2)$) is universal, we may assume C consists of $m = \text{poly}(n)$ NAND gates, the first n gates are the inputs (that is, the i-th gate is the input bit x_i for $i \in [n]$), and the m-th gate is the output gate. Let C_i be the subcircuit of C where the i-th gate is the output. Since we are assuming $\mathsf{Circuit\text{-}Eval} \in \mathscr{C}$, for all C_i there is always an equivalent $\mathscr{C}$ circuit T_i of $\text{poly}(n)$ size.

The overall guess-and-verify algorithm works as follows:

- Guess $m - n$ $\mathscr{C}$ circuits $T_{n+1}, T_{n+2}, \ldots, T_m$, such that T_i is intended to be equivalent to C_i. For $i \in [n]$, we set T_i to be a trivial circuit which always outputs the i-th bit of the input.
- For $i \in \{n+1, n+2, \ldots, m\}$, let i_1 and i_2 be the indices of the two gates which are inputs to the i-th gate of C. We want to verify

$$\mathsf{NAND}(T_{i_1}(x), T_{i_2}(x)) = T_i(x) \tag{5.1}$$

  is true for all $x \in \{0,1\}^n$. This can be reduced to solving SAT for $\mathsf{AND}_3 \circ \mathscr{C}$ circuits.
- If all the above checks pass, then we know T_m is equivalent to C.

The proof system view. The above approach requires using SAT algorithms to verify (5.1) is true for all $x \in \{0,1\}^n$, whereas we only want to assume non-trivial Gap-UNSAT algorithms (which could be much weaker). Here we present a different perspective on the above approach.

Letting $\pi(x) := (T_{n+1}(x), T_{n+2}(x), \ldots, T_m(x))$, we can view $\pi(x)$ as a certain "locally-checkable proof" for $C(x) = 1$. That is, $C(x) = 1$ if and only if there is a proof $\pi(x) \in \{0,1\}^{m-n}$ such that for the string $z = x \circ \pi(x)$ ($\circ$ means concatenation), we have $\mathsf{NAND}(z_{i_1}, z_{i_2}) = z_i$ for all $i \in \{n+1, n+2, \ldots, m\}$, and $z_m = 1$.

Can we obtain something better from the "locally-checkable" perspective? We may write all the constraints checked in our proof system as a 3-CNF formula φ on $z = x \circ \pi(x)$ of $\ell = O(m) = \text{poly}(n)$ clauses. (Note, this simply mimics the standard reduction from Circuit-Eval to 3-SAT.) Suppose the i-th clause is $F_i(z) := \vee_{j=1}^{3}(z_{i_j} \oplus b_{i,j})$.

- As before, we guess $\mathscr{C}$ circuits $T_{n+1}(x), T_{n+2}(x), \ldots, T_m(x)$, but this time with the intention that $T(x) = (T_{n+1}(x), T_{n+2}(x), \ldots, T_m(x))$ is the correct proof for input x.
- When C is a tautology, there is a guess $T(x)$ such that $\mathbb{E}_{x \in_R \{0,1\}^n} \mathbb{E}_{i \in [\ell]}[F_i(x \circ T(x))] = 1$.
- Otherwise, for all guessed $T(x)$, we have $\mathbb{E}_{x \in_R \{0,1\}^n} \mathbb{E}_{i \in [\ell]}[F_i(x \circ T(x))] \leq 1/n^{10} + \frac{\ell-1}{\ell}$, since for at least a $1 - 1/n^{10}$ fraction of inputs, we have $C(x) = 0$, and therefore at most $\ell - 1$ clauses can be satisfied by $x \circ T(x)$.

Note that $F_i(x \circ T(x))$ is an $\mathsf{OR}_3 \circ \mathscr{C}$ circuit. We can try to estimate $\mathbb{E}_{x \in_R \{0,1\}^n}[F_i(x \circ T(x))]$ for each $i \in [\ell]$ to distinguish between the above two cases. Note there is only a $1/\ell = 1/\mathrm{poly}(n)$ gap between the above two cases. Therefore, this argument does show that, if we assume to have non-trivial CAPP algorithms for $\mathsf{OR}_3 \circ \mathscr{C}$ with $1/\mathrm{poly}(n)$ error, the above guess-and-verify approach already suffices to obtain lower bounds against $\mathscr{C}$.

However, in our case, we are only assuming to have a Gap-UNSAT algorithm with a *constant* gap. It is not clear how to make further progress with the above idea.

A better proof system? The above idea does not work, essentially because the described "proof system" is a pretty *bad* PCP! Given the pair $(x, T(x))$, if the verifier draws a random $i \in [\ell]$ and checks whether the clause F_i is satisfied, it is only promised to detect an error with probability $\geq 1/\ell$ when $C(x) = 0$ and the proof $T(x)$ is incorrect. In other words, it has a completeness/soundness gap of only $1/\ell = 1/\mathrm{poly}(n)$. A natural response to this observation is to try using a better proof system for proving that $C(x) = 1$; it comes as no surprise that we turn to the PCP Theorem [ALM+98, AS98].

However, there is a subtle issue. In the above proof system, the verifier does not need to know the input x beforehand, and only needs to query a bit of x when verifying a clause F_i containing that bit. The most important property here is that *the verifier's queries do not depend on the input x*, as otherwise we cannot formulate the condition "the verifier accepts with the random index i and proof $T(x)$ on input x" as a simple function $F_i(x \circ T(x))$ which can be represented by an $\mathsf{OR}_3 \circ \mathscr{C}$ circuit.

Suppose we forced the verifier to access the input x using only $O(1)$ queries, as in the above proof system, but the circuit is computing a highly-sensitive function such as the parity of x. There is no way that a verifier querying x for only $O(1)$ times can correctly infer (with high probability) that the parity of x is odd! This is because if the parity of x is odd, the parity will change if we flip a random bit of x, so it is not possible for a verifier to distinguish between these two cases with constant probability, if the verifier can only query x for $O(1)$ times.

Error correcting codes and probabilistic checkable proofs of proximity. To avoid the above trivial counterexample, our next key idea is to provide the PCP verifier an *error-correcting encoding* of the input. Now we are at the right position to introduce the main technical concept used in this chapter: *Probabilistic Checkable Proofs of Proximity* (PCPP) for the Circuit-Eval problem. When properly applied, PCPPs allow us to reduce the error requirement on the CAPP algorithms from

inverse polynomial to only a *constant*.

In this type of proof system[2], a circuit E is fixed in advance, the verifier $V(E)$ gets oracle access to the input x of length n and a proof string π, tosses some random coins, and makes at most 3 *non-adaptive* queries. The proof system has constant parameters $\delta > 0$ and $s \in (0,1)$, and satisfies two important properties:

- (Perfect Completeness.) $E(x) = 1 \Rightarrow$ there is a π such that $\Pr[V(E) \text{ accepts } x \circ \pi] = 1$.
- (Soundness on inputs far from being correct.) If x is δ-far from the set $\{y : E(y) = 1\}$, where δ is the proximity parameter, then for all possible proofs π, $V(E)$ accepts $x \circ \pi$ with probability at most $s < 1$.

To clarify the second point, we are saying that if x has hamming distance more than δn from all y that satisfy E, then $V(E)$ has decent probability of rejection on *any* proof π.

Suppose we use a linear error correcting code with an efficient encoder Enc and decoder Dec, and define the circuit E by $E(y) := C(\mathsf{Dec}(y))$. That is, E treats its input y as an encoding of an input to the circuit C; it first decodes y to a string z, then feeds z to C to get its output.

Let $x \in \{0,1\}^n$ be an input to C. We instantiate a PCP of proximity proof system with the circuit E and the input $\mathsf{Enc}(x)$. It is not hard to see that when $C(x) = 0$, $\mathsf{Enc}(x)$ is δ_1-far from the accepting inputs for E for a constant δ_1 depending on the error correcting code. We can ensure that $\delta_1 > \delta$.

The final reduction. Now, suppose there are ℓ possible outcomes of the random coins, and assume that the proof π is of length ℓ as well. Let $F_i(\mathsf{Enc}(x) \circ T(x))$ be the indicator that given a random outcome $i \in [\ell]$, whether the verifier $V(E)$ accepts the oracle $\mathsf{Enc}(x) \circ T(x)$. By definition, $F_i(\mathsf{Enc}(x) \circ T(x))$ is a function on 3 coordinates of $\mathsf{Enc}(x) \circ T(x)$ (we can assume WLOG that F_i is simply an OR, by using a special PCP of proximity proof system; see Lemma 5.2.6). Note that a bit of $\mathsf{Enc}(x)$ is just a parity over a subset of bits in x. For simplicity, let us further assume $\mathscr{C} = \mathsf{THR} \circ \mathsf{THR}$, which can compute parity (note, this assumption can be removed). Then $F_i(\mathsf{Enc}(x) \circ T(x))$ can now be formulated as an $\mathsf{OR}_3 \circ \mathscr{C}$ circuit. Now we proceed similarly as before.

- We again try to guess $\mathscr{C}$ circuits $T_1(x), T_1(x), \ldots, T_\ell(x)$, but this time with the hope that $T(x) = (T_1(x), T_2(x), \ldots, T_\ell(x))$ is the correct proof for the verifier $V(E)$ given input $\mathsf{Enc}(x)$.
- When C is a tautology, there is a guess $T(x)$ such that $\mathbb{E}_{x \in_R \{0,1\}^n} \mathbb{E}_{i \in [\ell]}[F_i(\mathsf{Enc}(x) \circ T(x))] = 1$.
- Otherwise, for all guesses $T(x)$, $\mathbb{E}_{x \in_R \{0,1\}^n} \mathbb{E}_{i \in [\ell]}[F_i(\mathsf{Enc}(x) \circ T(x))] \leq 1/n^{10} + s$, since for at

[2] see Definition 5.2.2

least a $1 - 1/n^{10}$ fraction of inputs, we have $C(x) = 0$, and therefore at most an s fraction of F_i's can be satisfied by $\mathsf{Enc}(x) \circ T(x)$, because $\mathsf{Enc}(x)$ is δ-far from any accepting input to E.

In this new situation, it now suffices to estimate $\mathbb{E}_{x \in_R \{0,1\}^n}[F_i(x \circ T(x))]$ for each $i \in [\ell]$ within sufficiently small constant error. A careful examination of the above argument shows it suffices to use a non-trivial Gap-UNSAT algorithm for $\mathsf{AND}_3 \circ \mathscr{C}$ circuits with a constant gap (note that the negation of F_i is an $\mathsf{AND}_3 \circ \mathscr{C}$ circuit), because we have perfect completeness in the case where C is a tautology.

Lower bounds from CAPP algorithms for $\mathsf{OR}_2 \circ \mathscr{C}$, $\mathsf{AND}_2 \circ \mathscr{C}$, or $\oplus_2 \circ \mathscr{C}$ Circuits. The above shows how to use a non-trivial CAPP algorithm for $\mathsf{OR}_3 \circ \mathscr{C}$; how can we use a non-trivial CAPP algorithm for $\mathsf{OR}_2 \circ \mathscr{C}$, $\mathsf{AND}_2 \circ \mathscr{C}$, or $\oplus_2 \circ \mathscr{C}$? The natural idea is to instead use a 2-query PCPP for Circuit-Eval. Unfortunately, there is no PCPP with only 2 queries with perfect completeness for Circuit-Eval, unless $\mathsf{P} = \mathsf{NP}$.[3] Thus we must use a construction with imperfect completeness. Luckily, there is a 2-query PCPP for Circuit-Eval with a constant soundness/completeness gap (Lemma 5.2.7). We use that PCPP in the above argument, together with other ideas, to establish the connection with a non-trivial CAPP algorithm for $\mathsf{OR}_2 \circ \mathscr{C}$, $\mathsf{AND}_2 \circ \mathscr{C}$ or $\oplus_2 \circ \mathscr{C}$ circuits.

5.2 Preliminaries

5.2.1 Previous Known Containment Results

We need the following known circuit classes containment results for this chapter.

Proposition 5.2.1. *The following hold:*

1. $\mathsf{THR} \subseteq \mathsf{MAJ} \circ \mathsf{MAJ}$ *[GHR92, Hof96].*
2. $\mathsf{THR} \subseteq \mathsf{DOR} \circ \mathsf{ETHR}$ *[HP10] (also see Section 5.7.2).*
3. $\mathsf{MAJ} \circ \mathsf{THR}$ *and* $\mathsf{MAJ} \circ \mathsf{ETHR}$ *are contained in* $\mathsf{MAJ} \circ \mathsf{MAJ}$ *[GHR92, HP10].*
4. $\mathsf{ETHR} \circ \mathsf{ETHR} \subseteq \mathsf{THR} \circ \mathsf{THR}$ *[HP10].*
5. $\mathsf{AND} \circ \mathsf{ETHR} \subseteq \mathsf{ETHR}$ *[HP10].*
6. $\mathsf{EMAJ} \subseteq \mathsf{MAJ} \circ \mathsf{AND}_2$ *[HP10].*
7. $\oplus_k \circ \mathsf{THR} \circ \mathsf{THR} \subseteq \mathsf{THR} \circ \mathsf{THR}$ *for a constant k (see Section 5.7.2).*
8. $\mathsf{THR} \circ \mathsf{EMAJ} \subseteq \mathsf{THR} \circ \mathsf{MAJ}$ *[HP10].*

Moreover, all the above have corresponding polynomial-time, deterministic constructions.

[3] A 2-query PCPP for Circuit-Eval with perfect completeness implies a 2-query PCP for NP with perfect completeness [BSGH+06], which in turn implies $\mathsf{P} = \mathsf{NP}$, as 2-SAT is in P.

For the containment $\mathsf{THR} \subseteq \mathsf{DOR} \circ \mathsf{ETHR}$, we present an alternative proof in Section 5.7.2, which is more efficient than the previously known construction of Hansen and Podolskii [HP10].[4] The last containment is folklore; we present a proof in Section 5.7.2 for completeness.

5.2.2 Probabilistic Checkable Proofs of Proximity

The concept of probabilistically checkable proofs of proximity is crucial for this chapter. In the following we introduce its definition and several instantiations useful for this chapter.

Definition 5.2.2 (Probabilistic checkable proofs of proximity (PCP of proximity or PCPP))**.** *For $s, \delta : \mathbb{N} \to [0,1]$ and $r, q : \mathbb{N} \to \mathbb{N}$, a verifier V is a PCP of proximity system for a pair language L with proximity parameter δ, soundness parameter s, number of random bits r and query complexity q if the following holds for all x, y:*

- *If $(x, y) \in L$, there is a proof π such that $V(x)$ accepts oracle $y \circ \pi$ with probability 1.*
- *If y is $\delta(|x|)$-far from $L(x) := \{z : (x, z) \in L\}$, then for all proofs π, $V(x)$ accepts oracle $y \circ \pi$ with probability at most $s(|x|)$.*
- *$V(x)$ tosses $r(|x|)$ random coins, and makes at most $q(|x|)$ non-adaptive queries.*

Remark 5.2.3. *We can also relax the first condition to be that there is a proof π such that $V(x)$ accepts oracle $y \circ \pi$ with probability at least $c = c(|x|)$, where c is the completeness parameter. In the above definition we assume $c = 1$, i.e., the perfect completeness.*

Lemma 5.2.4 (Theorem 3.3 in [BSGH+06])**.** *For any constants $0 < \delta, s < 1$, there is a PCP of proximity system for* Circuit-Eval *with proximity δ, soundness s, number of random bits $r = O(\log n)$ and query complexity $q = O(1)$. Moreover, given the pair $(C, w) \in$ Circuit-Eval, a proof π making $V(C)$ always accepts can be constructed in $\mathrm{poly}(|C| + |w|)$ time.*

Remark 5.2.5. *The moreover part is not explicitly stated in [BSGH+06], but it is evident from the constructions.*

The exact number of queries used in a PCPP will be significant for us, so we use query-efficient PCPPs. They are already implicit in the literature; for completeness, we provide expositions for them in Section 5.7.1.

[4] Hansen and Podolskii [HP10] proved that a THR gate on n bits with weights of absolute value no greater than W, can be written as a DOR of $O(n^2 \cdot \log W)$ many ETHR gates. In Section 5.7.2 we show it can be improved to $O(n \cdot \log W)$ many ETHR gates.

Lemma 5.2.6. *(3-query* PCPP *with perfect completeness) For any constant $\delta > 0$ there is a constant $0 < s < 1$, such that there is a PCP of proximity system for* Circuit-Eval *with proximity δ, soundness s, random bits $r = O(\log n)$, and query complexity $q = 3$. Moreover, the system satisfies two additional properties:*

(1) Given the random coins, the verifier simply computes an OR *on these 3 queried bits or their negations, and accepts iff the OR is true.*

(2) Given the pair $(C, w) \in$ Circuit-Eval, we can construct a proof π in $\mathrm{poly}(|C| + |w|)$ time that makes $V(C)$ accept with probability 1.

Lemma 5.2.7. *(2-query* PCPP *with constant completeness/soundness gap) For any constant $\delta > 0$ there two constants $0 < s < c < 1$, such that there is a PCP of proximity system for* Circuit-Eval *with proximity δ, soundness s, completeness c, number of random bits $r = O(\log n)$ and query complexity $q = 2$. Moreover, it satisfies two additional properties:*

(1) Given the random coins, the verifier computes an OR *on the 2 queried bits or their negations, and accepts iff the* OR *is true.*

(2) Given the pair $(C, w) \in$ Circuit-Eval, a proof π can be constructed in $\mathrm{poly}(|C| + |w|)$ time that makes $V(C)$ accept with probability at least c.

5.2.3 Error Correcting Codes

We also need standard constructions of constant-rate linear error correcting codes.

Lemma 5.2.8 ([Spi96])**.** *There is a constant $\delta > 0$ such that there is a constant-rate linear error correcting code* ECC *with minimum relative distance δ, an efficient encoder* Enc *and an efficient decoder* Dec *recovering error up to $c_1 \cdot \delta$, where c_1 is a universal constant.*

We use a slight modification of the above construction, which is convenient when we want to guess-and-verify a circuit for the encoder.

Lemma 5.2.9. *There is a constant $\delta > 0$ such that there is a constant-rate linear error correcting code* ECC *with minimum relative distance δ, an efficient encoder* Enc *and an efficient decoder* Dec *recovering error up to $c_1 \cdot \delta$, where c_1 is a universal constant. Moreover, each bit of the codeword depends on at most $n/2$ bits of the input.*

Proof. Given a message $x \in \{0,1\}^n$, we split it into three parts x_1, x_2, x_3, each of length between $\lfloor n/3 \rfloor$ and $\lceil n/3 \rceil$. Let Enc' and Dec' be the corresponding encoder and decoder of Lemma 5.2.8.

We construct our new error correcting code by setting $\mathsf{Enc}(x) := \mathsf{Enc}'(x_1) \circ \mathsf{Enc}'(x_2) \circ \mathsf{Enc}'(x_3)$. Given a codeword y, we split it into three strings y_1, y_2, y_3 of appropriate lengths, and let $\mathsf{Dec}(y) := \mathsf{Dec}'(y_1) \circ \mathsf{Dec}'(y_2) \circ \mathsf{Dec}'(y_3)$. ■

5.2.4 Connections Between Nondeterministic Gap-UNSAT Algorithms and Circuit Lower Bounds.

We also appeal to several known connections between Gap-UNSAT algorithms which improve upon exhaustive search and circuit lower bounds against nondeterministic time classes [Wil10, JMV15, SW13, BV14].

Theorem 5.2.10 ([Wil10]). *If* Gap-UNSAT *with gap* $1 - 1/n^{10}$ *for circuits with* n *inputs and* $\mathrm{poly}(n)$ *size is solvable in* $2^n/n^{\omega(1)}$ *nondeterministic time, then* NEXP *doesn't have* $\mathrm{poly}(n)$*-size circuits.*

Theorem 5.2.11 ([MW18]). *If there is an* $\varepsilon > 0$ *such that* Gap-UNSAT *with gap* $1 - 1/n^{10}$ *for circuits with* n *inputs and* 2^{n^ε} *size is solvable in* 2^{n-n^ε} *nondeterministic time, then for every* k *there is a function in* $\mathsf{NTIME}[n^{\mathrm{poly}(\log n)}]$ *that does not have* $n^{\log^k n}$*-size circuits.*

Theorem 5.2.12 ([MW18]). *If there is an* $\varepsilon > 0$ *such that* Gap-UNSAT *with gap* $1 - 1/n^{10}$ *for circuits with* n *inputs and* $2^{\varepsilon n}$ *size is solvable in* $2^{n-\varepsilon n}$ *nondeterministic time, then for every* k *there is a function in* NP *that does not have* n^k*-size circuits.*

5.3 Tighter Connections between Derandomization and Circuit Lower Bounds

In this section we show that $\mathscr{C}$ circuit lower bounds for NEXP or NP follow from better-than-2^n time derandomization of $\mathsf{AND}_3 \circ \mathscr{C}$, $\mathsf{OR}_2 \circ \mathscr{C}$, $\oplus_2 \circ \mathscr{C}$ or $\mathsf{AND}_2 \circ \mathscr{C}$ circuits.

Reminder of Theorem 1.4.1. *There is an absolute constant* $\delta \in (0,1)$*, such that for every typical* $\mathscr{C}$*, if either:*

- *there is a* $2^n/n^{\omega(1)}$*-time* $\mathsf{Gap\text{-}UNSAT}_\delta$ *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{AND}_3 \circ \mathscr{C}$ *circuits, or*
- *there is a* $2^n/n^{\omega(1)}$*-time* CAPP_δ *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{OR}_2 \circ \mathscr{C}$, $\oplus_2 \circ \mathscr{C}$*, or* $\mathsf{AND}_2 \circ \mathscr{C}$ *circuits,*

then $\mathsf{NEXP} \not\subset \mathscr{C}$*. Moreover, in the second bullet,* $\mathscr{C}$ *does not need to be closed under negation.*

Proof. We will show there is an absolute constant $\delta > 0$, such that if one of the algorithmic assumptions of the theorem holds and $\mathsf{NEXP} \subset \mathscr{C}$, then Gap-UNSAT with gap $1 - 1/n^{10}$ for $\mathrm{poly}(n)$-size

general circuits can be solved in $2^n/n^{\omega(1)}$ non-deterministic time. This proves the theorem, since by Theorem 5.2.10, we have $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathrm{poly}}$, which is a contradiction to $\mathsf{NEXP} \subset \mathscr{C}$.

We are given a $\mathrm{poly}(n)$-size general circuit $C : \{0,1\}^n \to \{0,1\}$ with the promise that either C is unsatisfiable, or C has at least $(1 - 1/n^{10}) \cdot 2^n$ satisfying assignments. Our goal is to distinguish between these two cases in $2^n/n^{\omega(1)}$ non-deterministic time.

Let $\delta_1 > 0$ be the constant of Lemma 5.2.9. We fix a constant-rate linear error correcting code with minimum relative distance δ_1, as guaranteed by Lemma 5.2.9. Let $\mathsf{Enc} : \{0,1\}^n \to \{0,1\}^{cn}$ and $\mathsf{Dec} : \{0,1\}^{cn} \to \{0,1\}^n$ be the corresponding encoder and decoder, where $c \geq 1$ is a constant corresponding to the rate of the code. Let $\delta_{\mathsf{Dec}} = c_1 \cdot \delta_1$, which is error rate that Dec can recover.

We also need a $\mathscr{C}$ circuit for the parity function on $n/2$ bits for computing Enc (by Lemma 5.2.9, the code is linear, and each output bit depends on at most $n/2$ input bits). By the assumption $\mathsf{NEXP} \subset \mathscr{C}$, the parity function must have a $\mathscr{C}$ circuit of $\mathrm{poly}(n)$ size. We can guess a $\mathscr{C}$ circuit $\mathsf{Par}_{n/2}$, and brute-force verify that it is correct in $2^{n/2} \cdot \mathrm{poly}(n)$ time.

Let $D : \{0,1\}^{cn} \to \{0,1\}$ be the circuit defined as $D(y) = \neg C(\mathsf{Dec}(y))$. Since C has $\mathrm{poly}(n)$ size and Dec is efficient, D also has $\mathrm{poly}(n)$ size. Then we can see

$$\Pr_{x \in_R \{0,1\}^n}[C(x) = 0] = \Pr_{x \in_R \{0,1\}^n}[D(\mathsf{Enc}(x)) = 1] = \Pr_{x \in_R \{0,1\}^n}[(D, \mathsf{Enc}(x)) \in \textsf{Circuit-Eval}].$$

With non-trivial Gap-UNSAT algorithms for poly-size $\mathsf{AND}_3 \circ \mathscr{C}$ circuits. We first prove the theorem under the first assumption. For that purpose we make use of a PCP of proximity system V for Circuit-Eval, with $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{Dec}}$, $r = O(\log n)$, $q = 3$ and a constant $s < 1$, whose existence is guaranteed by Lemma 5.2.6. We fix the circuit to be D, and write the verifier as $V(D)$.

We can view the verification of $V(D)$ as $m = 2^{r(|D|)} = \mathrm{poly}(n)$ many constraints on the oracle $y \circ \pi$. We can also assume $|\pi| = \ell = \mathrm{poly}(n)$. Suppose there are $F_1, F_2, \ldots, F_m$ constraints on $y \circ \pi$, each constraint is an OR on $q = 3$ variables or their negations.

Then the properties of PCP of proximity system translate to:

- If $y = \mathsf{Enc}(x)$ such that $C(x) = 0$, then $D(y) = 1$ and there is a proof $\pi \in \{0,1\}^\ell$ such that all constraints F_i's are satisfied by $y \circ \pi$.
- If $y = \mathsf{Enc}(x)$ such that $C(x) = 1$, then for all $z \in \{0,1\}^{cn}$ with $\mathsf{dist}(z, y) \leq \delta_{\mathsf{Dec}}$, we have $D(z) = C(\mathsf{Dec}(z)) = C(x) = 0$. Therefore, y is δ_{Dec}-far from $\textsf{Circuit-Eval}(D) = \{z : z \in \{0,1\}^{cn} \text{ and } (D, z) \in \textsf{Circuit-Eval}\}$. Since $\delta_{\mathsf{Dec}} > \delta_{\mathsf{PCPP}}$, we have that for all proofs $\pi \in \{0,1\}^\ell$, at most a s fraction of constraints F_i's are satisfied by $y \circ \pi$.

When C is unsatisfiable, then there is a proof $\pi(x)$ for each $y = \mathsf{Enc}(x)$, such that $V(D)$ accepts $y \circ \pi(x)$ with probability 1. Note that by Lemma 5.2.6, such a proof $\pi(x)$ can be computed in polynomial time from y and D, which in particular means that $\pi(x)$ admits a polynomial-size circuit, hence each bit of $\pi(x)$ admits a $n_{\mathsf{proof}} = \mathrm{poly}(n)$ size $\mathscr{C}$ circuit (here we use the assumption that $\mathsf{NEXP} \subset \mathscr{C}$).

Next, we guess a list of n_{proof}-size $\mathscr{C}$ circuits $T_1, T_2, \ldots, T_\ell$ such that

$$T(x) = (T_1(x), T_2(x), \ldots, T_\ell(x))$$

is intended to be the proof $\pi(x)$ for $y = \mathsf{Enc}(x)$. Slightly abusing notation, we also use F_i to denote the function $F_i(x) := F_i(\mathsf{Enc}(x) \circ T(x))$. Since a bit of $\mathsf{Enc}(x)$ is just a parity on at most $n/2$ bits in x, and since $\mathscr{C}$ is typical, each F_i can be written as an $\mathsf{OR}_3 \circ \mathscr{C}$ circuit. We also set $E_i(x) = \neg F_i(x)$, which is an $\mathsf{AND}_3 \circ \mathscr{C}$ circuit.

Therefore, when C is unsatisfiable, by the previous discussion, on some guesses of the T_i's, we have

$$\Pr_{x \in_{\mathsf{R}} \{0,1\}^n}[V(D)^{\mathsf{Enc}(x) \circ T(x)} = 1] = \mathop{\mathbb{E}}_{x \in_{\mathsf{R}} \{0,1\}^n} \mathop{\mathbb{E}}_{i \in_{\mathsf{R}} [m]}[F_i(x)] = 1.$$

Therefore, for all $i \in [m]$,

$$\mathop{\mathbb{E}}_{x \in_{\mathsf{R}} \{0,1\}^n}[E_i(x)] = 0.$$

When C has at most $2^n/n^{10}$ unsatisfying assignments, for all possible $T_1, T_2, \ldots, T_\ell$, we have

$$\mathop{\mathbb{E}}_{x \in_{\mathsf{R}} \{0,1\}^n} \mathop{\mathbb{E}}_{i \in_{\mathsf{R}} [m]}[F_i(x)] \leq 1/n^{10} + s.$$

By an averaging argument, there must be an i such that

$$\mathop{\mathbb{E}}_{x \in_{\mathsf{R}} \{0,1\}^n}[F_i(x)] \leq 1/n^{10} + s,$$

or equivalently

$$\mathop{\mathbb{E}}_{x \in_{\mathsf{R}} \{0,1\}^n}[E_i(x)] \geq 1 - s - 1/n^{10} \geq \frac{1-s}{2}.$$

Next, we set $\delta = \frac{1-s}{2}$. When C is unsatisfiable, all E_i's are unsatisfiable on the correct guesses. When C has at most $1/n^{10} \cdot 2^n$ unsatisfying assignments, then for all guesses, there is at least one i such that E_i has at least $\delta \cdot 2^n$ satisfying assignments. Hence, solving $\mathsf{Gap\text{-}UNSAT}$ with gap δ for all

E_i's suffices to non-deterministically distinguish between the two cases. By the first assumption, that takes $2^n/n^{\omega(1)}$ time, and the theorem follows from Theorem 5.2.10.

With non-trivial CAPP **algorithms for** $\mathrm{poly}(n)$**-size** $\mathsf{AND}_2 \circ \mathscr{C}$**,** $\oplus_2 \circ \mathscr{C}$ **or** $\mathsf{OR}_2 \circ \mathscr{C}$ **circuits.** The theorem under the second assumption can be proved similarly if we use the 2-query PCP of proximity system for Circuit-Eval instead, which is given by Lemma 5.2.7. The proof here is similar in parts to the one we just described for $\mathsf{AND}_3 \circ \mathscr{C}$; for completeness we will give the proof in full.

Now we make use of a PCP of proximity system V for Circuit-Eval, with $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{Dec}}$, $r = O(\log n)$, $q = 2$ and constants $0 < s < c < 1$, whose existence is guaranteed by Lemma 5.2.7. We again fix the circuit to be D, and write the verifier as $V(D)$.

Similarly, we can view the verification of $V(D)$ as $m = 2^{r(|D|)} \leq \mathrm{poly}(n)$ many constraints on the oracle $y \circ \pi$. We can also assume $|\pi| = \ell \leq \mathrm{poly}(n)$. Suppose there are $F_1, F_2, \ldots, F_m$ constraints on $y \circ \pi$, where each constraint is a function on $q = 2$ coordinates of $y \circ \pi$.

Then the properties of PCP of proximity system translate to:

- If $y = \mathsf{Enc}(x)$ such that $C(x) = 0$, then $D(y) = 1$ and there is a proof $\pi \in \{0,1\}^\ell$ such that at least a c-fraction of F_i's are satisfied by $y \circ \pi$.
- If $y = \mathsf{Enc}(x)$ such that $C(x) = 1$, then for all $z \in \{0,1\}^{cn}$ with $\mathsf{dist}(z, y) \leq \delta_{\mathsf{Dec}}$, we have $D(z) = C(\mathsf{Dec}(z)) = C(x) = 0$. Therefore, y is δ_{Dec}-far from Circuit-Eval(D). Since $\delta_{\mathsf{Dec}} > \delta_{\mathsf{PCPP}}$, we have that for all proofs $\pi \in \{0,1\}^\ell$, at most an s-fraction of F_i's are satisfied by $y \circ \pi$.

If C is unsatisfiable, then there is a proof $\pi(x)$ for each $y = \mathsf{Enc}(x)$ that makes $V(D)$ accept $y \circ \pi(x)$ with probability at least c. By Lemma 5.2.6, such a proof $\pi(x)$ can be computed in polynomial time from y and D, which in particular means that $\pi(x)$ has a polynomial-size circuit. Therefore each output bit of $\pi(x)$ has an $n_{\mathsf{proof}} = \mathrm{poly}(n)$ size $\mathscr{C}$ circuit, from the assumption that $\mathsf{NEXP} \subset \mathscr{C}$.

The next step is to guess a list of n_{proof}-size $\mathscr{C}$ circuits $T_1, T_2, \ldots, T_\ell$ such that $T(x) = (T_1(x), T_2(x), \ldots, T_\ell(x))$ is supposed to the proof $\pi(x)$ given input $y = \mathsf{Enc}(x)$. Slightly abusing notation, F_i is also used to denote the function $F_i(x) := F_i(\mathsf{Enc}(x) \circ T(x))$.

When C is unsatisfiable, by the previous discussion, there is a guess of T_i's such that

$$\Pr_{x \in_R \{0,1\}^n}[V(D)^{\mathsf{Enc}(x) \circ T(x)} = 1] = \mathop{\mathbb{E}}_{x \in_R \{0,1\}^n} \mathop{\mathbb{E}}_{i \in_R [m]}[F_i(x)] \geq c.$$

When C has at most $2^n/n^{10}$ unsatisfying assignments, then for all possible $T_1, T_2, \ldots, T_\ell$, we have

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n} \mathop{\mathbb{E}}_{i \in_R [m]} [F_i(x)] \leq 1/n^{10} + s.$$

Now set $\delta_1 := \frac{c-s}{2}$. In order for us to non-deterministically distinguish between the above two cases, it suffices to estimate

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n} [F_i(x)]$$

to within δ_1, for each $i \in [m]$.

Since each output bit of $\mathsf{Enc}(x)$ is simply a parity on at most $n/2$ bits of x, each F_i can be written as a function $F_i(x) = P(C_1(x), C_2(x))$, where C_1, C_2 are two $\mathscr{C}$ circuits, and P is a function from $\{0,1\}^2 \to \{0,1\}$. (Recall that in this case, we do not require $\mathscr{C}$ to be closed under negation.)

Now we write P as a polynomial:

$$P(z_1, z_2) = \sum_{S \subseteq [2]} \alpha_S \cdot \prod_{i \in S} z_i = \sum_{S \subseteq [2]} \alpha_S \cdot \bigwedge_{i \in S} z_i,$$

where each coefficient $\alpha_S \in [-4, 4]$. Given two $\mathscr{C}$ circuits C_1, C_2, to estimate

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n} [P(C_1(x), C_2(x))] = \sum_{S \subseteq [2]} \alpha_S \cdot \mathop{\mathbb{E}}_{x \in_R \{0,1\}^n} \left[\bigwedge_{i \in S} C_i(x) \right]$$

within error δ_1, it suffices to estimate each

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n} \left[\bigwedge_{i \in S} C_i(x) \right]$$

within error $\delta = \delta_1/16$. Finally, we can apply our assumed non-trivial CAPP algorithm for $\mathrm{poly}(n)$-size $\mathsf{AND}_2 \circ \mathscr{C}$ circuits to non-deterministically distinguish the two cases, and the theorem follows from Theorem 5.2.10.

When we only have non-trivial CAPP algorithms for $\oplus_2 \circ \mathscr{C}$ or $\mathsf{OR}_2 \circ \mathscr{C}$ circuits, we can simply write P in the basis of OR functions or $\oplus$ functions instead. That is, we can write

$$P(z_1, z_2) = \sum_{S \subseteq [2]} \alpha'_S \cdot \bigoplus_{i \in S} z_i,$$

or

$$P(z_1, z_2) = \sum_{S \subseteq [2]} \alpha''_S \cdot \bigvee_{i \in S} z_i.$$

The rest of the argument is the same as the case of $\mathsf{AND}_2 \circ \mathscr{C}$ circuits. ■

Using Theorem 5.2.12, the following theorem can be proved with the same argument as of Theorem 1.4.1.

Reminder of Theorem 1.4.2. *There is an absolute constant $\delta \in (0,1)$, such that for any typical circuit class $\mathscr{C}$, if there is a constant $\varepsilon \in (0,1)$ such that one of the following holds:*

- $\mathsf{Gap\text{-}UNSAT}_\delta$ *for $2^{\varepsilon n}$-size $\mathsf{AND}_3 \circ \mathscr{C}$ circuits can be solved in $2^{n-\varepsilon n}$ time, or*
- CAPP_δ *for $2^{\varepsilon n}$-size $\mathsf{OR}_2 \circ \mathscr{C}$, $\oplus_2 \circ \mathscr{C}$, or $\mathsf{AND}_2 \circ \mathscr{C}$ circuits can be solved in $2^{n-\varepsilon n}$ time,*

then for every k there is a function in NP that doesn't have n^k-size $\mathscr{C}$ circuits. Moreover, in the second bullet, $\mathscr{C}$ does not need to be closed under negation.

5.4 Structure Lemmas for $\mathsf{THR} \circ \mathsf{THR}$ Circuits

Two major technical ingredients in our results are structure lemmas for $\mathsf{THR} \circ \mathsf{THR}$, which are of interest in their own right. Informally, our first structure lemma says that every $\mathsf{THR} \circ \mathsf{THR}$ is equivalent to a polynomial-sized OR of Threshold-of-Majority circuits. The second structure lemma says that every $\mathsf{THR} \circ \mathsf{THR}$ circuit is equivalent to a subexponential-sized OR of Majority-of-Majority circuits. For the program of proving $\mathsf{THR} \circ \mathsf{THR}$ lower bounds, this is significant, as exponential-size Majority-of-Majority and Threshold-of-Majority lower bounds are well-known [HMP+93, FKL+01].

In the following, DOR refers to a "disjoint" OR gate: an OR gate with the promise that at most one of its inputs is ever true, and $\mathsf{Gap\text{-}OR}_\delta$ refers to a "gapped" OR gate with a error parameter δ: an OR gate with the promise that either all inputs are false or at least a $1-\delta$ fraction of the inputs are true. We also use Gap-OR to denote $\mathsf{Gap\text{-}OR}_{1/2}$ for simplicity.

Lemma 5.4.1 (Structure Lemma I for $\mathsf{THR} \circ \mathsf{THR}$ circuits)**.** *Let n be the number of inputs, let $s = s(n) \geq n$ be a size function, and let $\delta = \delta(n) \in (0,1)$ be an error function. Every s-size $\mathsf{THR} \circ \mathsf{THR}$ circuit C is equivalent to a $\mathsf{Gap\text{-}OR}_\delta \circ \mathsf{THR} \circ \mathsf{MAJ}$ circuit C' such that:*

- *The top $\mathsf{Gap\text{-}OR}_\delta$ gate of C' has $\mathrm{poly}(s, \delta^{-1})$ fan-in.*
- *Each $\mathsf{THR} \circ \mathsf{MAJ}$ subcircuit of C' has size $\mathrm{poly}(s, \delta^{-1})$.*

The transformation from C to C' can be computed in deterministic $\mathrm{poly}(s, \delta^{-1})$ *time.*

Lemma 5.4.2 (Structure Lemma II for $\mathsf{THR} \circ \mathsf{THR}$ circuits)**.** *Let n be the number of inputs and let $s = s(n) \leq 2^{o(n)}$ be a size parameter. Let $\varepsilon \in \left(\frac{\log s}{n}, 1\right)$. Every s-size $\mathsf{THR} \circ \mathsf{THR}$ circuit C is equivalent to a $\mathsf{DOR} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ circuit such that:*

- *The top DOR gate has $2^{O(\varepsilon n)}$ fan-in.*
- *Each sub $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuit has size $s^{O(1/\varepsilon)}$.*

The reduction can be computed in randomized $2^{O(\varepsilon n)} \cdot s^{O(1/\varepsilon)}$ *time.*

Previously, Goldmann-Håstad-Razborov [GHR92] showed that every $\mathsf{THR} \circ \mathsf{THR}$ circuit has an equivalent $\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ circuit of polynomially larger size. The top OR gates in our structure lemmas have additional benefits: for instance, an $\mathsf{OR} \circ \mathscr{C}$ circuit is satisfiable, *if and only if* one of its $\mathscr{C}$ subcircuits is satisfiable. Therefore, solving SAT on an $\mathsf{OR} \circ \mathscr{C}$ circuit is easily reduced to solving SAT on $\mathscr{C}$ circuits.

5.4.1 A Simple Construction

We first need a simple construction, which will be used in both proofs.

Definition 5.4.3 (Mod p Exact Threshold Gate)**.** *Let G be an ETHR gate with n inputs, p be a prime and G^p be the "mod p" version of G. That is, let L and T be the corresponding linear function and threshold of G, $G^p(x) := [L(x) \equiv T \pmod{p}]$.*

Lemma 5.4.4. *Let G be an ETHR gate with n inputs and p be a prime. Then G^p can be written as a $\mathsf{DOR} \circ \mathsf{ETHR}$ circuit such that*

- *The top DOR gate has $O(n)$ fan-in.*
- *All ETHR gates have positive weights and thresholds smaller than $O(np)$.*[5]

Proof. Let $w_1, w_2, \ldots, w_n$ and T be the corresponding weights and threshold of G. Reduce each weight w_i in G to $w_i \bmod p$ (the corresponding integer between 0 and $p-1$). This yields another circuit with associate top linear function $L'(x)$, whose value is always at most np. Setting $t = T \bmod p$, $L(x) \equiv T \pmod{p}$ is equivalent to $L'(x) = t + k \cdot p$ for some $k \in \{0, 1, 2, \ldots, n\}$. Therefore, by taking an OR over all possible k on the condition $L'(x) = t + k \cdot p$, it is a disjoint OR, and we obtain the equivalent $\mathsf{DOR} \circ \mathsf{ETHR}$ circuit. ■

[5] Therefore, when $p \leq \mathrm{poly}(n)$, the ETHR gate can be seen as an EMAJ gate.

5.4.2 Proof of Structure Lemma I

Now we are ready to prove Lemma 5.4.1.

Proof of Lemma 5.4.1. Let C' be the given $\mathsf{THR} \circ \mathsf{THR}$ circuit. By negating some of its input gates (THR is closed under negation), we may assume all weights in the top THR gate of C' are ≤ 0. Since every THR can be converted into a $\mathsf{DOR} \circ \mathsf{ETHR}$ (item (2) of Proposition 5.2.1), C' can be transformed into an equivalent $\mathsf{THR} \circ \mathsf{ETHR}$ circuit C of size $t = \mathrm{poly}(s)$.

Let $G_1, G_2, \ldots, G_t, w_1, w_2, \ldots, w_t$ be the ETHR gates on the bottom layer and their corresponding weights in the top gate of C. By assumption, we also have $w_i \leq 0$ for all i. Let T be the threshold of the top gate. For all inputs x of n bits, we have

$$C(x) = \left[\sum_{i=1}^{t} w_i \cdot G_i(x) \geq T\right].$$

By construction, we may assume that the weights in G_i are bounded by 2^{n^c} for a constant c. Suppose we fix an input x, and let p be a random prime from 2 to $n^{2c} \cdot t^2 \cdot \delta^{-1} = \mathrm{poly}(s, \delta^{-1})$. With probability at least $1 - \delta/t$, we have $G_i^p(x) = G_i(x)$. Let C^p be the circuit obtained by replacing all G_i's in C by corresponding G_i^p's.

When $C(x) = 1$, it follows from a union bound that $C^p(x) = C(x) = 1$ with probability at least $1 - \delta$. When $C(x) = 0$, note that for all primes p, we have $G_i^p(x) \geq G_i(x)$ for all i, therefore we must have $\sum_i^s w_i \cdot G_i^p(x) \leq \sum_i^s w_i \cdot G_i(x) < T$ (all w_i's are ≤ 0) and $C^p(x) = 0$.

Therefore C is equivalent to a $\mathsf{Gap\text{-}OR}_\delta$ over *all* C^p's, for every prime p (recall their total number is $\mathrm{poly}(s, \delta^{-1})$). By Lemma 5.4.4, each C^p can be expressed as a $\mathrm{poly}(s, \delta^{-1})$-size $\mathsf{THR} \circ \mathsf{EMAJ}$ circuit. Converting each $\mathsf{THR} \circ \mathsf{EMAJ}$ into a $\mathsf{THR} \circ \mathsf{MAJ}$ (item (8) of Proposition 5.2.1) completes the proof. ■

5.4.3 Proof of Structure Lemma II

Now we turn to proving Lemma 5.4.2. The proof has two steps, provided by Lemma 5.4.5 and Lemma 5.4.7.

Lemma 5.4.5 (Weight Reduction at the Top THR gate)**.** *Every size-s $\mathsf{THR}_d \circ \mathscr{C}$ circuit (having a top THR gate of fan-in d) is equivalent to a $\mathsf{DOR} \circ \mathsf{ETHR} \circ \mathscr{C}$ circuit such that:*

- *The top DOR gate has $\mathrm{poly}(d)$ fan-in.*

- *Each* ETHR *gate has fan-in d, with positive weights and threshold value, all of which are less than* $\mathrm{poly}(d) \cdot 2^n$.
- *The $\mathscr{C}$-part is unchanged.*

The same statement also holds for $\mathsf{ETHR}_d \circ \mathscr{C}$ circuits. Moreover, the reductions can be computed in randomized $\mathrm{poly}(s)$ *time.*

Proof. We only consider the $\mathsf{THR}_d \circ \mathscr{C}$ case (the $\mathsf{ETHR}_d \circ \mathscr{C}$ case is even easier).

Let C be the given circuit. First, by the fact that $\mathsf{THR} \subseteq \mathsf{DOR} \circ \mathsf{ETHR}$ (item (2) of Proposition 5.2.1), C can be transformed to an equivalent $\mathsf{DOR} \circ \mathsf{ETHR} \circ \mathscr{C}$ circuit C'.

Let G be a ETHR gate in C'; note that G has fan-in d. Let D be the subcircuit with top gate G. By construction, G has weights of absolute value at most $M_{\mathsf{old}} = 2^{\mathrm{poly}(d)}$.

Next, we define $L : \{0,1\}^n \to \mathbb{Z}$ such that $L(x)$ is the value of the linear function associated with the gate G when the input is x. That is $D(x) = 1$ if and only if $L(x) = T$ for the threshold T of G.

Pick a random prime number m in the interval $[2, M_{\mathsf{new}}]$, where $M_{\mathsf{new}} = d^c \cdot 2^n$ and c is a sufficiently large constant. For a fixed $x \in \{0,1\}^n$, if $L(x) \neq T$, the probability that $L(x) \equiv T \pmod{m}$ is smaller than

$$\frac{\log(M_{\mathsf{old}})}{M_{\mathsf{new}}/\ln(M_{\mathsf{new}})} = \frac{\mathrm{poly}(d)}{\Theta(2^n \cdot d^c/(n + c\log d))} \leq d^{-c/2}/2^n,$$

for a sufficiently large c. Applying the union bound over all inputs x, with probability at least $1 - d^{-c/2}$, we have $L(x) \equiv T \pmod{m}$ if and only if $L(x) = T$ for all $x \in \{0,1\}^n$.

Finally, applying Lemma 5.4.4 with prime m, we can replace G with an equivalent $\mathsf{DOR} \circ \mathsf{ETHR}$ subcircuit, whose ETHR gates have positive weights and thresholds smaller than $\mathrm{poly}(d) \cdot 2^n$.

Union-bounding over all ETHR gates, and choosing c to be a large enough constant, this completes the randomized reduction. ∎

Remark 5.4.6. *One can observe that the above reduction indeed only introduces* one-sided error. *That is, even if it chooses some "bad" primes, the resulting circuit D satisfies the property that $D(x) = 1$ whenever $C(x) = 1$.*

Lemma 5.4.7 (Decomposition of the top ETHR gate)**.** *Given an $\mathsf{ETHR}_d \circ \mathscr{C}$ circuit C (a circuit with a top* ETHR *gate of fan-in d) of size s and a real $\varepsilon \in \left(\frac{\log d}{n}, 1\right)$, suppose the top* ETHR *gate in C has positive weights and threshold smaller than 2^{2n}. C is equivalent to a $\mathsf{DOR} \circ \mathsf{MAJ} \circ \mathsf{AND}_2 \circ \mathscr{C}$ circuit such that:*

- *The top* DOR *gate has* $2^{O(\varepsilon n)}$ *fan-in.*
- *Each* MAJ *gate has fan-in* $d^{O(1/\varepsilon)}$.
- *The* $\mathscr{C}$ *part is unchanged.*

Moreover, the reduction can be computed in deterministic

$$2^{O(\varepsilon n)} \cdot d^{O(1/\varepsilon)} + \mathrm{poly}(s)$$

time.

Proof. Let G_{top} be the top ETHR in C, and let $G_1, G_2, \ldots, G_d$ be its input gates. Let w_i's and T be the weights and the threshold of G_{top} and $L(x)$ be the associated linear function. We have for all $x \in \{0,1\}^n$ that

$$L(x) = \sum_{i=1}^{d} w_i \cdot G_i(x).$$

Observe that the binary representations of w_i's and T are of length at most $\log(2^{2n}) \leq 2n$. Break each of their binary representations into $D = \left\lceil \frac{\varepsilon \cdot n}{\log d} \right\rceil$ blocks, where each block has $B \leq 2/\varepsilon \cdot \log d$ bits. Let $w_{i,j}, T_j \in [2^B - 1]$ be the values of w_i's and T's j-th block, respectively (where blocks are numbered from the least significant bit to the most significant bit).

Consider adding the $w_i \cdot G_i(x)$'s in base 2^B, keeping track of all $D-1$ carries on each position, except for the highest one. Let $c = (c_1, c_2, \ldots, c_{D-1}) \in \{0, 1, \ldots, d-1\}^{D-1}$ be such a carry sequence. Observe that $\sum_{i=1}^{d} w_i \cdot G_i(x) = T$ with carry sequence c if and only if for all $j \in [D]$:

$$\sum_{i=1}^{d} w_{i,j} \cdot G_i(x) + c_{j-1} = T_j + 2^B \cdot c_j,$$

where we set C_D and C_0 to be 0 for notational convenience. That is, after we fix the carries c_j's for all j, the sums $\sum_{i=1}^{d} w_{i,j} \cdot G_i(x)$ are also forced to be $T_j^c = T_j + 2^B \cdot c_j - c_{j-1}$. Therefore, consider the sum

$$\sum_{j=1}^{\varepsilon \cdot n} \left(\sum_{i=1}^{d} w_{i,j} \cdot G_i(x) - T_j^c \right)^2.$$

Checking whether this sum is at most 0 can be formulated as a $\mathrm{poly}(d) \cdot 2^{O(B)} = d^{O(1/\varepsilon)}$ size $\mathsf{MAJ} \circ \mathsf{AND}_2$ subcircuit, with input gates $G_1, G_2, \ldots, G_d$.

Each of these addition checks corresponds to one carry sequence. By enumerating all possible d^{D-1} carry sequences, the above transforms G_{top} into a $\mathsf{DOR} \circ \mathsf{MAJ} \circ \mathsf{AND}_2$ subcircuit with input

gates $G_1, G_2, \ldots, G_d$, having top fan-in:

$$d^{D-1} = d^{O(\varepsilon \cdot n/\log d)} = 2^{O(\varepsilon \cdot n)},$$

which completes the proof. ■

Finally, the Structure Lemma II for $\mathsf{THR} \circ \mathsf{THR}$ circuits follows from applying Lemma 5.4.5 and Lemma 5.4.7 in the appropriate way.

Proof of Lemma 5.4.2. First, since $\mathsf{THR} \subseteq \mathsf{DOR} \circ \mathsf{ETHR}$ (item (2) of Proposition 5.2.1), C is equivalent to a $\mathrm{poly}(s)$-size $\mathsf{THR} \circ \mathsf{ETHR}$ circuit C_1. Moreover, we can convert C into C_1 in polynomial time.

Second, we apply Lemma 5.4.5 to transform C_1 into a $\mathsf{DOR}_{\mathrm{poly}(s)} \circ \mathsf{ETHR} \circ \mathsf{ETHR}$ circuit C_2, such that all middle-layer ETHR gates have positive weights and thresholds smaller than $\mathrm{poly}(s) \cdot 2^n < 2^{2n}$.

Third, we apply Lemma 5.4.7 to C_2, which changes all middle-layer ETHR gates of C_2 into $\mathsf{DOR} \circ \mathsf{MAJ} \circ \mathsf{AND}_2$ subcircuits, with top gate fan-in $2^{O(\varepsilon \cdot n)}$. This yields a $\mathsf{DOR} \circ \mathsf{MAJ} \circ \mathsf{AND} \circ \mathsf{ETHR}$ circuit. Converting the remaining $\mathsf{AND} \circ \mathsf{ETHR}$ subcircuits into ETHR's (item (5) of Proposition 5.2.1), we obtain a $\mathsf{DOR}_{2^{O(\varepsilon \cdot n)}} \circ \mathsf{MAJ} \circ \mathsf{ETHR}$ circuit where all $\mathsf{MAJ} \circ \mathsf{ETHR}$ subcircuits have size at most $s^{O(1/\varepsilon)}$.

Finally, converting each $\mathsf{MAJ} \circ \mathsf{ETHR}$ into a $\mathsf{MAJ} \circ \mathsf{MAJ}$ (item (3) of Proposition 5.2.1) completes the reduction. The running time bound follows from plugging in the time bounds of Lemma 5.4.5 and Lemma 5.4.7. ■

Setting the parameter ε carefully in Lemma 5.4.2, we have the following corollary.

Corollary 5.4.8. *Let n be the number of inputs and let $s = s(n) \leq 2^{o(n)}$ be a size parameter. Let $\varepsilon \in \left(\frac{\log s}{n}, 1\right)$. Every s-size $\mathsf{THR} \circ \mathsf{THR}$ circuit C is equivalent to a $\mathsf{DOR} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ circuit C' such that:*

- *The top DOR gate of C' has $s^{O(1/\varepsilon)}$ fan-in.*
- *Every sub $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuit of C' has size $2^{O(\varepsilon \cdot n)}$.*

The reduction can be computed in randomized $2^{O(\varepsilon n)} \cdot s^{O(1/\varepsilon)}$ time.

5.5 Equivalence between Algorithmic Analysis of $\mathsf{THR} \circ \mathsf{THR}$ and of $\mathsf{THR} \circ \mathsf{MAJ}$ or $\mathsf{MAJ} \circ \mathsf{MAJ}$

In this section, building on our new structure lemmas for $\mathsf{THR} \circ \mathsf{THR}$ circuits. We show several equivalence results between canonical circuit-analysis tasks (SAT or CAPP) of $\mathsf{THR} \circ \mathsf{THR}$ circuits and that of $\mathsf{THR} \circ \mathsf{MAJ}$ or $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits.

5.5.1 Poly-Size $\mathsf{THR} \circ \mathsf{MAJ}$ and $\mathsf{THR} \circ \mathsf{THR}$ are Equivalent for Circuit-Analysis Algorithms

We first show that, in terms of designing non-trivial circuit-analysis algorithms, $\mathsf{THR} \circ \mathsf{THR}$ and $\mathsf{THR} \circ \mathsf{MAJ}$ circuits are essentially *equivalent*.

Theorem 5.5.1. *The following two statements hold:*

- ***Equivalence of non-trivial*** SAT ***algorithms****: There is a non-trivial SAT algorithm for* $\mathsf{THR} \circ \mathsf{MAJ}$ *circuits of* $\mathrm{poly}(n)$*-size* if and only if *there is such an algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{THR} \circ \mathsf{THR}$ *circuits.*
- ***Equivalence of non-trivial*** CAPP ***algorithms with constant error****: For any constant* $\delta > 0$*, If there is a non-trivial* CAPP *algorithm with error* δ *for* $\mathsf{THR} \circ \mathsf{MAJ}$ *circuits of* $\mathrm{poly}(n)$ *size, then there is a non-trivial* CAPP *algorithm with error* $\delta + 1/n$ *for* $\mathrm{poly}(n)$*-size* $\mathsf{THR} \circ \mathsf{THR}$ *circuits.*

Proof. We begin with the first equivalence. We only have to show that a $2^n/n^{\omega(1)}$ time SAT algorithm for $\mathrm{poly}(n)$-size $\mathsf{THR} \circ \mathsf{MAJ}$ circuits implies such an algorithm for $\mathsf{THR} \circ \mathsf{THR}$ circuits. By Lemma 5.4.1, given any $\mathsf{THR} \circ \mathsf{THR}$ circuit of $\mathrm{poly}(n)$ size, in $\mathrm{poly}(n)$ time we can construct an equivalent $\mathrm{poly}(n)$-size $\mathsf{Gap\text{-}OR} \circ \mathsf{THR} \circ \mathsf{MAJ}$ circuit C. Applying the assumed SAT algorithm for $\mathsf{THR} \circ \mathsf{MAJ}$ circuits on all $\mathsf{THR} \circ \mathsf{MAJ}$ subcircuits of C completes the proof of the first equivalence.

For the second equivalence, given any $\mathsf{THR} \circ \mathsf{THR}$ circuit C of $\mathrm{poly}(n)$ size, we construct in $\mathrm{poly}(n)$ time a $\mathsf{Gap\text{-}OR}_{1/n} \circ \mathsf{THR} \circ \mathsf{MAJ}$ circuit D that is equivalent to C, by Lemma 5.4.1. Let $D_1, D_2, \ldots, D_m$ be the $\mathsf{THR} \circ \mathsf{MAJ}$ subcircuits of C, where $m = \mathrm{poly}(n)$.

By the definition of a $\mathsf{Gap\text{-}OR}_{1/n}$ gate, for all $x \in \{0,1\}^n$, we have

$$\left| C(x) - \mathop{\mathbb{E}}_{i \in_R [m]} D_i(x) \right| \leq 1/n.$$

Therefore, to estimate $\mathbb{E}_{x \in_R \{0,1\}^n}[C(x)]$ within error $\delta + 1/n$, it suffices to estimate $\mathbb{E}_{x \in_R \{0,1\}^n}[D_i(x)]$ for each $i \in [m]$ within error δ. Applying the non-trivial CAPP algorithm for $\mathsf{THR} \circ \mathsf{MAJ}$ circuits

from the assumption completes the proof. ■

With an argument similar to the proof of Theorem 5.5.1 and using the fact that $\mathsf{MAJ} \circ \mathsf{THR} \subseteq \mathsf{MAJ} \circ \mathsf{MAJ}$ (potentially multiple times), it is not hard to generalize Theorem 5.5.1 to hold for TC circuits of any constant depth d.

Corollary 5.5.2. *The following two statements hold for any constant d:*

- ***Equivalence of non-trivial*** SAT ***algorithms****: There is a non-trivial SAT algorithm for* $\mathsf{THR} \circ \widehat{\mathsf{LT}}_{d-1}$ *circuits of* $\mathrm{poly}(n)$*-size* if and only if *there is such an algorithm for* $\mathrm{poly}(n)$*-size* LT_d *circuits.*
- ***Equivalence of non-trivial*** CAPP ***algorithms with constant error****: For any constant $\varepsilon > 0$, if there is a non-trivial* CAPP *algorithm with error ε for* $\mathsf{THR} \circ \widehat{\mathsf{LT}}_{d-1}$ *circuits of* $\mathrm{poly}(n)$*-size, then there is a non-trivial* CAPP *algorithm with error $\varepsilon + 1/n$ for* $\mathrm{poly}(n)$*-size* LT_d *circuits.*

5.5.2 Weaker Equivalence Between Poly-Size $\mathsf{THR} \circ \mathsf{THR}$ and $\mathsf{MAJ} \circ \mathsf{MAJ}$

We also show a weaker equivalence for $\mathsf{THR} \circ \mathsf{THR}$ and $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits.

Theorem 5.5.3. *The following two statements hold:*

- ***Equivalence of*** $2^{(1-\varepsilon)n}$***-time*** SAT ***algorithms****: If* SAT *for* $\mathrm{poly}(n)$*-size* $\mathsf{MAJ} \circ \mathsf{MAJ}$ *circuits is in* $2^{(1-\varepsilon)n}$ *time for some constant $\varepsilon > 0$, then* SAT *for* $\mathrm{poly}(n)$*-size* $\mathsf{THR} \circ \mathsf{THR}$ *circuits is in* $2^{(1-\varepsilon')n}$ *time for some $\varepsilon' > 0$.*
- ***Equivalence of non-trivial*** CAPP ***algorithms with inverse-circuit-size error****: If there is a non-trivial* $\widetilde{\mathsf{CAPP}}$ *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{MAJ} \circ \mathsf{MAJ}$ *circuits, then there is a non-trivial* $\widetilde{\mathsf{CAPP}}$ *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{THR} \circ \mathsf{THR}$ *circuits.*

Proof. We begin with the first equivalence.

The first equivalence. Suppose we have a $2^{(1-\varepsilon_1)n}$ time SAT algorithm for $\mathrm{poly}(n)$ size $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits for a constant $\varepsilon_1 > 0$, and want to design a $2^{n-\Omega(n)}$ time SAT algorithm for $\mathrm{poly}(n)$ size $\mathsf{THR} \circ \mathsf{THR}$ circuits.

Let c be the hidden constant in the big-O of the fan-in of the top DOR gate from Lemma 5.4.2. Set $\varepsilon := \varepsilon_1/2c$, and apply Lemma 5.4.2 to the given $\mathrm{poly}(n)$-size $\mathsf{THR} \circ \mathsf{THR}$ circuit. We obtain an equivalent $\mathsf{DOR} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ circuit with top fan-in $2^{c\varepsilon n} = 2^{\varepsilon_1/2 \cdot n}$ and $\mathrm{poly}(n)$-size $\mathsf{MAJ} \circ \mathsf{MAJ}$ subcircuits. Then we can apply our SAT algorithm for $\mathrm{poly}(n)$-size $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits to solve the SAT problem for $\mathrm{poly}(n)$ size $\mathsf{THR} \circ \mathsf{THR}$ circuits, which completes the proof of the first equivalence.

The second equivalence. To show the second equivalence, suppose for all constants k', there is a non-trivial CAPP algorithm for $n^{k'}$-size MAJ $\circ$ MAJ circuits with error $1/n^{k'}$. We have to design such an algorithm for poly(n)-size THR $\circ$ THR circuits.

Given a THR $\circ$ THR circuit C of $s = \text{poly}(n) \leq n^k$ for a constant k, we want to estimate

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n}[C(x)] \tag{5.2}$$

within error $1/n^k$.

Since THR $\subseteq$ DOR $\circ$ ETHR (item (2) of Proposition 5.2.1), we can write C as a DOR of $m = \text{poly}(s) = \text{poly}(n)$ ETHR $\circ$ THR subcircuits $C_1, C_2, \ldots, C_m$. By the definition of DOR, we have

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n}[C(x)] = \mathop{\mathbb{E}}_{x \in_R \{0,1\}^n}\left[\sum_{i=1}^{m} C_i(x)\right] = \sum_{i=1}^{m} \mathop{\mathbb{E}}_{x \in_R \{0,1\}^n}[C_i(x)].$$

Therefore, in order to estimate (5.2) within error $1/n^k$, it suffices to estimate

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n}[C_i(x)]$$

within error $1/(m \cdot n^k)$ for each $i \in [m]$.

So fix an $i \in [m]$. Let $D = C_i$, and let D's top ETHR gate be G. By construction, G has weights of absolute value at most 2^{n^c}, for a constant c depending on k. Define $L : \{0,1\}^n \to \mathbb{Z}$ so that $L(x)$ is the value of the linear function associated with G on input x. That is, $D(x) = 1$ if and only if $L(x) = T$ for the threshold T of G.

Suppose we pick a random prime number p in the interval $[2, M]$, where $M = n^{2c} \cdot (2m \cdot n^k)^2 \leq \text{poly}(n)$. Then for a fixed $x \in \{0,1\}^n$, if $L(x) \neq T$, the probability that $L(x) \equiv T \pmod{p}$ is less than $1/(2m \cdot n^k)$.

Recall that for a prime p and an ETHR gate $G(x) = [\sum_{i=1}^{n} w_i \cdot x_i = T]$, we use G^p to denote its "mod p" version (see Definition 5.4.3). Let D^p denote the circuit obtained by replacing the top G gate in D by G^p. For all $x \in \{0,1\}^n$, by the above discussion, we have

$$\left| D(x) - \mathop{\mathbb{E}}_{\text{prime } p \in [2, M]}[D^p(x)] \right| \leq 1/(2m \cdot n^k).$$

Therefore, in order to estimate $\mathbb{E}_{x\in_R\{0,1\}^n}[D(x)]$ within error $1/(m\cdot n^k)$, it suffices to estimate

$$\mathop{\mathbb{E}}_{x\in_R\{0,1\}^n}[D^p(x)]$$

for all primes $p \le M$, within error $1/(2m\cdot n^k)$.

By Lemma 5.4.4, each D^p can be written as a DOR of $O(n)$ EMAJ $\circ$ ETHR circuits of poly(n) size. Since EMAJ $\subseteq$ MAJ $\circ$ AND$_2$, AND $\circ$ ETHR $\subseteq$ ETHR and MAJ $\circ$ ETHR $\subseteq$ MAJ $\circ$ MAJ (items (6), (5), and (3) of Proposition 5.2.1), D^p can be further written as a DOR of cn MAJ $\circ$ MAJ circuits $D^p_1, D^p_2, \ldots, D^p_{cn}$ of poly(n) size, for a universal constant c.

Therefore, to estimate $\mathbb{E}_{x\in_R\{0,1\}^n}[D^p(x)]$ within error $1/(2m\cdot n^k)$, it suffices to estimate $\mathbb{E}_{x\in_R\{0,1\}^n}[D^p_i(x)]$ within error $1/(2m\cdot n^k\cdot cn)$, for each $i\in[cn]$.

Observe that $2m\cdot n^k\cdot cn \le$ poly(n), and all D^p_i's are poly(n)-size MAJ $\circ$ MAJ circuits. Applying the assumed CAPP algorithm for MAJ $\circ$ MAJ with a sufficiently large k' completes the proof of the second equivalence. ■

Again applying the fact that MAJ $\circ$ THR $\subseteq$ MAJ $\circ$ MAJ, the generalization to TC circuits of any constant depth d is immediate.

Corollary 5.5.4. *The following two statements hold for any constant d:*

- ***Equivalence of** $2^{(1-\varepsilon)n}$**-time** SAT **Algorithms**: If SAT for $\widehat{\mathsf{LT}}_d$ circuits of* poly(n)*-size is in $2^{(1-\varepsilon)n}$ time for a constant $\varepsilon>0$, then SAT for* poly(n)*-size LT_d circuits is in $2^{(1-\varepsilon')n}$ time for some $\varepsilon'>0$.*
- ***Equivalence of Non-trivial** CAPP **Algorithms with inverse-circuit-error**: If there is a non-trivial $\widetilde{\mathsf{CAPP}}$ algorithm for* poly(n)*-size $\widehat{\mathsf{LT}}_d$ circuits, then there is a non-trivial $\widetilde{\mathsf{CAPP}}$ algorithm with for* poly(n)*-size LT_d circuits.*

5.6 Approaches for THR $\circ$ THR Circuit Lower Bounds

In this section we propose approaches for proving NEXP $\not\subset$ THR $\circ$ THR. We will see that surprisingly weak algorithms suffice for proving this lower bound.

Applying Theorem 1.4.1 and the fact that $\oplus_2 \circ$ THR $\circ$ THR $\subseteq$ THR $\circ$ THR, we first show that NEXP $\not\subset$ THR $\circ$ THR would follow from a non-trivial CAPP algorithm for poly(n)-size THR $\circ$ THR circuits.

Reminder of Theorem 1.4.4. *There is an absolute constant $\delta\in(0,1)$, such that if* CAPP$_\delta$ *for* poly(n)-

size $\mathsf{THR} \circ \mathsf{THR}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time, then* $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$. *The same holds with* SAT *in place of* CAPP_δ.

Proof. The theorem for CAPP follows directly from the fact that $\oplus_2 \circ \mathsf{THR} \circ \mathsf{THR} \subseteq \mathsf{THR} \circ \mathsf{THR}$ (item (7) of Proposition 5.2.1) and Theorem 1.4.1.

Suppose SAT for $\mathrm{poly}(n)$-size $\mathsf{THR} \circ \mathsf{THR}$ circuits can be solved in $2^n/n^{\omega(1)}$ time. By Theorem 1.4.1, it suffices to give a $2^n/n^{\omega(1)}$ time algorithm for solving SAT for $\mathsf{AND}_3 \circ \mathsf{THR} \circ \mathsf{THR}$ circuits of $\mathrm{poly}(n)$ size (note that Gap-UNSAT is easier than SAT).

Given such an $\mathsf{AND}_3 \circ \mathsf{THR} \circ \mathsf{THR}$ circuit C, we first use the fact that $\mathsf{THR} \subseteq \mathsf{DOR} \circ \mathsf{ETHR}$ (item (2) of Proposition 5.2.1) to transform it into a $\mathrm{poly}(n)$ size $\mathsf{AND}_3 \circ \mathsf{DOR} \circ \mathsf{ETHR} \circ \mathsf{ETHR}$ circuit C'.

Treating the DOR as an addition gate (that has at most one true input), and the AND_3 as a multiplication, we can apply distributivity to the circuit. Together with the fact that $\mathsf{AND} \circ \mathsf{ETHR} \subseteq \mathsf{ETHR}$ (item (5) of Proposition 5.2.1), C' is then equivalent to a $\mathsf{DOR} \circ \mathsf{ETHR} \circ \mathsf{ETHR}$ circuit C'' of $\mathrm{poly}(n)$ size.

Finally, observe that solving SAT for C'' can be reduced to solving SAT for its $\mathrm{poly}(n)$ $\mathsf{ETHR} \circ \mathsf{ETHR}$ subcircuits, and note that $\mathsf{ETHR} \circ \mathsf{ETHR}$ can be converted efficiently into $\mathsf{THR} \circ \mathsf{THR}$ (item (4) of Proposition 5.2.1). Therefore, applying the $2^n/n^{\omega(1)}$ time SAT algorithm for $\mathrm{poly}(n)$-size $\mathsf{THR} \circ \mathsf{THR}$ circuits from the assumption completes the proof. ■

In fact, similar results apply to TC circuits of any constant depth d (i.e., LT_d circuits). The following theorem can be proved in exactly the same way.

Theorem 5.6.1. *There is an absolute constant* $\delta > 0$, *such that for any constant* d, *if* CAPP_δ *for* $\mathrm{poly}(n)$*-size* LT_d *circuits can be solved in* $2^n/n^{\omega(1)}$ *time, then* $\mathsf{NEXP} \not\subset \mathsf{LT}_d$. *The same is true with* SAT *in place of* CAPP.

Now, combing Theorem 1.4.4 and our equivalence theorems (Theorem 5.5.1 and Theorem 5.5.3), the following corollary follows immediately.

Reminder of Theorem 1.4.5. *There is an absolute constant* $\delta \in (0,1)$ *such that if one of the following holds:*

1. CAPP_δ *(or* SAT*) for* $\mathrm{poly}(n)$*-size* $\mathsf{THR} \circ \mathsf{MAJ}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time, or*
2. $\widetilde{\mathsf{CAPP}}$ *for* $\mathrm{poly}(n)$*-size* $\mathsf{MAJ} \circ \mathsf{MAJ}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time.*

Then $\mathsf{NEXP} \not\subset \mathsf{THR} \circ \mathsf{THR}$.

5.7 Missing Proofs

5.7.1 Constructions of Super Query-Efficient PCP of Proximity Systems

In this appendix we present proofs for Lemma 5.2.6 and Lemma 5.2.7 for completeness. We remark that we did not make any effort to optimize the soundness/completeness constants s and c in our construction; any universal constant suffices in our applications.[6]

We start with Lemma 5.2.6 (restated below).

Reminder of Lemma 5.2.6. *(3-query PCPP with perfect completeness) For any constant $\delta > 0$ there is a constant $0 < s < 1$, such that there is a PCP of proximity system for Circuit-Eval with proximity δ, soundness s, random bits $r = O(\log n)$, and query complexity $q = 3$. Moreover, the system satisfies two additional properties:*

(1) Given the random coins, the verifier simply computes an OR on these 3 queried bits or their negations, and accepts if the OR is true.

(2) Given the pair $(C, w) \in$ Circuit-Eval, we can construct a proof π in $\mathrm{poly}(|C| + |w|)$ time that makes $V(C)$ accept with probability 1.

Proof. By Lemma 5.2.4, there is a PCP of proximity system V for Circuit-Eval with proximity δ, soundness $s = 1/2$, number of random bits $r = O(\log n)$ and query complexity $q = O_\delta(1)$.

Let the circuit be C. Suppose we are given random bits $R \in \{0,1\}^r$, so that $V(C)$ queries positions $k_1 = k_1(C,R), k_2 = k_2(C,R), \ldots, k_q = k_q(C,R)$ of the oracle $z = w \circ \pi$, computes a predicate $P = P(C,R)$ on these bits, and outputs $P(z_{k_1}, z_{k_2}, \ldots, z_{k_q})$.

We construct a new PCP of proximity system V' as follows. Fix the circuit C. and let $P_R = P(C,R)$. Slightly abusing notation, we let x_j denote the bit z_{k_j}. P_R can be computed by a circuit D_R of $O_q(1)$ size; therefore P_R can computed in size S for a universal constant S only depending on q. We can construct a group of auxiliary variables $\{y_{R,\ell}\}_{\ell \in [S]}$, and a group of constraints $\{F_{w,\ell}\}_{\ell \in [S]}$, where each constraint is an OR of three bits (or their negations) from the x_j's and $y_{R,\ell}$'s, such that $P_w(x_1, x_2, \ldots, x_q) = 1$ if and only if there exists an assignment to the $y_{R,\ell}$'s such that all constraints $F_{R,\ell}$'s are satisfied.

The verifier $V'(C)$ treats its oracle as three parts. The first two parts are w and π (where π is supposed to be a proof for $V(C)$), while the third part π_y is supposed to contain assignments to

[6] Here we are actually composing the PCPP from [BSGH+06] with some trivial PCPP constructions for constant-size functions. There are much better constructions, see e.g. [Din07].

all $y_{R,\ell}$'s for all $R \in \{0,1\}^r$ and $\ell \in [S]$. $V'(C)$ first tosses r random coins to get a random string $R \in \{0,1\}^r$, then tosses $\log(S)$ more coins to pick a random integer $\ell \in [S]$. Then $V'(C)$ queries the 3 bits appearing in the constraint $F_{R,\ell}$, and accepts if and only if the constraint is satisfied by those 3 bits. We denote its proof to be $\pi' = (\pi, \pi_y)$.

We claim that $V'(C)$ is a correct PCP of Proximity system. If $(C, w) \in$ Circuit-Eval, let π be a proof such that $V(C)$ accepts $w \circ \pi$ with probability 1. Then by our construction of $V'(C)$, there is a π_y such that $V'(C)$ accepts $w \circ (\pi \circ \pi_y)$ with probability 1.

Otherwise, suppose w is δ-far from the set $\{z : C(z) = 1\}$. Then for any proof π, $V(C)$ accepts $(w \circ \pi)$ with probability at most $1/2$. This means for all additional proofs π_y, at least a $1/2$-fraction of $R \in \{0,1\}^r$ are such that at least one constraint from $\{F_{R,\ell}\}_{\ell \in [S]}$ is not satisfied by $w \circ (\pi \circ \pi_y)$. Therefore, $V'(C)$ rejects with probability at least $1/2 \cdot 1/S = \Omega_q(1) = \Omega_\delta(1)$, which completes the proof. ■

In order to get a 2-query PCP of proximity system from the above, we use the following classical gadget by Garey, Johnson, and Stockmeyer [GJS76], originally used to prove the NP-hardness of MAX-2-SAT.

Lemma 5.7.1. *Let X_1, X_2, X_3 and Y be 4 Boolean variables. Consider the following 10 constraints:*

$$X_1, X_2, X_3, \neg X_1 \vee \neg X_2, \neg X_2 \vee \neg X_3, \neg X_3 \vee \neg X_1,$$
$$Y, X_1 \vee \neg Y, X_2 \vee \neg Y, X_3 \vee \neg Y.$$

If $X_1 \vee X_2 \vee X_3$, then there exists an assignment to Y such that 7 of the above constraints are satisfied. Otherwise, all assignments to Y satisfy at most 6 of the above constraints.

Reminder of Lemma 5.2.7. *(2-query PCPP with constant completeness/soundness gap) For any constant $\delta > 0$ there two constants $0 < s < c < 1$, such that there is a PCP of proximity system for* Circuit-Eval *with proximity δ, soundness s, completeness c, number of random bits $r = O(\log n)$ and query complexity $q = 2$. Moreover, the system satisfies two additional properties:*

(1) Given the random coins, the verifier computes an OR *on the 2 queried bits or their negations, and accepts iff the* OR *is true.*

(2) Given the pair $(C, w) \in$ Circuit-Eval, a proof π can be constructed in $\text{poly}(|C| + |w|)$ time that makes $V(C)$ accept with probability at least c.

Proof. By Lemma 5.2.6, there is a PCP of proximity system V for Circuit-Eval with proximity δ, soundness $s = s(\delta) < 1$, number of random bits $r = O(\log n)$ and query complexity $q = 3$. The verifier computes an OR on these 3 queried bits or their negations, and accepts if it is true.

Let the circuit be C. We begin as in the previous proof. Suppose we have randomness $R \in \{0,1\}^r$, and $V(C)$ queries positions $k_1 = k_1(C,R), k_2 = k_2(C,R), k_3 = k_3(C,R)$ of the oracle $z = w \circ \pi$, computes a predicate $P = P(C,R)$ on these bits, then outputs $P(z_{k_1}, z_{k_2}, z_{k_3})$. Slightly abusing notation, we use x_j to denote the bit z_{k_j}. By Lemma 5.2.6, we can assume

$$P(x_1, x_2, x_3) = \vee_{j \in [3]}(x_j \oplus b_j),$$

where $b_j = b_j(C,R)$ is whether it negates the bit x_j.

Our new PCP of proximity system V' works as follows. Fix the circuit C and let $P_R = P(C,R)$. By Lemma 5.7.1, we can construct an auxiliary variable y_R and a group of constraints $\{F_{R,\ell}\}_{\ell \in [10]}$, each is an OR of 2 bits (or their negations) from x_j's and y_R[7] such that if $P_w(x_1, x_2, x_3) = 1$ then there is an assignment to the y_R such that 7 constraints from $\{F_{R,\ell}\}_{\ell \in [10]}$ are satisfied; otherwise, for all assignments to y_R, at most 6 constraints from $\{F_{R,\ell}\}_{\ell \in [10]}$ are satisfied.

As in the 3-query PCPP, $V'(C)$ treats its oracle as three parts: the first two are w and π (π is intended to be a proof in $V(C)$), and the third part π_y is intended to contain assignments to all y_R's, for all $R \in \{0,1\}^r$. Our $V'(C)$ first tosses r random coins to get $R \in \{0,1\}^r$, then tosses $O(1)$ more coins to pick a random integer $\ell \in [10]$. Then it simply queries the 2 bits appearing in the constraint $F_{R,\ell}$, and accepts iff that constraint is satisfied. We denote its proof to be $\pi' = (\pi, \pi_y)$.

Let us argue $V'(C)$ satisfies our requirement. If $(C,w) \in$ Circuit-Eval, let π be a proof such that $V(C)$ accepts $w \circ \pi$ with probability 1. Then by our construction of $V'(C)$, there is a π_y such that $V'(C)$ accepts $w \circ (\pi \circ \pi_y)$ with probability at least $7/10$.

Now suppose w is δ-far from the set $\{z : C(z) = 1\}$. Then for all proofs π, $V(C)$ accepts $(w \circ \pi)$ with probability at most s. This means that for any additional proof π_y, there is at most an s-fraction of $R \in \{0,1\}^r$ such that 7 constraints from $\{F_{R,\ell}\}_{\ell \in [S]}$ are satisfied by $w \circ (\pi \circ \pi_y)$; for the remaining R's, at most 6 constraints from $\{F_{R,\ell}\}_{\ell \in [S]}$ are satisfied. Therefore, $V'(C)$ accepts with probability at most $s \cdot 7/10 + (1-s) \cdot 6/10 < 7/10$, which completes the proof. ∎

[7] In Lemma 5.7.1, constraint X_i can be written as $X_i \vee X_i$, which is an OR of 2 bits.

5.7.2 Proofs for $\mathsf{THR} \subseteq \mathsf{DOR} \circ \mathsf{ETHR}$ and $\oplus_k \circ \mathsf{THR} \circ \mathsf{THR} \subseteq \mathsf{THR} \circ \mathsf{THR}$

Here we present an alternative proof that $\mathsf{THR} \subseteq \mathsf{DOR} \circ \mathsf{ETHR}$, which has a better weight dependence than prior work [HP10] and is arguably simpler. We first give a construction for the special case when all the weights and the threshold value are non-negative. Then we show that the general case can be easily reduced to this case.

Lemma 5.7.2. *Let G be a THR gate on n bits defined as $G(x) := [\sum_{i=1}^{n} w_i \cdot x_i > T]$, such that all w_i's and T are integers in $[0, 2^L - 1]$ for some $L \in \mathbb{N}$. Then G can be written as a DOR of $O(n \cdot L)$ many ETHR gates, each with weights and threshold from $[0, 2^{L+1} - 1]$.*

Proof. For each weight $w_i \in [0, 2^L - 1]$, write it in its binary representation

$$w_{i,L}, w_{i,L-2}, \ldots, w_{i,1} \in \{0,1\}^L,$$

such that

$$w_i = \sum_{j=1}^{L} 2^{j-1} \cdot w_{i,j}.$$

In this way, we can view w as a Boolean matrix from $\{0,1\}^{n \times L}$. For each position $(a, b) \in [n] \times [L]$, we build a partial matrix $w^{(a,b)}$ as follows: for $(i, j) \in [n] \times [L]$,

$$w^{(a,b)}_{i,j} = \begin{cases} w_{i,j} & (j > b) \text{ or } ((j = b) \text{ and } (i \geq a)) \\ 0 & \text{otherwise} \end{cases}$$

That is, $w^{(a,b)}$ is the sub-matrix of w, consisting of entries which are either to the right of (a, b), or directly above (a, b). (We number the rows of the matrix from bottom to top, and the columns of the matrix from left to right.)

Given $x \in \{0,1\}^n$, we define

$$w^{(a,b)} \cdot x := \sum_{i=1}^{n} \left(\sum_{j=1}^{L} 2^{j-1} \cdot w^{(a,b)}_{i,j} \right) \cdot x_i.$$

That is, we treat each row of $w^{(a,b)}$ as the L-bit binary representation of the corresponding weight on x_i. By definition, we have $w^{(1,1)} \cdot x = w \cdot x = \sum_{i=1}^{n} w_i \cdot x_i$.

Now, fix $x \in \{0,1\}^n$, and consider the sequence S, defined as:

$$w^{(n,L)} \cdot x, w^{(n-1,L)} \cdot x, \ldots, w^{(1,L)} \cdot x, w^{(n,L-1)} \cdot x, \ldots, w^{(1,L-1)} \cdot x, \ldots, w^{(n,1)} \cdot x, \ldots, w^{(1,1)} \cdot x.$$

By definition of $w^{(a,b)}$, we are including 1-entries of the matrix w one-by-one, hence the sequence S is non-decreasing (and begins with 0).

Suppose $w \cdot x = w^{(1,1)} \cdot x > T$. Then there must be a *unique* position $(a,b) \in [n] \times [L]$ such that $w^{(a,b)} \cdot x$ is the first value in the sequence S which is greater than T. For each $(a,b) \in [n] \times [L]$, we will use an ETHR gate $E^{(a,b)}$ to specify the condition that $w^{(a,b)} \cdot x$ is the first value greater than T from S. Then when $G(x)$ is true, exactly one of the $E^{(a,b)}(x)$'s is true, and when $G(x)$ is false, all of the $E^{(a,b)}(x)$'s are false.

To see that an ETHR gate suffices, we observe that $w^{(a,b)} \cdot x$ is the first value greater than T from the sequence S, if and only if the following conditions hold:

1. $w^{(a,b)} \cdot x > T$ *(it is greater than T)*,
2. $(w_{a,b} = 1) \wedge (x_a = 1)$ *(it is bigger than the previous value)*, and
3. $w^{(a,b)} \cdot x - 2^{b-1} \leq T$ *(the previous value is no greater than T)*.

We crucially observe that $w^{(a,b)} \cdot x$ is a multiple of 2^{b-1}. In the matrix $w^{(a,b)}$, we only include nonzero $w_{i,j}$'s where $j \geq b$. Thus in $w^{(a,b)} \cdot x$, every 1 in x is getting multiplied by a power of two which is at least 2^{b-1}.

By division, $T = 2^{b-1} \cdot T_b + T_r$, for some $0 \leq T_r < 2^{b-1}$ and $T_b > 0$. Then $w^{(a,b)} \cdot x > T$ if and only if $w^{(a,b)} \cdot x \geq 2^{b-1} \cdot (T_b + 1)$. Furthermore, $w^{(a,b)} \cdot x - 2^{b-1} \leq T$ if and only if $w^{(a,b)} \cdot x \leq 2^{b-1} \cdot (T_b + 1)$. Therefore, the above conditions are equivalent to

1. $(w_{a,b} = 1) \wedge (x_a = 1)$, and
2. $w^{(a,b)} \cdot x = 2^{b-1} \cdot (T_b + 1)$.

Now all these conditions are linear equations, so we can define an ETHR function $E^{(a,b)}$ that checks all of them. In particular, set $E^{(a,b)}$ to be the constant function $\mathbf{0}$ if $w_{a,b} = 0$; otherwise set

$$E^{(a,b)}(x) := \left[(2 \cdot w^{(a,b)} \cdot x) + x_a = 2^b \cdot (T_b + 1) + 1 \right].$$

This completes the proof. ■

Now we reduce the general case to the non-negative weights and thresholds case, and complete the reduction from THR to DOR $\circ$ ETHR.

Lemma 5.7.3. *Let G be a* THR *gate on n bits, $G(x) := [\sum_{i=1}^{n} w_i \cdot x_i > T]$, such that all w_i's and T are integers from $[-W, W]$ for some $W \in \mathbb{N}$. Then G can be written as a* DOR *of $O(n \cdot \log W)$ many* ETHR *gates, each with weights and threshold from $[-\Theta(W), \Theta(W)]$.*

Proof. We start by defining n new variables $z_1, z_2, \ldots, z_n \in \{0,1\}^n$. Set $z_i := x_i$ if $w_i \geq 0$, and $z_i := 1 - x_i$ otherwise. Letting $S = \{i : w_i \geq 0\}$, we have

$$\begin{aligned}\sum_{i=1}^{n} w_i \cdot x_i &= \sum_{i \in S} w_i \cdot z_i + \sum_{i \notin S} w_i \cdot (1 - z_i) \\ &= \sum_{i \in S} w_i \cdot z_i + \sum_{i \notin S} -w_i \cdot z_i + \sum_{i \notin S} w_i.\end{aligned}$$

Let $\widehat{w}_i = |w_i|$, and $\widehat{T} = T - \sum_{i \notin S} w_i$. Observe that

$$\left[\sum_{i=1}^{n} w_i \cdot x_i > T\right] \Leftrightarrow \left[\sum_{i=1}^{n} \widehat{w}_i \cdot z_i > \widehat{T}\right].$$

If $\widehat{T}$ is negative, then $G(x) = 1$ on all Boolean inputs x (since all $\widehat{w}_i$'s are non-negative, and the z_i are Boolean) and we are done. Otherwise, we can apply Lemma 5.7.2 with $\widehat{G}(z) := \left[\sum_{i=1}^{n} \widehat{w}_i \cdot z_i > \widehat{T}\right]$. Substituting each z_i by $1 - x_i$, we obtain the desired DOR decomposition for $G(x)$. ■

Lemma 5.7.4. *Let k be a constant, a $\oplus_k \circ$* THR $\circ$ THR *circuit of $s = s(n)$ size on n bits is equivalent to a* THR $\circ$ THR *circuit of $s^{O(k)}$ size. Moreover, the corresponding* THR $\circ$ THR *circuit can be constructed deterministically in $s^{O(k)}$ time.*

Proof. First, by Lemma 5.7.3, the given $\oplus_k \circ$ THR $\circ$ THR circuit can be transformed into a $\oplus_k \circ$ THR $\circ$ ETHR circuit C of $t = \text{poly}(s)$ size.

Let $C_1, C_2, \ldots, C_k$ be the THR $\circ$ ETHR subcircuits of C. For each C_i, let $E_{i,1}, \ldots, E_{i,t}$ be its ETHR gates, $w_{i,1}, \ldots, w_{i,t}$ be the corresponding weights in the output threshold function, and T_i be the threshold value of the output threshold function. We have

$$C_i(x) := \left[\sum_{j=1}^{t} w_{i,j} \cdot E_{i,j}(x) > T_i\right].$$

By slightly perturbing the $w_{i,j}$'s and T_i, we can ensure that $\sum_{j=1}^{t} w_{i,j} \cdot E_{i,j}(x) - T_i$ is never equal to

0, over all $x \in \{0,1\}^n$. Next, we define

$$F(x) = \left[\prod_{i=1}^{k} \left(T_i - \sum_{j=1}^{t} w_{i,j} \cdot E_{i,j}(x) \right) < 0 \right]. \tag{5.3}$$

Noting that $C_i(x) = 1$ if and only if $T_i - \sum_{j=1}^{t} w_{i,j} \cdot E_{i,j}(x) < 0$, we observe that $F(x) = 1$ when an odd number of $C_i(x)$'s are 1, and $F(x) = 0$ otherwise. Therefore, F computes the same function as the original circuit C.

Finally, expanding the product of k sums in (5.3) into a sum of $s^{O(k)}$ products, and recalling that $\mathsf{AND} \circ \mathsf{ETHR} \subseteq \mathsf{ETHR}$ (Proposition 5.2.1), F can be written as a $\mathsf{THR} \circ \mathsf{ETHR}$ circuit. Converting this back to a $\mathsf{THR} \circ \mathsf{THR}$ circuit (Proposition 5.2.1), the proof is complete. ■

Chapter 6

Refuters for NTIME Hierarchy Theorems

The goal of this chapter is to develop the refuter framework and prove Theorem 1.5.8 and Theorem 1.5.9 (restated blow).

Reminder of Theorem 1.5.8. *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, $\mathsf{NTIME}[T(n)] \not\subset$ i.o.-$\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.*

Reminder of Theorem 1.5.9. *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, there is a language $L \in \mathsf{NTIME}[T(n)]$ and an algorithm $\mathcal{R}$ such that:*

1. ***Input.*** *The input to $\mathcal{R}$ is a pair $(M, 1^n)$, with the promise that M describes a nondeterministic Turing machine running in $o(T(n))$ time and guessing at most $n/10$ bits.*
2. ***Output.*** *For every fixed M and every sufficiently large n, $\mathcal{R}(M, 1^n)$ outputs a string $x \in \{0,1\}^n$ such that $M(x) \neq L(x)$.*
3. ***Complexity.*** *$\mathcal{R}$ runs in $\mathrm{poly}(T(n))$ time with adaptive access to an SAT oracle.*

Since $\mathcal{R}$ can find counterexamples to any faster algorithm attempting to decide L, we call $\mathcal{R}$ a refuter.

We begin with some notation in Section 6.1. In Section 6.2, we define the hard language L^T_{FS} and prove Theorem 1.5.8. In Section 6.3, we construct the corresponding refuter for Theorem 1.5.8 and thus prove Theorem 1.5.9. In Section 6.4, we discuss an adaptation to the "robustly often" lower bound of [FS17], which will be useful in the construction of rigid matrices.

6.1 Preliminaries

For a definition of $\mathsf{NTIMEGUESS}[T(n), g(n)]$, see Definition 3.4.1. We fix a natural enumeration of all nondeterministic Turing machines. Note that the specific model does not really matter, see

the discussion in [Wil13a, Section 2.1]. We use the integer M to denote the M-th nondeterministic Turing machine in the enumeration. For simplicity, we assume all machines are random-access machines in the following.

For a nondeterministic Turing machine M, an input x, and a witness z, let $\mathcal{V}_M(x, T, z)$ denote the result of running M on the input x for at most T steps with witness z. More precisely:

- $\mathcal{V}_M(x, T, z) = \perp$ if the simulation fails. That is, either (1) M does not stop after T steps, or (2) M guesses more than $|z|$ bits during the simulation.
- Otherwise, if the simulation succeeds, $\mathcal{V}_M(x, T, z) = 1$ if M accepts x with witness z, and $\mathcal{V}_M(x, T, z) = 0$ otherwise. (It is possible that M only uses a prefix of z as the witness.)

Note that for an M using $o(T(n))$ time and $g(n)$ guesses, we have

$$M(x) = 1 \Leftrightarrow \bigvee_{w \in \{0,1\}^{g(n)}} \mathcal{V}_M(x, T(n), w) = 1.$$

We also need a standard encoding of two integers and a Boolean string. Given $M, n \in \mathbb{N}$ and $z \in \{0,1\}^*$, we encode them by

$$\langle M, n, z \rangle = 1^M 01^{(n-M-2-|z|)} 0z.$$

Note that we require $n \geq M + 2 + |z|$ for the encoding to be valid. Observe that $|\langle M, n, z \rangle| = n$. Given an input $x \in \{0,1\}^\star$, one can unambiguously determine the triple M, n, z such that $\langle M, n, z \rangle = x$, or that x is not a valid encoding of any triple.

6.2 Almost-everywhere NTIME Hierarchy with Sublinear Witness Length

To provide more intuition for our results, we first prove the almost-everywhere NTIME hierarchy theorem of Fortnow and Santhanam [FS16].

Reminder of Theorem 1.5.8. *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, $\mathsf{NTIME}[T(n)] \not\subset$ i.o.-$\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.*

An Almost-Everywhere Diagonalizing Language. We start by defining a language constructed by diagonalization (adapted from [FS17, FS16]), which is used as the hard language in Theorem 1.5.8. Later in Section 6.3, we will construct an efficient refuter for this language as well.

Definition 6.2.1. *For every time-constructible function $T(n)$ such that $n \le T(n) \le 2^{\mathrm{poly}(n)}$, we define a nondeterministic algorithm $\mathcal{A}^T_{\mathsf{FS}}$ as follows:*

- *Given an input x, parse it as $x = \langle M, n, z\rangle$, and reject immediately if there is no valid parsing, or if $|z| \neq n/10$.*[1]
- *$\mathcal{A}^T_{\mathsf{FS}}$ accepts x if and only if both of the following hold:*
  - *M rejects $\langle M, n, 0^{n/10}\rangle$ in $T(n)$ steps with witness z. That is, $\mathcal{V}_M(\langle M, n, 0^{n/10}\rangle, T(n), z) = 0$.*
  - *(This condition is only required for $z \neq 1^{n/10}$.) M accepts $\langle M, n, z+1\rangle$ in $T(n)$ steps while guessing at most $n/10$ bits.*[2]

By the construction above, it is clear that $\mathcal{A}^T_{\mathsf{FS}}$ is an $\mathsf{NTIME}[T(n)]$ algorithm. We let $L^T_{\mathsf{FS}} \in \mathsf{NTIME}[T(n)]$ be the language computed by $\mathcal{A}^T_{\mathsf{FS}}$.

The following list of inputs $\{x_i^{M,(n)}\}$ greatly simplifies our discussion and the proof. For clarity, let $w_1^{(n)}, w_2^{(n)}, \ldots, w_{2^{n/10}}^{(n)}$ be the list of all $2^{n/10}$-length binary strings, sorted in lexicographical order. We define

$$x_i^{M,(n)} := \langle M, n, w_i^{(n)}\rangle.$$

When M is clear from the context, we omit the superscript M, and use $x_i^{(n)}$ to denote $x_i^{M,(n)}$. Note that when the encoding is valid, all strings $x_i^{(n)}$ have length exactly n.

The following lemma summarizes the behavior of $\mathcal{A}_{\mathsf{FS}}$ on the strings in the list $\{x_i^{M,(n)}\}$ when M is an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm, which will be used frequently in the rest of the section.

Lemma 6.2.2. *Suppose M runs in $o(T(n))$ time and guesses no more than $n/10$ bits of witness. For every sufficiently large n and $i \in \{1, \ldots, 2^{n/10}\}$, we have*[3]

$$\mathcal{A}_{\mathsf{FS}}(x_i^{(n)}) = \begin{cases} [\mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0] \wedge M(x_{i+1}^{(n)}) & 1 \le i < 2^{n/10} \\ [\mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0] & i = 2^{n/10}. \end{cases}$$

Proof. Since M is an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm, for every sufficiently large n, the algorithm $\mathcal{A}^T_{\mathsf{FS}}$ can simulate M faithfully on the entire collection of inputs $\{x_i^{(n)}\}$. The conclusion follows from the definition of $\mathcal{A}^T_{\mathsf{FS}}$. ∎

[1] Here, and in the rest of this section, $n/10$ means $\lfloor n/10\rfloor$ when n is not a multiple of 10.

[2] We use $z+1$ to denote the lexicographically next string after z in $\{0,1\}^{n/10}$.

[3] In the following, we use the Iverson bracket notation where for a Boolean-valued predicate P, $[P] = 1$ if P is true and $[P] = 0$ otherwise.

Now we turn to the proof of Theorem 1.5.8.

Proof of Theorem 1.5.8. It suffices to show that $L^T_{\mathsf{FS}} \notin \text{i.o.-}\mathsf{NTIMEGUESS}[o(T(n)), n/10]$, *i.e.*, the language decided by $\mathcal{A}^T_{\mathsf{FS}}$ cannot be decided in $o(T(n))$ time with at most $n/10$ nondeterministic guess bits.

Let M be an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ machine. We will show that for every sufficiently large n, the machine M and $\mathcal{A}^T_{\mathsf{FS}}$ cannot agree on all inputs in $\{0,1\}^n$.

So for the sake of contradiction, suppose for large enough n that $M(x) = \mathcal{A}^T_{\mathsf{FS}}(x)$ holds for all $x \in \{0,1\}^n$. In particular, this means for all $i \in [2^{n/10}]$, we have $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = M(x_i^{(n)})$.

Consider one special input: $x_1^{(n)} = \langle M, n, w_1^{(n)} \rangle$. We consider the two possible outputs of $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)})$, and show that both of them lead to a contradiction.

- **Case 1: $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 1$.** By assumption, $M(x_1^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 1$. We show that $M(x_i^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$ by induction on i. The base case of $i = 1$ is already established. Assuming $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = M(x_i^{(n)}) = 1$, it is easy to see from Lemma 6.2.2 that $M(x_{i+1}^{(n)}) = 1$ as well, hence also $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = M(x_{i+1}^{(n)}) = 1$ by assumption.
  We conclude that $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$, which in turn implies (by Lemma 6.2.2)

$$\text{for all } i \in [2^{n/10}], \quad \mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0.$$

  This means that M on $x_1^{(n)}$ rejects every possible witness, and hence $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = M(x_1^{(n)}) = 0$, a contradiction.
- **Case 2: $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 0$.** By assumption $M(x_1^{(n)}) = 0$. That is, M on $x_1^{(n)}$ rejects every witness, which means

$$\text{for all } i \in [2^{n/10}], \quad \mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0. \tag{6.1}$$

  By Lemma 6.2.2, we have $M(x_{2^{n/10}}^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_{2^{n/10}}^{(n)}) = 1$. Using backward induction (from $i = 2^{n/10}$ down to $i = 1$), we will show that $M(x_i^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$.
  First, the base case of $i = 2^{n/10}$ is already established. Now, assuming $M(x_i^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$, it is easy to see from (6.1) and Lemma 6.2.2 that $M(x_{i-1}^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_{i-1}^{(n)}) = 1$ as well. Therefore, we conclude that $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$, which contradicts our assumption that $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 0$.

Therefore, we conclude that for all such M and for every sufficiently large n, M fails to compute

L^T_{FS} on inputs of length n. This completes the proof. ■

6.3 Construction of the Refuter

The proof above shows that every $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ machine fails to compute L^T_{FS} on all sufficiently large input lengths: for every such machine M and sufficiently large n, there is an $x_n \in \{0,1\}^n$ such that $M(x_n) \neq \mathcal{A}^T_{\mathsf{FS}}(x_n)$. Now we design an algorithm to find such an x_n efficiently.

To begin with, we need the following binary search algorithm.

Lemma 6.3.1. *There is an algorithm A satisfying the following.*

- ***Input.*** *A is given an explicit integer $n \geq 2$ (written in binary form) as input, together with oracle access to a list $(a_1, \ldots, a_n) \in \{0,1\}^n$ such that $a_1 \neq a_n$.*
- ***Output.*** *An index $p \in [1, n-1]$ such that $a_p \neq a_{p+1}$.*
- ***Efficiency.*** *A runs in $O(\log n)$ time, makes at most $O(\log n)$ queries to the list.*

Proof. The algorithm A works as follows.

1. **Initialize:** We set $L = 1, R = n$ and query a_L, a_R. From the promise on input we have $a_L \neq a_R$.
2. As long as $R - L \geq 2$, we set $q = \lfloor \frac{L+R}{2} \rfloor$ and query a_q. Since $a_L \neq a_R$, a_q cannot equal to both of them. If $a_q \neq a_L$, we update $R = q$, otherwise we update $L = q$. Note that the invariant $a_L \neq a_R$ always holds.
3. We repeat Step (2) until $R - L \leq 1$. Since $a_L \neq a_R$, it must be the case $R = L + 1$. Return L.

■

We need a special form of this algorithm, which we state as a corollary below.

Corollary 6.3.2. *There is an algorithm A' satisfying the following.*

- ***Input.*** *A' is given an explicit integer $n \geq 1$ (written in binary form) as input, together with oracle access to two lists $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in \{0,1\}^n$, with the promise that for every $i < n$, $a_i = b_{i+1}$, as well as $b_1 \neq a_n$.*
- ***Output.*** *An index $p \in [1, n]$ such that $a_p \neq b_p$.*
- ***Efficiency.*** *A' runs in $O(\log n)$ time, makes at most $O(\log n)$ queries to the list.*

Proof. A' simulates A from Lemma 6.3.1 with parameter $n+1$ and the list $(b_1, \ldots, b_n, a_n)$. Since $a_n \neq b_1$ from our assumption, this list satisfies the promise of Lemma 6.3.1, and all queries to

the list can be answered by querying the two lists $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ accordingly. A' then outputs the index $p \in [1, n]$ reported by A. Now we can see that $a_p \neq b_p$: If $p < n$, then $a_p = b_{p+1} \neq b_p$ as desired, otherwise we have $a_n \neq b_n$, which is also valid. ■

We are now ready to construct a refuter algorithm that explicitly witnesses the hardness of the language $\mathcal{A}^T_{\mathsf{FS}}$ from Theorem 6.4.1.

Theorem 6.3.3 (Formal version of Theorem 1.5.9)**.** *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, there is an algorithm $\mathcal{R}^T$ such that:*

1. ***Input.*** *The input for $\mathcal{R}^T$ is a pair $(M, 1^n)$, with the promise that the M-th nondeterministic Turing machine runs in $o(T(n))$ time and guesses no more than $n/10$ bits of witness.*
2. ***Output.*** *For every fixed M and all large enough n, $\mathcal{R}^T(M, 1^n)$ outputs a string $x \in \{0,1\}^n$ such that $\mathcal{A}^T_{\mathsf{FS}}(x) \neq M(x)$.*
3. ***Complexity.*** *$\mathcal{R}^T$ is a deterministic algorithm running in $\mathrm{poly}(T(n))$ time with adaptive access to a SAT oracle.*

Since the output of $\mathcal{R}^T$ can explicitly refute any $o(T(n))$-time nondeterministic algorithm which claims to decide L^T_{FS}, we also call $\mathcal{R}^T$ a refuter.

Proof. Let $(M, 1^n)$ be an input to $\mathcal{R}^T$. Suppose M satisfies the stated promise. The proof of Theorem 1.5.8 shows that a *differing input* must exist in the list $\{x_i^{M,(n)}\}$ (that is, $M(x_j^{M,(n)}) \neq \mathcal{A}^T_{\mathsf{FS}}(x_j^{M,(n)})$ for some $j \in [2^{n/10}]$). In the following, we fix the machine M and write $x_i^{M,(n)}$ instead of $x_i^{(n)}$ for brevity. We now consider the following two cases:

- **Case 1:** $M(x_1^{(n)}) = 0$. In this case, M on $x_1^{(n)}$ rejects every witness, then Lemma 6.2.2 implies

$$\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = \begin{cases} M(x_{i+1}^{(n)}) & \text{if } i < 2^{n/10}, \\ 1 & \text{if } i = 2^{n/10}. \end{cases}$$

  The pair of lists $(\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}), \ldots, \mathcal{A}^T_{\mathsf{FS}}(x_{2^{n/10}}^{(n)}))$ and $(M(x_1^{(n)}), \ldots, M(x_{2^{n/10}}^{(n)}))$ satisfy the promise of Corollary 6.3.2 since $M(x_1^{(n)}) = 0 \neq \mathcal{A}^T_{\mathsf{FS}}(x_{2^{n/10}}^{(n)}) = 1$. We then invoke the algorithm in Corollary 6.3.2 to find an index p such that $\mathcal{A}^T_{\mathsf{FS}}(x_p^{(n)}) \neq M(x_p^{(n)})$. The query to the lists can be answered with the help of the SAT oracle in $\mathrm{poly}(T(n))$ time. Therefore, the whole algorithm runs in $\mathrm{poly}(T(n)) \cdot O(\log 2^{n/10}) = \mathrm{poly}(T(n))$ time.
- **Case 2:** $M(x_1^{(n)}) = 1$. In this case, M on $x_1^{(n)}$ accepts at least one witness. Let j be the minimum index from $[2^{n/10}]$ such that the witness $w_j^{(n)}$ is accepted by $M(x_1^{(n)})$. By a binary

search, the index j can be found with the help of the SAT oracle in $\mathrm{poly}(T(n))$ time. Now we focus on the partial list $(x_1^{(n)}, \ldots, x_j^{(n)})$, given that $w_j^{(n)}$ is the first witness accepted by $M(x_1^{(n)})$, Lemma 6.2.2 implies

$$\mathcal{A}_{\mathsf{FS}}^T(x_i^{(n)}) = \begin{cases} M(x_{i+1}^{(n)}) & \text{if } i < j, \\ 0 & \text{if } i = j. \end{cases}$$

The pair of lists $(\mathcal{A}_{\mathsf{FS}}^T(x_1^{(n)}), \ldots, \mathcal{A}_{\mathsf{FS}}^T(x_j^{(n)}))$ and $(M(x_1^{(n)}), \ldots, M(x_j^{(n)}))$ satisfy the promise of Corollary 6.3.2 since $M(x_1^{(n)}) = 1 \neq \mathcal{A}_{\mathsf{FS}}^T(x_j^{(n)}) = 0$. We can then invoke the algorithm in Corollary 6.3.2 to find an index p such that $\mathcal{A}_{\mathsf{FS}}^T(x_p^{(n)}) \neq M(x_p^{(n)})$. Each query to the lists can be answered with the help of the SAT oracle in $\mathrm{poly}(T(n))$ time. Therefore, the whole algorithm runs in $\mathrm{poly}(T(n)) \cdot O(\log 2^{n/10}) = \mathrm{poly}(T(n))$ time.

In summary, the algorithm works well in both cases and the theorem is proved.

■

6.4 Refuter for "Robustly Often" Lower Bounds

Finally, we generalize our refuter construction to the "robustly often" NTIME hierarchy (Theorem 6.4.1) by Fortnow and Santhanam [FS17].

Theorem 6.4.1. *For any non-decreasing time-constructible function $T(n)$ such that $T(n) \geq n$ and $T(n+1) = O(T(n))$, there exists a language $L \in \mathsf{NTIME}[T(n)]$ such that, for any $L' \in \mathsf{NTIME}[o(T(n))]$, for all but finitely many n, there exists $m \in [n, n + T(n)]$ such that L and L' cannot agree on all inputs of length m.*

Theorem 6.4.1 also yields a corresponding refuter. An advantage of this refuter is that it does not have restrictions on the size of the witness, which will be crucial later in our construction of rigid matrices. A drawback of this refuter is it can only output a "bad" input having length in an interval $[n, n + T(n)]$.

Theorem 6.4.2. *For any non-decreasing time-constructible function $T(n)$ such that $T(n) \geq n$ and $T(n+1) = O(T(n))$, there is an $\mathsf{NTIME}[T(n)]$ machine $\mathcal{A}_{\mathsf{RO}}^T$ and an algorithm $\mathcal{R}_{\mathsf{RO}}^T$ such that:*

1. ***Input.*** *The input for $\mathcal{R}_{\mathsf{RO}}^T$ is a pair $(M, 1^n)$ with the promise that the M-th nondeterministic Turing machine runs in $o(T(n))$ time.*

2. ***Output.*** *For every fixed M and all large enough n, $\mathcal{R}^T_{\mathsf{RO}}(M, 1^n)$ outputs a string x such that $|x| \in [n, n + T(n)]$ and $\mathcal{A}^T_{\mathsf{RO}}(x) \neq M(x)$.*
3. ***Complexity.*** *$\mathcal{R}^T_{\mathsf{RO}}$ is a deterministic algorithm running in $\mathrm{poly}(T(\mathrm{poly}(T(n))))$ time with adaptive access to a SAT oracle.*

The proof is similar to the proof of Theorem 1.5.8 and Theorem 6.3.3.

Proof Sketch. Informally, the intuition behind the proof of Theorem 1.5.8 is as follows: We designed the algorithm $\mathcal{A}^T_{\mathsf{FS}}$ such that, if an algorithm M accepts a particular string $x = \langle M, n, 0^{n/10} \rangle$, then by the design of $\mathcal{A}^T_{\mathsf{FS}}$ and the assumption that M and $\mathcal{A}^T_{\mathsf{FS}}$ agree on all n-bit inputs, $\mathcal{A}^T_{\mathsf{FS}}$ (implicitly) enumerates all possible witnesses for $M(x)$, and forces $M(x)$ to reject all of them. This implies that M rejects x, which is a contradiction. The case where M rejects the string x can be handled similarly.

If we no longer have the restriction that M only guesses $n/10$ bits as the witness, then $\mathcal{A}^T_{\mathsf{FS}}$ cannot enumerate all witnesses on the same input length. Instead, we can first pad the input length to be sufficiently long (i.e. $n + T(n)$), then follow the same approach: enumerate all possible witnesses, and force $M(x)$ to reject all of them.

Encoding. Recall that we fix a natural enumeration of all nondeterministic Turing machines, and associate the integer M with the M-th such machine. In the following, we use a slightly different encoding than that in Theorem 6.3.3, to deal with the fact that the witness could be longer than n. For $M, n \in \mathbb{N}$ such that $n \geq M + 2$ and $z \in \{0,1\}^*$, we encode them by

$$\langle M, n, z \rangle_{\mathsf{ro}} := 1^M 0 1^{n-M-2} 0 z.$$

Note that $|\langle M, n, z \rangle_{\mathsf{ro}}| = n + |z|$.

The Algorithm $\mathcal{A}^T_{\mathsf{RO}}$. $\mathcal{A}^T_{\mathsf{RO}}$ is defined as follows.

- Given an input x, we parse it as $x = \langle M, n, z \rangle_{\mathsf{ro}}$. We reject immediately if there is no valid parsing, or $|z| > T(n)$.
- If $|z| < T(n)$, $\mathcal{A}^T_{\mathsf{RO}}$ accepts x if both of the following hold:
  1. $z = 0^\ell$ for some ℓ.
  2. M accepts $x' = \langle M, n, 0^{|z|+1} \rangle_{\mathsf{ro}}$ in $T(|x'|)$ steps.
- Otherwise, $|z| = T(n)$, and $\mathcal{A}^T_{\mathsf{RO}}$ accepts x if both of the following hold:

1. M rejects $\langle M, n, \varepsilon \rangle_{\mathsf{ro}}$ in $T(n)$ steps with witness z.[4] That is, $\mathcal{V}_M(\langle M, n, \varepsilon \rangle_{\mathsf{ro}}, T(n), z) = 0$.
2. (This condition is only required when $z \neq 1^{T(n)}$.) M accepts $x' = \langle M, n, z+1 \rangle_{\mathsf{ro}}$ in $T(|x'|)$ steps.

By the construction above, the assumption $T(n+1) \leq O(T(n))$, and the fact that nondeterministic algorithms can be simulated with only constant overhead in running time, $\mathcal{A}^T_{\mathsf{RO}}$ is a nondeterministic algorithm running in $O(T(n))$ time.

Construction of the Refuter. Now we design the refuter $\mathcal{R}^T_{\mathsf{RO}}$. Let M be an $o(T(n))$-time nondeterministic machine. For every sufficiently large n, we want to find an input x of length $|x| \in [n, n + T(n)]$ such that $M(x) \neq \mathcal{A}^T_{\mathsf{RO}}(x)$. Let $\mathcal{L}^{(n)}$ be the list consisting of strings of the form $\langle M, n, z \rangle_{\mathsf{ro}}$ for all $z = 0^\ell$ where $\ell \in \{0, 1, 2, \ldots, T(n) - 1\}$ and $z \in \{0,1\}^{T(n)}$, sorted in lexicographical order.

The following lemma can be proved similarly as Lemma 6.2.2.

Lemma 6.4.3. *If M runs in $o(T(n))$. For every sufficiently large n and integer $1 \leq i \leq |\mathcal{L}^{(n)}|$, if $\mathcal{L}^{(n)}_i = \langle M, n, z \rangle_{\mathsf{ro}}$, then*

$$\mathcal{A}^T_{\mathsf{RO}}(\mathcal{L}^{(n)}_i) = \begin{cases} M(\mathcal{L}^{(n)}_{i+1}) & |z| < T(n), \\ \left[\mathcal{V}_M(\mathcal{L}^{(n)}_1, T(n), z) = 0\right] \wedge M(\mathcal{L}^{(n)}_{i+1}) & |z| = T(n) \text{ and } i < |\mathcal{L}^{(n)}|, \\ \left[\mathcal{V}_M(\mathcal{L}^{(n)}_1, T(n), z) = 0\right] & i = |\mathcal{L}^{(n)}|. \end{cases}$$

Applying Lemma 6.4.3, one can verify that the same algorithm in the proof of Theorem 6.3.3 also works given the list

$$\mathcal{L}^{(n)}_1, \mathcal{L}^{(n)}_2, \ldots, \mathcal{L}^{(n)}_{|\mathcal{L}^{(n)}|}.$$

Finally, note that $\mathcal{R}^T_{\mathsf{RO}}$ runs in

$$O(\log |\mathcal{L}^{(n)}|) \cdot \mathrm{poly}(T(\mathrm{poly}(T(n)))) = \mathrm{poly}(T(\mathrm{poly}(T(n))))$$

time, which completes the proof. ■

[4] We use ε to denote empty string.

Chapter 7

Average-case Lower Bounds and PRGs from an XOR Lemma

The goal of this chapter is to introduce our new XOR Lemma based on approximate linear sums (Lemma 1.5.10) and apply it to prove Theorem 1.5.2, Theorem 1.5.5, and Theorem 1.5.6. We restate them below.

Reminder of Lemma 1.5.10. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function, $\delta \in (0, \frac{1}{2})$, and $k, s \in \mathbb{N}_{\geq 1}$. Let $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right)$. If f cannot be $(1-\delta)$-approximated in ℓ_1 distance by $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity $O\left(\frac{n \cdot s}{(\delta \cdot \varepsilon_k)^2}\right)$, then $f^{\oplus k}$ cannot be $(\frac{1}{2} + \varepsilon_k)$-approximated by $\mathscr{C}$ circuits of size s.*[1]

Reminder of Theorem 1.5.5. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ such that unary NE does not admit $(1/2 + 2^{-n^\delta})$-approximate 2^{n^δ}-size $\mathscr{C}$ witnesses.*

Reminder of Theorem 1.5.2. *Let $\mathscr{C}$ be a typical and concrete circuit class. Suppose there is an $\varepsilon \in (0,1)$ such that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a language $L \in \mathsf{E}^{\mathsf{NP}}$ and a constant $\delta \in (0,1)$ such that, for every sufficiently large $n \in \mathbb{N}_{\geq 1}$, L_n cannot be $(1/2 + 2^{-n^\delta})$-approximated by $\mathscr{C}$ circuits of size 2^{n^δ}.*

Reminder of Theorem 1.5.6. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ and*

[1] Recall that the function $f^{\oplus k}$ partitions its kn-bit input into k blocks $x_1, x_2, \ldots, x_k$ of length n each, and outputs $\bigoplus_{i=1}^k f(x_i)$.

an infinity often nondeterministic PRG for 2^{n^δ}-size $\mathscr{C}$ circuits with error 2^{-n^δ}, seed-length $\mathrm{poly}(n)$, and $2^{\mathrm{poly}(n)}$ running time. Consequently, $\mathsf{MA}_{\mathscr{C}} \subseteq$ i.o.-$\mathsf{NTIME}[2^{\log^\beta n}]$ for some $\beta \in \mathbb{N}_{\geq 1}$.

In Section 7.1, we prove Lemma 1.5.10, which is crucial for proving average-case lower bounds via the algorithmic method. In Section 7.2 we give an algorithm for solving the Average-of-Product product for $\mathsf{Sum} \circ \mathscr{C}$ circuits together with some notation for the PCPP reduction. Both of them will be used frequently by later proofs. In Section 7.3, applying the new XOR Lemma and PCP of proximity, we prove Theorem 1.5.5 and Theorem 1.5.2. The proof of Theorem 1.5.2 also needs the refuter constructed in Chapter 6.

7.1 An XOR Lemma Based on Approximate Linear Sums

In this section, we prove Lemma 1.5.10. We remark that the proof is already somewhat implicit in Levin's proof of XOR Lemma [Lev87, GNW11].

Proof of Lemma 1.5.10. We prove the contrapositive, i.e., given a $\mathscr{C}$ circuit C of size s approximating $f^{\oplus k}$ on at least a $\left(\frac{1}{2} + \varepsilon_k\right)$-fraction of inputs, we show how to construct a $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit Q approximating f with a much better accuracy.

For an input x to $f^{\oplus k}$, we write $x = yz$ such that $|y| = n, |z| = (k-1)n$. Our proof is by induction on k. The case $k = 1$ is clearly trivial. Assuming the hypothesis holds for $k-1$, we now consider the following two cases.

Case 1. Suppose for some $y \in \{0,1\}^n$, we have

$$\left|\Pr_z[f^{\oplus k}(y,z) = C(y,z)] - \frac{1}{2}\right| > \frac{\varepsilon_k}{1-\delta} = (1-\delta)^{k-2} \cdot \left(\frac{1}{2} - \delta\right) = \varepsilon_{k-1}.$$

Then, we can fix one such y, and note that either circuit $C'(z) := C(y,z)$ or $\neg C'(z)$ approximates $f^{\oplus(k-1)}$ well enough so that we can reduce it to the case of $k-1$.

Case 2. Otherwise, we have that for all $y \in \{0,1\}^n$:

$$\left|\Pr_z[f^{\oplus k}(y,z) = C(y,z)] - \frac{1}{2}\right| \leq \frac{\varepsilon_k}{1-\delta}.$$

We define

$$\begin{aligned} T(y) &:= \Pr_z[C(y,z) = f^{\oplus k}(y,z)] \\ &= \Pr_z[C(y,z) = f(y) \oplus f^{\oplus(k-1)}(z)]. \end{aligned}$$

From the definition, it follows directly that

$$\left| T(y) - \frac{1}{2} \right| \leq \frac{\varepsilon_k}{1-\delta}. \tag{7.1}$$

Also, since C approximates $f^{\oplus k}$ on at least $\frac{1}{2} + \varepsilon_k$ fraction of inputs, we have

$$\mathop{\mathbb{E}}_y[T(y)] \geq 1/2 + \varepsilon_k. \tag{7.2}$$

Now let $Z_1, Z_2, \ldots, Z_\ell$ be a sequence of i.i.d. random variables, where each Z_i is uniformly random from $\{0,1\}^{n(k-1)}$. We then define

$$\widetilde{T}(y) := \mathop{\mathbb{E}}_{i \leftarrow [\ell]} \left[C(y, Z_i) = f(y) \oplus f^{\oplus(k-1)}(Z_i) \right]. \tag{7.3}$$

Setting $\ell = O\left(\frac{n}{(\delta\varepsilon_k)^2}\right)$, and applying a Chernoff bound, for every fixed $y \in \{0,1\}^n$, we have

$$\Pr_{\{Z_i\}} \left[\left| T(y) - \widetilde{T}(y) \right| \geq \frac{\delta\varepsilon_k}{2(1-\delta)} \right] \leq 2^{-n-1}.$$

By a union bound, we can fix an assignment $Z_i = z_i$ for each of Z_i such that

$$\left| T(y) - \widetilde{T}(y) \right| \leq \frac{\delta\varepsilon_k}{2(1-\delta)} \tag{7.4}$$

holds, for all $y \in \{0,1\}^n$. Then from (7.2), (7.4), and (7.1) it follows directly that

$$\mathop{\mathbb{E}}_y[\widetilde{T}(y)] \geq \frac{1}{2} + \varepsilon_k - \frac{\delta\varepsilon_k}{2(1-\delta)}, \tag{7.5}$$

and for all y,

$$\left| \widetilde{T}(y) - \frac{1}{2} \right| \leq \frac{\varepsilon_k}{1-\delta} + \frac{\delta\varepsilon_k}{2(1-\delta)}.$$

Letting $r := \frac{2\varepsilon_k + \delta\varepsilon_k}{1-\delta}$, we define:

$$\widetilde{P}(y) := \left(\frac{\widetilde{T}(y) - \frac{1}{2}}{r} + \frac{1}{2} \right).$$

Note that $\widetilde{P}(y) \in [0,1]$ since $\left|\widetilde{T}(y) - \frac{1}{2}\right| \leq \frac{r}{2}$. From (7.5), we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{y}[\widetilde{P}(y)] &= \left(\frac{\mathbb{E}_y[\widetilde{T}(y)] - \frac{1}{2}}{r} + \frac{1}{2} \right) \\
&\geq \left(\frac{\varepsilon_k - \frac{\delta\varepsilon_k}{2(1-\delta)}}{r} + \frac{1}{2} \right) \\
&= \left(\frac{\varepsilon_k - \frac{\delta\varepsilon_k}{2(1-\delta)} + \frac{\varepsilon_k}{1-\delta} + \frac{\delta\varepsilon_k}{2(1-\delta)}}{r} \right) \\
&= \frac{\varepsilon_k \left(1 + \frac{1}{1-\delta}\right)}{\varepsilon_k \cdot \frac{2+\delta}{1-\delta}} = \frac{2-\delta}{2+\delta} = 1 - \frac{2\delta}{2+\delta} \geq 1 - \delta. \qquad (7.6)
\end{aligned}$$

Finally, we use the samples $\{(z_i, f^{\oplus(k-1)}(z_i)\}$, to construct a $\mathsf{Sum} \circ \mathscr{C}$ circuit Q as follows:

$$Q(y) := \left(\frac{\Pr_i[f^{\oplus(k-1)}(z_i) \neq C(y, z_i)] - \frac{1}{2}}{r} + \frac{1}{2} \right).$$

Note that Q can be implemented as a sum of $\ell + 1$ $\mathscr{C}$ circuits (one for the constant function $\mathbf{1}$), as $f^{\oplus(k-1)}(z_i)$ can all be replaced by the corresponding constants. The size of Q is bounded by $O\left(\frac{n \cdot s}{(\delta\varepsilon_k)^2}\right)$. The sum of absolute values of coefficients in Q is bounded by $O(1/r) = O\left(\frac{n}{(\delta\varepsilon_k)^2}\right)$. Therefore, the complexity of Q is $O\left(\frac{n \cdot s}{(\delta\varepsilon_k)^2}\right)$.

Finally, note that

$$C(y, Z_i) = f(y) \oplus f^{\oplus(k-1)}(Z_i) \Leftrightarrow f(y) = C(y, Z_i) \oplus f^{\oplus(k-1)}(Z_i),$$

and therefore, from (7.3), it follows that

$$Q(y) = \begin{cases} \widetilde{P}(y) & f(y) = 1 \\ 1 - \widetilde{P}(y) & f(y) = 0 \end{cases}.$$

From the above, one can see that for all y, we have $Q(y) - f = \widetilde{P}(y) - 1$. Therefore, from (7.6)

and noting $\widetilde{P}(y) \in [0,1]$, we have

$$\left\|Q - f\right\|_1 \leq \mathop{\mathbb{E}}_{y}\left[1 - \widetilde{P}(y)\right] \leq \delta.$$

Also, since $\widetilde{P}(y) \in [0,1]$ for all y, $Q(y) \in [0,1]$ for all y as well. Putting everything together, Q is the desired $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit and this completes the proof. ■

7.2 Preliminaries

7.2.1 An Algorithm for the Average-of-Product of $\mathsf{Sum} \circ \mathscr{C}$ Circuits

First, it will be useful to define the following Average-of-Product problem, variants of which were also studied in [Wil18, CW19, CR20].

Definition 7.2.1. *Given $k, n \in \mathbb{N}$, the Average-of-Product problem takes as input k functions $f_1, f_2, \ldots, f_k \colon \{0,1\}^n \to \{0,1\}$ (may be implicitly given), and the task is to compute $\mathbb{E}_{x \in \{0,1\}^n}[\prod_{i=1}^{k} f_i(x)]$.*

We need the following lemma showing that a non-trivial CAPP algorithm for $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be used to solve the Average-of-Product problem for four $\mathsf{Sum} \circ \mathscr{C}$ circuits.

Lemma 7.2.2. *Let $\mathscr{C}$ be a typical circuit class, and let $S, E \colon \mathbb{N} \to \mathbb{N}$ be such that $S(n) \leq o(E(n))$. Suppose there is an algorithm solving CAPP on $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $E(n)$ and n inputs within an additive error of $1/E(n)$, running in $2^n/E(n)$ time. Then given four $\mathsf{Sum} \circ \mathscr{C}$ circuits $C_1, \ldots, C_4 \colon \{0,1\}^n \to \mathbb{R}$, each of complexity at most $S(n)$, the Average-of-Product of $\{C_i(x)\}_{i \in [4]}$ can be computed within an additive error of $S(n)^4/E(n)$ in $O\left(S(n)^4 \cdot 2^n/E(n)\right)$ time.*

Proof. Let $C_i = \sum_{j=1}^{m_i} \alpha_{i,j} \cdot C_{i,j}$. From the assumption, it follows that each $C_{i,j}$ is of at most $S(n)$ size, $m_i \leq S(n)$, and $\sum_{j=1}^{m_i} |\alpha_{i,j}| \leq S(n)$. By adding some dummy coefficients and dummy circuits, we can assume without loss of generality that all the m_i are equal to $m \leq S(n)$.

For each tuple $(j_1, j_2, j_3, j_4) \in [m]^4$, we run the promised CAPP algorithm $\mathcal{A}_{\mathsf{CAPP}}$ on $\bigwedge_{i \in [4]} C_{i,j_i}$ to output an estimate $\mathcal{A}_{\mathsf{CAPP}}(\bigwedge_{i \in [4]} C_{i,j_i})$ such that

$$\left| \mathcal{A}_{\mathsf{CAPP}}\left(\bigwedge_{i \in [4]} C_{i,j_i} \right) - \mathop{\mathbb{E}}_{x}\left[\prod_{i=1}^{4} C_{i,j_i}(x) \right] \right| \leq 1/E(n).$$

Hence, we can estimate $\mathbb{E}_x[\prod_{i=1}^4 C_i(x)]$ by computing the quantity

$$\sum_{(j_1,j_2,j_3,j_4)\in[m]^4} \mathcal{A}_{\mathsf{CAPP}}\left(\bigwedge_{i\in[4]} C_{i,j_i}\right)\cdot\prod_{i=1}^4 \alpha_{i,j_i}. \tag{7.7}$$

The error can be bounded by

$$\begin{aligned}
&\left|\sum_{(j_1,j_2,j_3,j_4)\in[m]^4} \mathcal{A}_{\mathsf{CAPP}}\left(\bigwedge_{i\in[4]} C_{i,j_i}\right)\cdot\prod_{i=1}^4 \alpha_{i,j_i} - \sum_{(j_1,j_2,j_3,j_4)\in[m]^4} \mathop{\mathbb{E}}_x\left[\prod_{i=1}^4 C_{i,j_i}(x)\right]\cdot\prod_{i=1}^4 \alpha_{i,j_i}\right| \\
&\le \sum_{(j_1,j_2,j_3,j_4)\in[m]^4}\prod_{i=1}^4 |\alpha_{i,j_i}|\cdot E(n)^{-1} \\
&\le E(n)^{-1}\cdot\prod_{i=1}^4\left(\sum_{j=1}^m |\alpha_{i,j}|\right) \\
&\le S(n)^4/E(n).
\end{aligned}$$

Computing (7.7) can be done in $O(2^n/E(n)\cdot m^4)\le O(2^n/E(n)\cdot S(n)^4)$ time, which completes the proof. ■

7.2.2 Notation, and the PCPP Reduction

We need the following lemma.

Lemma 7.2.3 ([CW19, VW20]). *There are constants $0 < s_{\mathsf{pcpp}} < c_{\mathsf{pcpp}} < 1$ and a polynomial-time transformation that, given a circuit D on ℓ inputs of size $s \ge \ell$, outputs a 2-SAT instance F on the variable set $\mathcal{Y}\cup\mathcal{Z}$ where $|\mathcal{Y}|\le \mathrm{poly}(\ell), |\mathcal{Z}|\le \mathrm{poly}(s)$, and the following hold for all $x\in\{0,1\}^\ell$:*

- *If $D(x)=1$, then $F|_{\mathcal{Y}=\mathsf{Enc}(x)}$ on variable set $\mathcal{Z}$ has a satisfying assignment $\mathcal{Z}_x$ such that at least c_{pcpp}-fraction of the clauses are satisfied. Furthermore, there is a $\mathrm{poly}(s)$ time algorithm that given x outputs $\mathcal{Z}_x$.*
- *If $D(x)=0$, then there is no assignment to the $\mathcal{Z}$ variables in $F|_{\mathcal{Y}=\mathsf{Enc}(x)}$ satisfies more than s_{pcpp}-fraction of the clauses.*

Moreover, the number of clauses in the 2-SAT instance F is a power of 2, and for each $i\in[|\mathcal{Y}|]$, $\mathsf{Enc}_i(x)$ is a parity function depending on at most $\ell/2$ bits of x.

Let D be an ℓ-input circuit of size s. The PCPP reduction of Lemma 7.2.3 applying to D gives us a 2-SAT instance over variables $\mathcal{Y}\cup\mathcal{Z}$ with $m\le\mathrm{poly}(s)$ clauses such that m is a power of 2.

Let $\{\mathsf{Cons}_i\}_{i=1}^m$ be the set of clauses[2], where each clause is an OR of two variables in $\mathcal{Y} \cup \mathcal{Z}$ or their negations. For $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, we use $\mathcal{Y}_s$ and $\mathcal{Z}_t$ to denote the s-th variable in $\mathcal{Y}$ and the t-th variable in $\mathcal{Z}$, respectively.

To help presentation in this chapter, we introduce some useful notation. For each clause Cons_i, it extends to a degree-2 polynomial, denoted as $\widetilde{\mathsf{Cons}_i}$.[3]

1. By a "**real-valued proof**" we mean a pair of two lists of proof functions (Y, Z) for PCPP, where $Y = (Y_s)_{s\in[|\mathcal{Y}|]}$, $Z = (Z_t)_{t\in[|\mathcal{Z}|]}$ and each Y_s and Z_t is a function from $\{0,1\}^\ell \to \mathbb{R}$. Based on (Y, Z), we define the following terminologies:
   - Recall each clause Cons_i involves two variables. We define indicators $T_{i1}^{(Y,Z)}$ and $T_{i2}^{(Y,Z)}$ to indicate the corresponding functions in (Y, Z). We also let $\mathsf{From}(T_{ij}^{(Y,Z)}) \in \mathcal{Y} \cup \mathcal{Z}$ be the variable it corresponds to.
   - Each clause Cons_i extends to a polynomial $\widetilde{\mathsf{Cons}_i}$, we define $F_i^{(Y,Z)} := \widetilde{\mathsf{Cons}_i}(T_{i1}^{(Y,Z)}, T_{i2}^{(Y,Z)})$.

   Note that these objects all depend on the given proof (Y, Z), when the context is clear, we also omit the superscript, and simply write them as T_{ij} and F_i.
2. By a "**Boolean-valued proof**" we mean a pair of two lists of proof functions $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ where $\widehat{Y}_s(x) = \mathsf{Enc}_s(x)$ for every $x \in \{0,1\}^\ell$ and $s \in [|\mathcal{Y}|]$, $\widehat{Z} = (\widehat{Z}_t)_{t\in[|\mathcal{Z}|]}$, and each $\widehat{Z}_t$ is function from $\{0,1\}^\ell \to \{0,1\}$. Recall that $\mathsf{Enc}\colon \{0,1\}^\ell \to \{0,1\}^{|\mathcal{Y}|}$ is the fixed $\mathbb{F}_2$-linear error correcting code used in Lemma 7.2.3. Similar to the case of real-valued proofs, the proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ induces $\widehat{T}_{ij}^{(\widehat{Y},\widehat{Z})}$ and $\widehat{F}_i^{(\widehat{Y},\widehat{Z})}$. When the context is clear, we omit the superscript and write them as $\widehat{T}_{ij}$ and $\widehat{F}_i$.

To clarify, we always use (Y, Z) to denote a real-valued proof, and $(\widehat{Y}, \widehat{Z})$ to denote a Boolean-valued proof. We also stress here that $Y_s, \widehat{Y}_s, Z_t, \widehat{Z}_t, T_{i,j}, \widehat{T}_{i,j}, F_i, \widehat{F}_i$ are all functions mapping from $\{0,1\}^\ell$.

7.3 Strongly Average-case Witness Lower Bounds and Almost-everywhere Lower Bounds

To prove Theorem 1.5.2 (restated below), we will follow the proof of Theorem 1.5.1 by first defining a nondeterministic algorithm APCPP that tries to speed up a language $L \in \mathsf{NTIME}[T]$, using

[2] Cons_i is also called "constraints", we use "clauses" and "constraints" interchangeably.

[3] We use the natural arithmetization: The Boolean 0 (false) and 1 (true) correspond to real 0 and 1, respectively. Boolean AND corresponds to real multiplication. Boolean OR corresponds to the real polynomial $\mathsf{OR}(a, b) = 1 - (1 - a) \cdot (1 - b)$.

both PCP and PCPP; see Algorithm 7.1. APCPP is considerably more complicated than APCP (see Algorithm 4.1), with two subroutines Validity-Test and Estimate that will be described later. We will make frequent use of notation from Section 7.2.2.

Algorithm 7.1: The algorithm APCPP attempting to speed up L

Setting: Let $T(n)$ be a time-constructive function, and $L \in \mathsf{NTIME}[T(n)]$. Let $\ell(n) = \log T + O(\log\log T)$ be the number of random bits from Lemma 4.4.1 when the running time is set to $T(n)$. Let $\delta_{\mathsf{v}} = \frac{(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}})}{10^6}$.

Parameters: A constant $\varepsilon \in (0,1)$.

Assumption: $\widetilde{\mathsf{CAPP}}$ for 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be solved in 2^{n-n^ε} time.

Input: $z \in \{0,1\}^n$

1 Apply Lemma 4.4.1 to L to obtain a $\mathrm{poly}(\ell)$-size AC^0_2 oracle circuit $\mathsf{VPCP}_z \colon \{0,1\}^\ell \to \{0,1\}$ that queries an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$;

2 **Guess** a $2^{\ell^{\varepsilon/4}}$-size ℓ-input circuit C and set $D = \mathsf{VPCP}^C_z$;

3 Apply Lemma 7.2.3 to the circuit D to obtain a 2-SAT instance over variables $\mathcal{Y} \cup \mathcal{Z}$ with $m \le \mathrm{poly}(\mathsf{SIZE}(D))$ clauses $\{\mathsf{Cons}_i\}_{i\in[m]}$ such that m is a power of 2;

4 **Guess** two lists of proof circuits $(Y,Z) = \left((Y_i)_{i\in[|\mathcal{Y}|]}, (Z_j)_{j\in[|\mathcal{Z}|]}\right)$ such that $Y_i, Z_j \in (\mathsf{Sum} \circ \mathscr{C})[2^{\ell^{\varepsilon/2}}]$ for every $i \in [|\mathcal{Y}|]$ and $j \in [|\mathcal{Z}|]$.

5 **if** $\mathsf{Validity\text{-}Test}(\{\mathsf{Cons}_i\}_{i\in[m]}, Y, Z) =$ **False then reject**;

6 **if** $\mathsf{Estimate}(\{\mathsf{Cons}_i\}_{i\in[m]}, Y, Z) \ge \frac{1}{2}\cdot(c_{\mathsf{pcpp}} + s_{\mathsf{pcpp}})$ **then accept**;

7 **else reject**;

Lemma 7.3.1. *Let $T, L, \ell, \varepsilon, \mathsf{VPCP}_z, \delta_{\mathsf{v}}, (Y_i)_{i\in[|\mathcal{Y}|]}, (Z_j)_{j\in[|\mathcal{Z}|]}$ be stated as in Algorithm 7.1, and $c_{\mathsf{pcpp}}, s_{\mathsf{pcpp}}$ be stated in Lemma 7.2.3. Under the assumption from Algorithm 7.1, the following holds:*

1. *APCPP runs in $\mathrm{poly}(n, \log T(n)) + o(T(n))$ time and guesses $2^{O(\ell^{\varepsilon/2})}$ bits.*
2. *For every sufficiently long $z \in \{0,1\}^*$ such that $L(z) = 0$, it holds that $\mathsf{APCPP}(z) = 0$.*[4]
3. *For every sufficiently long $z \in \{0,1\}^*$ such that $L(z) = 1$ and $\mathsf{APCPP}(z) = 0$, there exists $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ such that*

$$\Pr_{r\in_R\{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}_z(r) = 1] = 1.$$

[4] Formally, we mean that there is a universal constant $N_0 \in \mathbb{N}_{\ge 1}$, such that for every $z \in \{0,1\}^*$ with $|z| \ge N_0$, our statement holds.

For every $2^{\ell^{\varepsilon/4}}$-size circuit C such that $\Pr_{r\in_R\{0,1\}^\ell}[\mathsf{VPCP}_z^C(r)=1]=1$, letting $D=\mathsf{VPCP}_z^C$, the following holds:

(a) There exists a Boolean proof $(\widehat{Y}=\mathsf{Enc}(x),\widehat{Z})$ such that[5]

$$\mathop{\mathbb{E}}_{i\in[m]}\mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell}\widehat{F}_i(x)\geq c_{\mathsf{pcpp}}.$$

(b) For every Boolean proof $(\widehat{Y}=\mathsf{Enc}(x),\widehat{Z})$ satisfying the above, letting $f\colon\{0,1\}^{\log m+1+\ell}\to\{0,1\}$

$$f(i,j,u):=\widehat{T}_{ij}(u)\ \text{ for }(i,j,u)\in[m]\times\{0,1\}\times\{0,1\}^\ell,$$

it holds that f is δ_{v}-far in ℓ_1-distance from every $[0,1]\mathsf{Sum}\circ\mathscr{C}[2^{\ell^{\varepsilon/4}}]$ circuit.[6]

Before proving Lemma 7.3.1, we first show it implies strongly average-case witness lower bounds (Theorem 1.5.5) and strongly average-case almost-everywhere lower bounds (Theorem 1.5.2) together with the new XOR Lemma and the refuters.

Reminder of Theorem 1.5.2. *Let $\mathscr{C}$ be a typical and concrete circuit class. Suppose there is an $\varepsilon\in(0,1)$ such that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4\circ\mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a language $L\in\mathsf{E}^{\mathsf{NP}}$ and a constant $\delta\in(0,1)$ such that, for every sufficiently large $n\in\mathbb{N}_{\geq 1}$, L_n cannot be $(1/2+2^{-n^\delta})$-approximated by $\mathscr{C}$ circuits of size 2^{n^δ}.*

Proof. Let $k=1/\varepsilon$ and $T(n)=2^{\log^k n}$. Let L and $\mathcal{R}$ be the $\mathsf{NTIME}[T]$ language and the corresponding refuter from Theorem 1.5.9. Now we consider APCPP (described in Algorithm 7.1) with T,L and parameter ε. Note that the assumption of Algorithm 7.1 is exactly the assumption of the theorem and let $\ell,\delta_{\mathsf{v}},\mathsf{VPCP}_z$ be as stated in Algorithm 7.1.

By Item (1) of Lemma 7.3.1, APCPP is an $\mathsf{NTIMEGUESS}[o(T(n)),n/10]$ algorithm. Hence, the refuter $\mathcal{R}$ can be applied to find a differing an input $z\in\{0,1\}^n$ such that $\mathsf{APCPP}(z)\neq L(z)$ (by Item (2) of Lemma 7.3.1, it must be the case that $L(z)=1$ and $\mathsf{APCPP}(z)=0$), for every sufficiently large $n\in\mathbb{N}_{\geq 1}$.

We are now ready to design our new hard language $\mathcal{A}_{\mathsf{avgHARD}}$. On an input y of length τ such that τ is sufficiently large, let $n=2^{\tau^{1/3k}}$. Applying the refuter $\mathcal{R}^T$, we can find in $\mathrm{poly}(T(n))$ time with a SAT oracle a string $z\in\{0,1\}^n$ such that $L(z)=1$ while $\mathsf{APCPP}(z)=0$.

[5]These Boolean proofs correspond to the 2-SAT instance constructed by applying Lemma 7.2.3 to the circuit D, as in Line 3 in Algorithm 7.1.

[6]Here we identify $[m]$ with the set $\{0,1\}^{\log m}$ (note that $\log m\in\mathbb{N}_{\geq 1}$).

Now, we consider the execution of APCPP on z. Depending on whether there is a circuit C of $2^{\ell^{\varepsilon/4}}$ size such that $\Pr_{x\in_R\{0,1\}^\ell}[\mathsf{VPCP}_z^C(x)=1]=1$ (this can be checked with a call to a SAT oracle in $2^{O(\ell)}$ time), we consider the following two cases.

Case 1. There is no circuit C of $2^{\ell^{\varepsilon/4}}$ size such that $\Pr_{x\in_R\{0,1\}^\ell}[\mathsf{VPCP}_z^C(x)=1]=1$. In this case, we simply find the truth table of the lexicographically first function $f\colon\{0,1\}^\ell\to\{0,1\}$ (with a SAT oracle) such that $\mathsf{VPCP}^f(x)$ is a tautology. Such an f exists by Item (3) of Lemma 7.3.1. Note that f does not have $2^{\ell^{\varepsilon/4}}$-size circuits, from the assumption of Case 1.

By Theorem 3.3.1, in $2^{O(\ell)}$ time, we can construct from f a new function $f_{\mathsf{amp}}\colon\{0,1\}^{O(\ell)}\to\{0,1\}$, which cannot be $(\frac{1}{2}+2^{-\ell^{\Omega(1)}})$-approximated by circuits of size $2^{\ell^{\Omega(1)}}$.

Case 2. There is a circuit of size $2^{\ell^{\varepsilon/4}}$ such that $\Pr_{x\in_R\{0,1\}^\ell}[\mathsf{VPCP}_z^C(x)=1]=1$. Then given access to a SAT oracle, in $2^{O(\ell)}$ time we can find the lexicographically first such circuit C and the first Boolean proof $(\widehat{Y}=\mathsf{Enc}(x),\widehat{Z})$ such that

$$\mathop{\mathbb{E}}_{i\in[m]}\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell}\widehat{F}_i(x)\geq c_{\mathsf{pcpp}}.$$

The Boolean proof $(\widehat{Y}=\mathsf{Enc}(x),\widehat{Z})$ above exists by Item (3.a) of Lemma 7.3.1. Let $r=\log m\leq O(\ell^{\varepsilon/4})$ and $f\colon\{0,1\}^{r+1+\ell}\to\{0,1\}$ be defined as in Item (3.b) of Lemma 7.3.1. We know that f is δ_{v}-far (in ℓ_1 distance) from $[0,1]\mathsf{Sum}\circ\mathscr{C}[2^{\ell^{\varepsilon/4}}]$ circuits.

Using Lemma 1.5.10 and setting $d=\ell^{\varepsilon/5}$, we conclude that $f^{\oplus d}$ cannot be $\left(\frac{1}{2}+(1-\delta_{\mathsf{v}})^{d-1}\right)$-approximated by $\mathscr{C}$ circuits of $2^{\ell^{\varepsilon/4}-O_{\delta_{\mathsf{v}}}(d)}$ size. In other words, the function $f^{\oplus d}$, taking $O(\ell^{1+\varepsilon/5})$ bits of input, cannot be $\left(\frac{1}{2}+2^{-\ell^{\Omega(1)}}\right)$-approximated by $\mathscr{C}$ circuits of size $2^{\ell^{\Omega(1)}}$.

Construction of the hard language $\mathcal{A}_{\mathsf{avgHARD}}$. Recall that $\ell=\log T(n)+O(\log\log T(n))=\log^k n+O(\log\log n)$. Using a SAT oracle, we can decide whether there is a circuit C of $2^{\ell^{\varepsilon/4}}$ size such that VPCP_z^C is a tautology in $\mathrm{poly}(T(n))\leq 2^{O(\ell)}$ time. From the discussions above, in both cases we can construct an average-case hard function f on inputs of length $\ell'\leq O(\max(\ell,\ell^{1+\varepsilon/5}))\leq\ell^2$ in $2^{O(\ell)}$ time with a SAT oracle, such that f cannot be $(1/2+2^{-\ell^{\Omega(1)}})$-approximated by $2^{\ell^{\Omega(1)}}$-size $\mathscr{C}$ circuits.

Note that $\ell^2\leq O(\log^{2k}n)\leq\tau$ form our choice of $n=2^{\tau^{1/3k}}$. We can then pad the input length to τ in the natural way. That is, we define $g\colon\{0,1\}^\tau\to\{0,1\}$ such that $g(y)=f(y_{\leq\ell'})$ where $y_{\leq\ell'}$ is the first ℓ' bits of y. It follows that g cannot be $\left(1/2+2^{-m^{\Omega(1)}}\right)$-approximated by $\mathscr{C}$ circuits

of size $2^{\ell^{\Omega(1)}} = 2^{m^{\Omega(1)}}$. Finally, $\mathcal{A}_{\mathsf{avgHARD}}$ simply outputs $g(y)$ on the input y. Note that it runs in $2^{O(\ell)} \le 2^{O(m)}$ time with a SAT oracle, and this completes the proof. ■

Reminder of Theorem 1.5.5. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ such that unary NE does not admit $(1/2 + 2^{-n^\delta})$-approximate 2^{n^δ}-size $\mathscr{C}$ witnesses.*

Proof. Let $T(n) = 2^n$ and L be a unary language such that $L \in \mathsf{NTIME}[T(n)] \setminus \mathsf{NTIME}[T(n)/n]$, whose existence is guaranteed by the non-deterministic time hierarchy theorem [Žák83]. Now we consider APCPP with T, L and parameter ε. Note that the assumption of Algorithm 7.1 is exactly the assumption of the theorem and let $\ell, \delta_{\mathsf{v}}, \mathsf{VPCP}_z$ be as stated in Algorithm 7.1. Note that $\ell(n) = n + O(\log n)$.

By Item (1) of Lemma 7.3.1, APCPP is an $\mathsf{NTIME}[o(T(n))]$ algorithm. Hence, there are infinitely many $n \in \mathbb{N}_{\ge 1}$ such that $L(1^n) = 1$ and $\mathsf{APCPP}(1^n) = 0$. Let $\mathcal{S}$ be the set of all such integers n. Depending on whether there is a succinct circuit for VPCP_{1^n} as a correct oracle (meaning that $\mathsf{VPCP}^C_{1^n}(r) = 1$ for all r), we will use different constructions. Formally, we let

$$\mathcal{S}_1 = \{\text{there is no circuit } C \text{ of size } 2^{\ell^{\varepsilon/4}} \text{ such that } \mathsf{VPCP}^C_{1^n}(x) \text{ is a tautology} : n \in \mathcal{S}\} \quad \text{and} \quad \mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1.$$

Since $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, at least one of $\mathcal{S}_1$ and $\mathcal{S}_2$ must be an infinite set. We will discuss each case separately.

Case 1: $|\mathcal{S}_1|$ is infinite. In the following we only consider $n \in \mathcal{S}_1$. Our verifier $V(x, y)$ works as follows:

1. V first checks if $x = 1^{2n^2}$ for some $n \in \mathbb{N}_{\ge 1}$, and rejects immediately otherwise. Let $\tau = 2n^2$. Let $f_y \colon \{0,1\}^\tau \to \{0,1\}$ be the function $\mathsf{func}(y)$. V first treats the truth-table of $f_y(0^{\tau/2}, \cdot)$ (which has length $2^{\tau/2} \gg 2^{\ell(n)}$) as the description of a function $g \colon \{0,1\}^\ell \to \{0,1\}$ and verifies that $\mathsf{VPCP}^g_{1^n}(x)$ is a tautology (thus, g does not have a $2^{\ell(n)^{\varepsilon/4}}$-size circuit; it rejects immediately if it is not the case).
2. Then it applies Theorem 3.3.1 to obtain $\mu = O(n)$ and a hard truth-table $\mathsf{Amp}^g \colon \{0,1\}^\mu \to \{0,1\}$ such that Amp^g cannot be $(1/2 + 2^{-\Omega(\ell(n)^{\varepsilon/4})})$-approximated by $2^{\Omega(\ell(n)^{\varepsilon/4})}$-size circuits. V then verifies that for every $\alpha \in \{0,1\}^{\tau/2} \setminus \{0^{\tau/2}\}$ and $\beta \in \{0,1\}^{\tau/2}$, $f_y(\alpha, \beta) = \mathsf{Amp}^g(\beta_{\le \mu})$.

Case 2: $|\mathcal{S}_2|$ is infinite. In the following we only consider $n \in \mathcal{S}_2$. Our verifier $V(x, y)$ works as follows:

1. V first checks if $x = 1^{2n^2}$ for some $n \in \mathbb{N}_{\geq 1}$, and rejects immediately otherwise. Let $\tau = 2n^2$. Let $f_y \colon \{0,1\}^\tau \to \{0,1\}$ be the function $\mathsf{func}(y)$. V first treats the truth-table of $f_y(0^{\tau/2}, \cdot)$ (which has length $2^{\tau/2} \gg 2^{O(\ell(n))}$) as the description of a $2^{\ell(n)^{\varepsilon/4}}$-size circuit C and a Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$. It then verifies that $\mathsf{VPCP}^C_{1^n}(x)$ is a tautology, and

$$\mathop{\mathbb{E}}_{i \in_R [m]} \mathop{\mathbb{E}}_{x \in_R \{0,1\}^\ell} \widehat{F}_i(x) \geq c_{\mathsf{pcpp}}.$$

   Let $r = \log m \leq O(\ell^{\varepsilon/4})$ and $f \colon \{0,1\}^{r+1+\ell} \to \{0,1\}$ be defined as in Item (3.b) of Lemma 7.3.1. We know that f is δ_{v}-far (in ℓ_1 distance) from $[0,1]\mathsf{Sum} \circ \mathscr{C}[2^{\ell^{\varepsilon/4}}]$ circuits.
2. Using Lemma 1.5.10 and setting $d = \ell^{\varepsilon/5}$, we conclude that $f^{\oplus d}$ cannot be $\left(\frac{1}{2} + (1-\delta_{\mathsf{v}})^{d-1}\right)$-approximated by $\mathscr{C}$ circuits of $2^{\ell^{\varepsilon/4} - O_{\delta_{\mathsf{v}}}(d)}$ size. In other words, the function $f^{\oplus d}$, taking $\mu = O(\ell^{1+\varepsilon/5})$ bits of input, cannot be $\left(\frac{1}{2} + 2^{-\ell^{\Omega(1)}}\right)$-approximated by $\mathscr{C}$ circuits of size $2^{\ell^{\Omega(1)}}$. V then verifies that for every $\alpha \in \{0,1\}^{\tau/2} \setminus \{0^{\tau/2}\}$ and $\beta \in \{0,1\}^{\tau/2}$, $f_y(\alpha, \beta) = f^{\oplus d}(\beta_{\leq \mu})$.

In both cases, by the construction (and a very similar argument as in the proof of Theorem 1.5.2), for infinitely many $n \in \mathbb{N}_{\geq 1}$, letting $\tau = 2n^2$, it follows that $V(1^\tau, y)$ accepts some $y \in \{0,1\}^{2^\tau}$, and the corresponding f_y for every accepted $y \in \{0,1\}^{2^\tau}$ cannot be $\left(1/2 + 2^{-\tau^{\Omega(1)}}\right)$-approximated by $2^{\tau^{\Omega(1)}}$-size $\mathscr{C}$ circuits. This completes the proof.

■

7.3.1 Completeness and Soundness for PCP and PCPP

Now we start analyzing APCPP, we start by proving some useful claims and eventually use them to prove Lemma 7.3.1. We first note that from Lemma 4.4.1 and Lemma 7.2.3, we have $\mathsf{SIZE}(D) \leq \mathrm{poly}(\ell) \cdot 2^{\ell^{\varepsilon/4}}$ and

$$m \leq \mathrm{poly}(\mathsf{SIZE}(D)) \leq 2^{O(\ell^{\varepsilon/4})}. \tag{7.8}$$

In Line 1, APCPP applies the PCP from Lemma 4.4.1, and the following claim follows directly from Lemma 4.4.1.

Claim 7.3.2. *The following statements hold.*

1. *If $L(z) = 1$, then there is $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{x \in_R \{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}_z(x) = 1] = 1$.*
2. *If $L(z) = 0$, then for every $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$, it holds that $\Pr_{x \in_R \{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}_z(x) = 1] \leq \frac{1}{n^{10}}$.*

In Line 3, APCPP attempts to distinguish the two cases in Claim 7.3.2 by further applying the PCPP reduction of Lemma 7.2.3. From Lemma 7.2.3, we have the following claim.

Claim 7.3.3. *For every $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$, the following statements hold.*

1. *If $\Pr_{x\in_R\{0,1\}^\ell}[\mathsf{VPCP}_z^{\mathcal{O}}(x)=1]$, then there is a Boolean proof $(\widehat{Y}=\mathsf{Enc}(x),\widehat{Z})$ such that*

$$\mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell}\mathop{\mathbb{E}}_{i\in_R[m]}\widehat{F}_i(x)\geq c_{\mathsf{pcpp}}.$$

2. *If $\Pr_{x\in_R\{0,1\}^\ell}[\mathsf{VPCP}_z^{\mathcal{O}}(x)=1]\leq 1/n^{10}$, then for every sufficiently large n, for every Boolean proof $(\widehat{Y}=\mathsf{Enc}(x),\widehat{Z})$, we have*

$$\mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell}\mathop{\mathbb{E}}_{i\in_R[m]}\widehat{F}_i(x)<c_{\mathsf{pcpp}}-\frac{9}{10}(c_{\mathsf{pcpp}}-s_{\mathsf{pcpp}}).$$

7.3.2 Guessing Succinct $\mathsf{Sum}\circ\mathscr{C}$ Circuits and the Validity Test

Next, in Line 4, APCPP guesses two lists of proof circuits $(Y,Z)=\left((Y_i)_{i\in[|\mathcal{Y}|]},(Z_j)_{j\in[|\mathcal{Z}|]}\right)$ such that $Y_i,Z_j\in(\mathsf{Sum}\circ\mathscr{C})[2^{\ell^{\varepsilon/2}}]$ for every $i\in[|\mathcal{Y}|]$ and $j\in[|\mathcal{Z}|]$.

Note that ideally we only want to consider $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits, but it is not clear how to verify that a given $\mathsf{Sum}\circ\mathscr{C}$ circuit satisfies the $[0,1]\mathsf{Sum}\circ\mathscr{C}$ promise. Therefore, at Line 5, APCPP applies a certain validity test on the guessed proof (Y,Z), and reject immediately if the test fails. Although passing the test does not guarantee all members of Y,Z are $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits, it does mean they are "close enough" to $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits so that our analysis still goes through. The validity test $\mathsf{Validity\text{-}Test}(Y,Z)$ will be described later.

Since $F_i(x):=\widetilde{\mathsf{Cons}}(T_{i1}(x),T_{i2}(x))$ is a degree-2 polynomial, evaluating $\mathbb{E}_x[F(x)]$ reduces to an Average-of-Product problem over (T_{i1},T_{i2}). By Lemma 7.2.2 and the assumed CAPP algorithm for 2^{ℓ^ε}-size $\mathsf{AND}_4\circ\mathscr{C}$ circuits, we can estimate $\mathbb{E}_x[F_i(x)]$ within an additive error of $2^{O(\ell^{\varepsilon/2})-\Omega(\ell^\varepsilon)}\leq o(1)$ in $2^{\ell-\ell^\varepsilon+O(\ell^{\varepsilon/2})}$ time. Let $\widetilde{\mathbb{E}}_x(F_i)$ be the output of the estimation algorithm when evaluating $\mathbb{E}_x[F_i(x)]$.

Now we can describe the Estimate subroutine below (the assumption of Algorithm 7.2 is satisfied from the discussions above).

Now we know that conditioning on the $\mathsf{Validity\text{-}Test}(Y,Z)$ is passed, APCPP accepts if and only if

$$\mathop{\mathbb{E}}_{i\in_R[m]}\widetilde{\mathbb{E}}_x(F_i)\geq c_{\mathsf{pcpp}}-\frac{5}{10}\left(c_{\mathsf{pcpp}}-s_{\mathsf{pcpp}}\right). \tag{7.9}$$

Algorithm 7.2: The subroutine Estimate

Input: Constraints $\{\mathsf{Cons}_i\}_{i\in[m]}$ and a real-valued proof $(Y,Z) = \Big((Y_i)_{i\in[|\mathcal{Y}|]}, (Z_j)_{j\in[|\mathcal{Z}|]}\Big)$.

Setting: Let $T_{i,j}$ and F_i be defined as in Section 7.2.2 from $\{\mathsf{Cons}_i\}_{i\in[m]}$ and (Y,Z).

Assumption: There is an algorithm estimating $\mathbb{E}_x[F_i(x)]$ within an additive error of $o(1)$ in $2^{\ell-\ell^\varepsilon+O(\ell^{\varepsilon/2})}$ time. Let $\widetilde{\mathbb{E}}_x(F_i)$ be the output of the estimation algorithm when evaluating $\mathbb{E}_x[F_i(x)]$.

1 **return** $\mathbb{E}_{i\in_R[m]}\widetilde{\mathbb{E}}_x(F_i)$;

The running time of Estimate can also be easily bounded as follows.

Claim 7.3.4. Estimate *runs in* $2^{\ell-\Omega(\ell^\varepsilon)} \le o(T(n))$ *time.*

Proof. From the assumption of Algorithm 7.2 and (7.8), the running time of Estimate is at most

$$m\cdot 2^{\ell-\ell^\varepsilon+O(\ell^{\varepsilon/2})} = 2^{\ell-\ell^\varepsilon+O(\ell^{\varepsilon/2})+O(\ell^{1/4})} = 2^{\ell-\Omega(\ell^\varepsilon)} \le o(T(n)). \qquad \blacksquare$$

This completes the description of APCPP except for the validity test, which is described next.

Validity test on guessed $\mathsf{Sum}\circ\mathscr{C}$ **circuits.** For every $T_{ij}(x)$, consider the function

$$P_{ij}(x) = \begin{cases} T_{ij}(x)^2(1-T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij})\in\mathcal{Z}, \\ (\mathsf{Enc}_s(x)-T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij}) = \mathcal{Y}_s \text{ for } s\in[|\mathcal{Y}|], \end{cases} \tag{7.10}$$

and

$$Q_{ij}(x) = T_{ij}(x)^2. \tag{7.11}$$

We want to estimate the expectations

$$\mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell} P_{ij}(x) \text{ and } \mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell} Q_{ij}(x) \text{ for each } i,j\in[m]\times[2].$$

It is clear that for every $(i,j)\in[m]\times[2]$, $Q_{ij}(x)$ is a polynomial over $T_{ij}(x)$ of degree 4. For $P_{ij}(x)$, it is also a degree-4 polynomial over $T_{ij}(x)$ when $\mathsf{From}(T_{ij})\in\mathcal{Z}$. Hence, in these two cases, the evaluation of these expectations reduces to computing Average-of-Product problems for the $T_{i,j}$, which can in turn be estimated by Lemma 7.2.2. When $\mathsf{From}(T_{ij}) = \mathcal{Y}_s$, $\mathsf{Enc}_s(x)$ depends on at most $\frac{\ell}{2}$ bits (the moreover part of Lemma 7.2.3). We can then enumerate all these bits, and solve the

Average-of-Product problem on the remaining part of inputs to estimate $\mathbb{E}_{x\in_R\{0,1\}^\ell} P_{ij}(x)$.[7] Hence, the whole estimation procedure runs in time

$$2^{O(\ell^{\varepsilon/4})} \cdot \left(2^{\ell-\ell^{\varepsilon}+O(\ell^{\varepsilon/2})} + 2^{\ell/2} \cdot 2^{(\ell/2)-(\ell/2)^{\varepsilon}+O(\ell^{\varepsilon/2})}\right) = o(T(n)).$$

Now we are ready to described our validity test Validity-Test (the assumption of Algorithm 7.3 follows from the above discussions).

Algorithm 7.3: The subroutine Validity-Test

Input: Constraints $\{\mathsf{Cons}_i\}_{i\in[m]}$ and a real-valued proof $(Y,Z) = \left((Y_i)_{i\in[|\mathcal{Y}|]}, (Z_j)_{j\in[|\mathcal{Z}|]}\right)$.

Setting: Let $T_{i,j}$ and F_i be defined as in Section 7.2.2 from $\{\mathsf{Cons}_i\}_{i\in[m]}$ and (Y,Z). Let $P_{i,j}$ and $Q_{i,j}$ be defined by (7.10) and (7.11). Let $\delta_{\mathsf{v}} = \frac{(s_{\mathsf{pcpp}}-c_{\mathsf{pcpp}})^2}{10^6}$.

Assumption: There is an algorithm estimating $\mathbb{E}_x(P_{ij})$ and $\mathbb{E}_x(Q_{ij})$ within an additive error of $o(1)$ for every $(i,j)\in[m]\times[2]$ in $o(T(n))$ time. Let $\widetilde{\mathbb{E}}_x(P_{i,j})$ and $\widetilde{\mathbb{E}}_x(Q_{i,j})$ be the corresponding estimation output.

1 **if** $\mathbb{E}_{i,j\in_R[m]\times[2]} \widetilde{\mathbb{E}}_x(P_{ij}) > 2\delta_{\mathsf{v}}$ **then return False**;

2 **if** $\widetilde{\mathbb{E}}_x(Q_{ij}) > 1+\delta_{\mathsf{v}}$ *for some* $(i,j)\in[m]\times[2]$ **then return False**;

3 **return True**;

In other words, the proof (Y,Z) passes the test if and only if both of the following conditions hold:

1. $\mathbb{E}_{i,j\in_R[m]\times[2]} \widetilde{\mathbb{E}}_x(P_{ij}) \le 2\delta_{\mathsf{v}}$.
2. $\widetilde{\mathbb{E}}_x(Q_{ij}) \le 1+\delta_{\mathsf{v}}$ for every $(i,j)\in[m]\times[2]$.

The following claim follows directly from the assumption of Algorithm 7.3.

Claim 7.3.5. Validity-Test *runs in* $2^{\ell-\Omega(\ell^{\varepsilon})} \le o(T(n))$ *time.*

Completeness and soundness of the validity test. Recall that in Section 7.2.2 we have defined $F_i(x) := \widetilde{\mathsf{Cons}_i}(T_{i1}(x), T_{i2}(x))$, where $\widetilde{\mathsf{Cons}_i}$ is the polynomial extension of Cons_i. For a Boolean-valued proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, we also defined $\widehat{F}_i(x) := \widetilde{\mathsf{Cons}_i}(\widehat{T}_{i1}(x), \widehat{T}_{i2}(x))$. The following lemma summarizes the properties we need from the validity test. We defer its proof to the end of this section.

Lemma 7.3.6. *Let* δ_{v} *be stated as in Algorithm 7.3. We have the following completeness and soundness conditions for* Validity-Test.

[7] After fixing all the bits that $\mathsf{Enc}_s(x)$ depends on, $\mathsf{Enc}_s(x)$ can be replaced by a constant, so $P_{ij}(x)$ becomes a degree-4 polynomial over T_{ij}.

1. ***Completeness.*** *Every (Y, Z) satisfying the following conditions passes* Validity-Test*:*
   (1.a) For every $(s,t) \in [|\mathcal{Y}|] \times [|\mathcal{Z}|]$ and every input $x \in \{0,1\}^n$, $Y_s(x), Z_t(x) \in [0,1]$.
   (1.b) There is a Boolean-valued proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z}(x))$ such that

$$\mathop{\mathbb{E}}_{(i,j)\in_R[m]\times[2]} \left\| T_{ij} - \widehat{T}_{ij} \right\|_1 \leq \delta_{\mathsf{v}}.$$

2. ***Soundness.*** *If (Y, Z) passes* Validity-Test*, the following statements hold.*
   (2.a) There is a Boolean-valued proof $(\widehat{Y}(x) = \mathsf{Enc}(x), \widehat{Z}(x))$ such that

$$\mathop{\mathbb{E}}_{(i,j)\in_R[m]\times[2]} \left\| T_{ij} - \widehat{T}_{ij} \right\|_2 \leq \sqrt{12\delta_{\mathsf{v}}}.$$

   (2.b) For any Boolean-valued proof $(\widehat{Y}(x) = \mathsf{Enc}(x), \widehat{Z}(x))$, for every $i \in [m]$, it holds that

$$\left\| F_i - \widehat{F}_i \right\|_1 \leq 6 \cdot \mathop{\mathbb{E}}_{j\in_R[2]} \left\| T_{ij} - \widehat{T}_{ij} \right\|_2.$$

Intuitively speaking, the condition (2.a) above says that there is a Boolean-valued proof $(\widehat{Y}, \widehat{Z})$ that is close to (Y, Z), and (2.b) says that the closeness in (1.a) can be used to bound the difference between F_i and $\widehat{F}_i$.

7.3.3 Proof of Lemma 7.3.1

Now we are ready to prove Lemma 7.3.1. We begin by Item (1) of Lemma 7.3.1.

The running time and witness length of APCPP**.** We first bound the running time and the amount of nondeterminism of APCPP.

Proof of Item (1) of Lemma 7.3.1. Note that the running time of APCPP consists of the following: (1) the running time of applying Lemma 4.4.1 and Lemma 7.2.3, and (2) the running time of the subroutines Validity-Test and Estimate. Also recall that $m \leq 2^{O(\ell^{\varepsilon/4})}$ by (7.8).

By Lemma 4.4.1, Lemma 7.2.3, Claim 7.3.4, and Claim 7.3.5, the total running time can then be bounded by

$$\mathrm{poly}(n, \log T) + \mathrm{poly}(m) + o(T(n)) \leq \mathrm{poly}(n, \log T) + o(T(n)).$$

Also, from Algorithm 7.1, APCPP guesses a circuit C and two lists (Y, Z). The description of C consists of $2^{O(\ell^{\varepsilon/4})}$ bits. Note that $|\mathcal{Y}|$ and $|\mathcal{X}|$ are both bounded by $\mathrm{poly}(m)$, and each $(\mathsf{Sum} \circ$

$\mathscr{C})[2^{\ell^{\varepsilon/2}}]$ circuit takes $2^{O(\ell^{\varepsilon/2})}$ bits to describe. Hence (Y, Z) can be described by $2^{O(\ell^{\varepsilon/2})}$ bits. In total, the number of nondeterminism can be bounded by $2^{O(\ell^{\varepsilon/2})}$. ■

Proof of Item (2) of Lemma 7.3.1. Fix a sufficiently long input $z \in \{0,1\}^*$ and assume $L(z) = 0$. We wish to show that $\mathsf{APCPP}(z) = 0$.

Suppose that at Line 2 and Line 4, APCPP guessed a circuit C as the oracle for VPCP_z and (Y, Z) as its lists of proof circuits. By Item (2) of Claim 7.3.2, VPCP_z^C outputs 1 on at most a $\frac{1}{n^{10}}$ fraction of inputs. In the following we assume that (Y, Z) passes the validity test, as otherwise APCPP immediately rejects.

For every Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, by Item (2) of Claim 7.3.3, we have

$$\mathop{\mathbb{E}}_{i\in_R[m]} \mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell} \widehat{F}_i(x) < c_{\mathsf{pcpp}} - \frac{9}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).$$

By Item (2.a) of Lemma 7.3.6, for some Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, it holds that

$$\mathop{\mathbb{E}}_{(i,j)\in_R[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 \le \sqrt{12\delta_{\mathsf{v}}}.$$

By Item (2.b) of Lemma 7.3.6, it follows that

$$\begin{aligned}
\mathop{\mathbb{E}}_{i\in_R[m]} \mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell} F_i(x) &\le \mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell} \widehat{F}_i(x) + \mathop{\mathbb{E}}_{i\in_R[m]} \left\|\hat{F}_i - F_i\right\|_1 \\
&\le \mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell} \widehat{F}_i(x) + 6 \cdot \mathop{\mathbb{E}}_{i,j\in_R[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 \\
&< c_{\mathsf{pcpp}} - \frac{7}{10}\left(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}\right). \qquad (7.12)
\end{aligned}$$

The last inequality holds since $\delta_{\mathsf{v}} = \frac{(s_{\mathsf{pcpp}} - c_{\mathsf{pcpp}})^2}{10^6}$ (see Algorithm 7.1) and $6 \cdot \sqrt{12\delta_{\mathsf{v}}} \le \frac{1}{5} \cdot (c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}})$.

Recall that we use $\widetilde{\mathbb{E}}_x(F_i)$ to denote the output of the estimation algorithm on $\mathbb{E}_x[F_i(x)]$. Since T_{i1} and T_{i2} are $\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity $2^{O(\ell^{\varepsilon/2})}$, by Lemma 7.2.2 it follows that

$$\left|\mathop{\mathbb{E}}_{i\in_R[m]} \mathop{\mathbb{E}}_{x\in_R\{0,1\}^\ell} F_i(x) - \mathop{\mathbb{E}}_{i\in_R[m]} \widetilde{\mathbb{E}}_x(F_i)\right| \le o(1) \le \frac{1}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}). \qquad (7.13)$$

By (7.12) and (7.13), APCPP rejects on (Y, Z). Therefore, $\mathsf{APCPP}(z) = 0$ since it rejects every C and (Y, Z), and the conclusion follows. ■

We restate the statement of Item (3) of Lemma 7.3.1 here for reference.

Reminder of Item (3) of Lemma 7.3.1. *Let $T, L, \ell, \varepsilon, \mathsf{VPCP}_z, \delta_{\mathsf{v}}, (Y_i)_{i\in[|\mathcal{Y}|]}, (Z_j)_{j\in[|\mathcal{Z}|]}$ be stated as in Algorithm 4.1, and $c_{\mathsf{pcpp}}, s_{\mathsf{pcpp}}$ be stated in Lemma 7.2.3. Under the assumption from Algorithm 4.1, the following holds:*

3 *For every sufficiently long $z \in \{0,1\}^*$ such that $L(z) = 1$ and $\mathsf{APCPP}(z) = 0$, there exists $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$ such that*

$$\Pr_{r\in_{\mathsf{R}}\{0,1\}^\ell}[\mathsf{VPCP}_z^{\mathcal{O}}(r) = 1] = 1.$$

For every $2^{\ell^{\varepsilon/4}}$-size circuit C satisfying the above (as the oracle), letting $D = \mathsf{VPCP}_z^C$, the following holds:

(a) There exists a Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ such that[8]

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in_{\mathsf{R}}\{0,1\}^\ell} \widehat{F}_i(x) \geq c_{\mathsf{pcpp}}.$$

(b) For every Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ satisfying the above, letting $f\colon \{0,1\}^{\log m+1+\ell} \to \{0,1\}$

$$f(i,j,u) := \widehat{T}_{ij}(u) \text{ for } (i,j,u) \in [m]\times\{0,1\}\times\{0,1\}^\ell,$$

it holds that f is δ_{v}-far from every $[0,1]\mathsf{Sum}\circ\mathscr{C}[2^{\ell^{\varepsilon/4}}]$ circuit.[9]

Proof of Item (3) of Lemma 7.3.1. Let z, C, D be stated as in Item (3) of Lemma 7.3.1, and $r = \log m$. Recall that $D = \mathsf{VPCP}_z^C$ and $\Pr_{x\in_{\mathsf{R}}\{0,1\}^\ell}[D(x) = 1] = 1$.

From Item (1) of Claim 7.3.3, it follows that there is a Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ satisfying

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in_{\mathsf{R}}\{0,1\}^\ell} \widehat{F}_i(x) \geq c_{\mathsf{pcpp}}. \tag{7.14}$$

To establish Item (3), let $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ be any Boolean proof satisfying (7.14) and suppose there is a circuit $E\colon \{0,1\}^{r+1+\ell} \to \mathbb{R}$ from $[0,1]\mathsf{Sum}\circ\mathscr{C}[2^{\ell^{\varepsilon/4}}]$ such that $\|E - f\|_1 \leq \delta_{\mathsf{v}}$. We are going to construct a real-valued proof (Y, Z) that makes APCPP accept (together with the circuit C), contradicting to the assumption $\mathsf{APCPP}(z) = 0$ from Item (3) of Lemma 7.3.1.

[8] These Boolean proofs correspond to the 2-SAT instance constructed by applying Lemma 7.2.3 to the circuit D, as in Line 3 in Algorithm 7.1.

[9] Here we identify $[m]$ with the set $\{0,1\}^{\log m}$ (note that $\log m \in \mathbb{N}_{\geq 1}$).

Construction of the proof (Y, Z). Similar to f, we also think of E as a function on $\{0,1\}^r \times \{0,1\} \times \{0,1\}^\ell$, and often use i and j to denote the first two parts of input. Recall that we identify $[m]$ with $\{0,1\}^r$.

For $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, we define

$$Y_s(x) := \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=\mathcal{Y}_s} E(i,j,x) \text{ and } Z_t(x) := \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=\mathcal{Z}_t} D(i,j,x).$$

Since $m \leq 2^{O(\ell^{\varepsilon/4})}$ (see (7.8)) and for each fixed $(i,j) \in [m] \times [2]$, $E(i,j,\cdot)$ is a $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit of complexity at most $2^{\ell^{\varepsilon/4}}$. It is clear that for every $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, Y_s and Z_t are $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity at most $2^{\ell^{\varepsilon/2}}$.

We will show that $\mathsf{APCPP}(z)$ accepts the real-valued proof (Y, Z) defined above together with the circuit C as the guess at Line 2. Let T_{ij} and $\widehat{T}_{ij}$ be the indicators for (Y, Z) and $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, respectively (see Section 7.2.2).

(Y, Z) passes the validity test. We first apply Item (1) of Lemma 7.3.6 to prove that (Y, Z) passes the validity test (Validity-Test) in APCPP. Since Item (1.a) of Lemma 7.3.6 (all the Y_s and Z_t are $[0,1]$-valued) is already satisfied, it suffices to verify Item (1.b) of Lemma 7.3.6, which is

$$\mathop{\mathbb{E}}_{i,j \in_R [m] \times [2]} \|\widehat{T}_{i,j} - T_{i,j}\| \leq \delta_{\mathsf{v}}.$$

We have

$$\begin{aligned} \mathop{\mathbb{E}}_{i,j \in_R [m] \times [2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 &= \frac{1}{2m} \cdot \sum_{i,j \in [m] \times [2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \\ &= \frac{1}{2m} \cdot \sum_{X \in \mathcal{Y} \cup \mathcal{Z}} \sum_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1. \end{aligned} \tag{7.15}$$

To bound (7.15), we need the following simple fact.

Fact 7.3.7. *For every $(v_1, v_2, \ldots, v_d) \in [0,1]^d$ and $b \in \{0,1\}$, it holds that*

$$\frac{1}{d} \sum_i |v_i - b| = \left| \frac{1}{d} \sum_i v_i - b \right|.$$

To verify this fact, note that when $b = 0$, it is equivalent to $\frac{1}{d} \sum_i |v_i| = \left|\frac{1}{d} \sum_i v_i\right|$, which is true since all $v_i \geq 0$. The case for $b = 1$ is symmetric, since all $v_i \leq 1$.

By Fact 7.3.7, for every $X \in \mathcal{Y} \cup \mathcal{Z}$, it follows that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \\
&= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \mathop{\mathbb{E}}_{x \in_R \{0,1\}^\ell} \left|T_{ij}(x) - f(i,j,x)\right| && \text{(Definition of } f\text{)} \\
&= \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|T_{ij}(x) - f(i,j,x)\right| \\
&= \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|E(i,j,x) - f(i,j,x)\right| \\
&= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|E(i,j,\cdot) - f(i,j,\cdot)\right\|_1 . && (7.16)
\end{aligned}$$

The second-to-last equality above holds by fixing a particular x and setting v as the collection of the $E(i,j,x)$ for all $\mathsf{From}(T_{ij}) = X$ and $b = f(i,j,x)$ (note that b only depends on $\mathsf{From}(T_{i,j})$ since x is fixed), and applying Fact 7.3.7 to show

$$\begin{aligned}
\mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|E(i,j,x) - f(i,j,x)\right| &= \frac{1}{d}\sum_i \left|v_i - b\right| \\
&= \left|\frac{1}{d}\sum_i v_i - b\right| \\
&= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|T_{ij}(x) - f(i,j,x)\right| .
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{i,j \in_R [m]\times[2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 &= \frac{1}{2m} \cdot \sum_{X \in \mathcal{Y}\cup\mathcal{Z}} \sum_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \\
&= \mathop{\mathbb{E}}_{i,j \in_R [m]\times[2]} \left\|f(i,j,\cdot) - E(i,j,\cdot)\right\|_1 && \text{(by (7.16))} \\
&= \left\|f - E\right\|_1 \\
&\le \delta_{\mathsf{v}}. && (7.17)
\end{aligned}$$

Hence, by (7.17) and Item (1) of Lemma 7.3.6, (Y, Z) passes the validity test.

APCPP **accepts** (Y, Z)**.** Now we turn to establish that APCPP accepts (Y, Z). Note that

$$\|T_{ij} - \widehat{T}_{ij}\|_2 \le \|T_{ij} - \widehat{T}_{ij}\|_\infty^{1/2} \cdot \|T_{ij} - \widehat{T}_{ij}\|_1^{1/2} \le \|T_{ij} - \widehat{T}_{ij}\|_1^{1/2}. \tag{7.18}$$

The first inequality above follows from the fact that $\mathbb{E}_i[a_i^2] \le \max_i |a_i| \cdot \mathbb{E}_i |a_i|$ for every real vector a, and the second inequality is implied by $\|T_{ij} - \widehat{T}_{ij}\|_\infty \le 1$.

Hence, it follows from (7.17), (7.18), and Jensen's inequality that

$$\mathop{\mathbb{E}}_{(i,j)\in_{\mathsf{R}}[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 \le \mathop{\mathbb{E}}_{ij} \|T_{ij} - \widehat{T}_{ij}\|_1^{1/2} \le \left(\mathop{\mathbb{E}}_{ij} \|T_{ij} - \widehat{T}_{ij}\|_1\right)^{1/2} \le \sqrt{\delta_{\mathsf{v}}}. \tag{7.19}$$

Finally, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{i\in_{\mathsf{R}}[m]} \mathop{\mathbb{E}}_{x\in_{\mathsf{R}}\{0,1\}^\ell} F_i(x) &\ge \mathop{\mathbb{E}}_i \mathop{\mathbb{E}}_x \widehat{F}_i(x) - \mathop{\mathbb{E}}_i \left\|F_i - \widehat{F}_i\right\|_1 \\
&\ge \mathop{\mathbb{E}}_i \mathop{\mathbb{E}}_x \widehat{F}_i(x) - 6 \cdot \mathop{\mathbb{E}}_{ij} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 && \text{(Item (2.b) of Lemma 7.3.6)} \\
&\ge \mathop{\mathbb{E}}_i \mathop{\mathbb{E}}_x \widehat{F}_i(x) - 6\sqrt{\delta_{\mathsf{v}}} && \text{(by (7.19))} \\
&\ge c_{\mathsf{pcpp}} - \frac{1}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).
\end{aligned}$$

The last inequality follows from $\mathbb{E}_i \mathbb{E}_x \widehat{F}_i(x) \ge c_{\mathsf{pcpp}}$ and $\delta_{\mathsf{v}} = \frac{(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}})}{10^6}$.

Finally, by (7.13), it follows that

$$\mathop{\mathbb{E}}_{i\in[m]} \widetilde{\mathbb{E}}_x(F_i) \ge \mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in_{\mathsf{R}}\{0,1\}^\ell} F_i(x) - \delta_{\mathsf{v}} \ge c_{\mathsf{pcpp}} - \frac{5}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).$$

It shows that (Y, Z) and C are accepted by $\mathsf{APCPP}(z)$, which implies $\mathsf{APCPP}(z) = 1$, contradicting the assumption of Item (3) Lemma 7.3.1. ■

7.3.4 Proof of Lemma 7.3.6

Finally, we present the proof of Lemma 7.3.6. We need the following simple fact.

Lemma 7.3.8. *Let $P_{i,j}, Q_{i,j}, \widetilde{\mathbb{E}}_x(P_{ij}), \widetilde{\mathbb{E}}_x(Q_{ij})$ be stated as in Algorithm 7.3 and δ_{v} be stated as in Algorithm 7.1. For every sufficiently large $n \in \mathbb{N}_{\ge 1}$ and every $(i,j) \in [m] \times [2]$, it holds that*

$$\left|\widetilde{\mathbb{E}}_x(P_{ij}) - \mathop{\mathbb{E}}_x P_{ij}(x)\right| \le \delta_{\mathsf{v}} \text{ and } \left|\widetilde{\mathbb{E}}_x(Q_{ij}) - \mathop{\mathbb{E}}_x Q_{ij}(x)\right| \le \delta_{\mathsf{v}}.$$

Proof. This follows from Lemma 7.2.2 by setting $S(\ell) = 2^{O(\ell^{\varepsilon/2})}$ and $E(\ell) = 2^{\Omega(\ell^\varepsilon)}$. ■

Proof of Lemma 7.3.6. First we establish the completeness condition.

Completeness. Suppose that (Y, Z) satisfies Item (1.a) and (1.b). That is, (1.a) for every $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, Y_s and Z_t are valid $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits, and (1.b) there is a Boolean-valued proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z}(x))$ such that

$$\mathop{\mathbb{E}}_{(i,j)\in_{\mathsf{R}}[m]\times[2]} \|T_{ij} - \widehat{T}_{ij}\|_1 \le \delta_{\mathsf{v}}.$$

Recall that $Q_{ij}(x) = T_{ij}(x)^2$, it is clear that $\mathbb{E}_x Q_{ij}(x) \in [0,1]$ since $T_{ij}(x) \in [0,1]$.

Now we consider $P_{ij}(x)$. Recall that

$$P_{ij}(x) = \begin{cases} T_{ij}(x)^2(1 - T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij}) \in \mathcal{Z}, \\ (\mathsf{Enc}(x)_s - T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij}) = \mathcal{Y}_s \text{ for } s \in [|\mathcal{Y}|]. \end{cases} \tag{7.20}$$

Since $T_{ij}(x) \in [0,1]$ and $\widehat{T}_{ij}(x) \in \{0,1\}$, it follows that

$$T_{ij}(x) \cdot (1 - T_{ij}(x)) \le |\widehat{T}_{ij}(x) - T_{ij}(x)|$$

and

$$P_{ij}(x) \le \left(\widehat{T}_{ij}(x) - T_{ij}(x)\right)^2 \le \left|\widehat{T}_{ij}(x) - T_{ij}(x)\right|.$$

Then we have

$$\mathop{\mathbb{E}}_{(i,j)\in_{\mathsf{R}}[m]\times[2]} \mathop{\mathbb{E}}_{x\in_{\mathsf{R}}\{0,1\}^\ell} P_{ij}(x) \le \mathop{\mathbb{E}}_{(i,j)\in_{\mathsf{R}}[m]\times[2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \le \delta_{\mathsf{v}}.$$

Hence, by Lemma 7.3.8,

$$\begin{cases} \mathop{\mathbb{E}}_{(i,j)\in_{\mathsf{R}}[m]\times[2]} \widetilde{\mathbb{E}}_x(P_{ij}) \le \delta_{\mathsf{v}} + \mathop{\mathbb{E}}_{(i,j)\in_{\mathsf{R}}[m]\times[2]} \mathop{\mathbb{E}}_x P_{ij}(x) \le 2\delta_{\mathsf{v}}, \\ \widetilde{\mathbb{E}}_x(Q_{ij}) \le \delta_{\mathsf{v}} + \mathop{\mathbb{E}}_x Q_{ij}(x) \le 1 + \delta_{\mathsf{v}} \text{ for every } (i,j) \in [m] \times [2]. \end{cases}$$

Therefore, (Y, Z) passes the validity test.

Soundness. Now, suppose (Y, Z) passes the test. That is, the following two conditions are satisfied:

$$\mathop{\mathbb{E}}_{(i,j)\in_{\mathsf{R}}[m]\times[2]} \widetilde{\mathbb{E}}_x(P_{ij}) \le 2\delta_{\mathsf{v}} \text{ and} \tag{7.21}$$

$$\widetilde{\mathbb{E}}_x(Q_{ij}) \le 1 + \delta_{\mathsf{v}} \text{ for every } (i,j) \in [m] \times [2]. \tag{7.22}$$

In the following we prove Item (2.a) and (2.b) separately.

(2.a) We let $\widehat{Y}$ be determined by $\mathsf{Enc}(x)$, and define $\widehat{Z}$ as

$$\widehat{Z}_i(x) = \begin{cases} 0, & Z_i(x) \le \frac{1}{2} \\ 1, & Z_i(x) > \frac{1}{2} \end{cases}. \tag{7.23}$$

We need the following claim for the proof.

Claim 7.3.9. *For every* $(i,j) \in [m] \times [2]$,

$$(T_{ij}(x) - \widehat{T}_{ij}(x))^2 \le 4 \cdot P_{ij}(x). \tag{7.24}$$

Let us now prove the claim. Depending on whether $\mathsf{From}(T_{ij}) = \mathcal{Y}_s$ or $\mathsf{From}(T_{ij}) = \mathcal{Z}_t$, there are two cases.

1. ($\mathsf{From}(T_{ij}) = \mathcal{Y}_s$.) In this case,

$$P_{ij}(x) = (\mathsf{Enc}_s(x) - T_{ij}(x))^2 = \left(\widehat{T}_{ij}(x) - T_{ij}(x)\right)^2$$

since $\widehat{T}_{ij}(x) = \widehat{Y}_s(x) = \mathsf{Enc}_s(x)$.

2. ($\mathsf{From}(T_{ij}) = \mathcal{Z}_t$.) In this case, note that $\left|Z_t(x) - (1 - \widehat{Z}_t(x))\right| \ge 1/2$ by the definition of $\widehat{Z}_t(x)$. Hence

$$\begin{aligned} (T_{ij}(x) - \widehat{T}_{ij}(x))^2 &= \left(\widehat{Z}_i(x) - Z_i(x)\right)^2 \\ &\le \left(\widehat{Z}_t(x) - Z_t(x)\right)^2 \cdot 4 \cdot \left(Z_t(x) - (1 - \widehat{Z}_t(x))\right)^2 \\ &= 4 \cdot (Z_t(x) - 0)^2 \cdot (Z_t(x) - 1)^2 \qquad (\widehat{Z}_t(x) \in \{0,1\}) \\ &= 4 \cdot P_{ij}(x). \end{aligned}$$

This completes the proof of Claim 7.3.9.

It follows that

$$\begin{aligned}
\mathop{\mathbb{E}}_{(i,j)\in \mathsf{R}[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 &\le \mathop{\mathbb{E}}_{(i,j)\in \mathsf{R}[m]\times[2]} \left(\mathop{\mathbb{E}}_{x\in \mathsf{R}\{0,1\}^\ell} 4\cdot P_{ij}(x)\right)^{1/2} && \text{(Claim 7.3.9)}\\
&\le \left(\mathop{\mathbb{E}}_{(i,j)\in \mathsf{R}[m]\times[2]} \mathop{\mathbb{E}}_{x\in \mathsf{R}\{0,1\}^\ell} 4\cdot P_{ij}(x)\right)^{1/2} && \text{(Jensen's inequality)}\\
&\le \left(\mathop{\mathbb{E}}_{(i,j)\in \mathsf{R}[m]\times[2]} 4\cdot \left(\widetilde{\mathbb{E}}_x(P_{ij}) + \delta_{\mathsf{v}}\right)\right)^{1/2} && \text{(Lemma 7.3.8)}\\
&\le \sqrt{12\delta_{\mathsf{v}}}. && \text{(by (7.21))}
\end{aligned}$$

(2.b) Note that by the Cauchy-Schwarz inequality, we have

$$\begin{aligned}
\left\|T_{i1}\cdot T_{i2} - \widehat{T}_{i1}\cdot\widehat{T}_{i2}\right\|_1 &\le \left\|(T_{i1}-\widehat{T}_{i1})\cdot T_{i2}\right\|_1 + \left\|\widehat{T}_{i1}\cdot(\widehat{T}_{i2}-T_{i2})\right\|_1\\
&\le \left\|(T_{i1}-\widehat{T}_{i1})\right\|_2\cdot\left\|T_{i2}\right\|_2 + \left\|\widehat{T}_{i1}\right\|_2\cdot\left\|(\widehat{T}_{i2}-T_{i2})\right\|_2\\
&\le 2\cdot\sum_{j=1,2}\left\|T_{ij}-\widehat{T}_{ij}\right\|_2.
\end{aligned}$$

The last inequality above follows from

$$\begin{aligned}
\|T_{ij}\|_2 &= \left(\mathop{\mathbb{E}}_x Q_{ij}(x)\right)^{1/2} && \text{(Definition of } Q_{ij}(x))\\
&\le \left(1+\widetilde{\mathbb{E}}_x(Q_{ij})\right)^{1/2} \le 2. && \text{(by (7.22))}
\end{aligned}$$

Recall that $F_i(x) = \widetilde{\mathsf{Cons}_i}(T_{i1}(x), T_{i2}(x))$, and $\widehat{F}_i(x) = \widetilde{\mathsf{Cons}_i}(\widehat{T}_{i1}(x), \widehat{T}_{i2}(x))$. Since Cons_i is an OR on two input bits or their negations, one can write

$$\widetilde{\mathsf{Cons}_i}(y_1,y_2) = \sum_{S\subseteq[2]}\alpha_S\cdot\prod_{i\in S} y_i,$$

such that all $|\alpha_S|\le 1$. Therefore, it follows that

$$\|F_i-\widehat{F}_i\|_1 \le \sum_{j=1}^{2}\left\|T_{ij}-\widehat{T}_{ij}\right\|_1 + \left\|T_{i1}\cdot T_{i2}-\widehat{T}_{i1}\cdot\widehat{T}_{i2}\right\|_1 \le 6\cdot\mathop{\mathbb{E}}_{j\in[2]}\left\|T_{ij}-\widehat{T}_{ij}\right\|_2. \quad \blacksquare$$

7.4 From Strong Average-case Witness Lower Bounds to Nondeterministic PRGs

In this section, we apply Theorem 1.5.5 to give a construction of NPRG (defined in Section 3.5) and prove Theorem 1.5.6.

We will need the famous Nisan-Wigderson(NW) PRG ([NW94]), which uses a (presumably hard) function f in its design. Recall that Junta_k is the family of k-juntas, *i.e.*, functions that only depend on k input bits. The key property of the NW PRG is that, given a $\mathscr{C}$ circuit that breaks the NW PRG based on some function, the complexity of approximating that function is $\mathscr{C} \circ \mathsf{Junta}_a$ for some parameter a. Therefore, in order to fool $\mathscr{C}$ circuits, the hard function f used by the NW PRG needs to be hard to approximate by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits.[10]

Lemma 7.4.1 ([NW94])**.** *Let $m, \ell, a \in \mathbb{N}_{\geq 1}$ be such that $a \leq \ell$, and $t = O(\ell^2 \cdot m^{1/a}/a)$. Let $\mathscr{C}$ be a typical and concrete circuit class. There is a function $G^{\mathsf{NW}} \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that the following hold. For any function $Y \colon \{0,1\}^\ell \to \{0,1\}$ represented as a length-2^ℓ truth table, if Y cannot be $(1/2 + \varepsilon/m)$-approximated by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits whose top $\mathscr{C}$ circuit has size S, then $G(Y, \mathcal{U}_t)$ [11] ε-fools every $\mathscr{C}$ circuit of size S and input length m. That is, for any $\mathscr{C}$ circuit $C \colon \{0,1\}^m \to \{0,1\}$ of size S,*

$$\left| \Pr_{\mathbf{s} \in \{0,1\}^t} [C(G^{\mathsf{NW}}(Y, \mathbf{s})) = 1] - \Pr_{\mathbf{x} \in \{0,1\}^m} [C(\mathbf{x}) = 1] \right| \leq \varepsilon.$$

Moreover, the function G^{NW} is computable in $\mathrm{poly}(m, 2^t)$ time.

Now we are ready to prove Theorem 1.5.6 (restated below).

Reminder of Theorem 1.5.6. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in $2^{n - n^\varepsilon}$ time. Then there is a $\delta \in (0,1)$ and an infinity often nondeterministic PRG for 2^{n^δ}-size $\mathscr{C}$ circuits with error 2^{-n^δ}, seed-length $\mathrm{poly}(n)$, and $2^{\mathrm{poly}(n)}$ running time. Consequently, $\mathsf{MA}_{\mathscr{C}} \subseteq$ i.o.-$\mathsf{NTIME}[2^{\log^\beta n}]$ for some $\beta \in \mathbb{N}_{\geq 1}$.*

Proof. Apply Theorem 1.5.5 with $\mathscr{C} \circ \mathsf{AC}^0_2$ (note that the required $\widetilde{\mathsf{CAPP}}$ algorithm for $\mathscr{C} \circ \mathsf{AC}^0_2$ is guaranteed by our assumption here), there is a constant $\gamma \in (0,1)$ and a verifier V such that for infinitely many $n \in \mathbb{N}_{\geq 1}$, $V(1^n, \cdot)$ accepts some $y \in \{0,1\}^{2^n}$, and for every accepted y, $\mathsf{func}(y)$ cannot be $(1/2 + 2^{-n^\gamma})$-approximated by 2^{n^δ}-size $\mathscr{C} \circ \mathsf{AC}^0_2$ circuits. Now, note that a $\mathscr{C} \circ \mathsf{Junta}_{n^{\gamma/2}}$

[10] See, *e.g.*, [CR21, Lemma 2.1] for a formal proof.
[11] Recall that $\mathcal{U}_m$ denotes uniform distribution over $\{0,1\}^m$.

circuits whose top $\mathscr{C}$ circuit has size at most $2^{n^{\gamma/2}}$ can be converted into an $\mathscr{C} \circ \mathsf{AC}^0_2$ circuits with size at most $2^{O(n^{\gamma/2})} \leq 2^{n^\gamma}$.

Therefore, setting $\ell = n, a = n^{\gamma/2}$, and $m = S = 2^{n^{\gamma/2}}$. From the above discussions, we know that every accepted y of $V(1^n, \cdot)$ cannot be $(1/2 + S^{-2})$-approximated by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits whose top $\mathscr{C}$ circuit has size S. Hence, applying Lemma 7.4.1, for some $t = t(n) = O(\ell^2 \cdot m^{1/a}/a) \leq \mathrm{poly}(n)$, it follows that $G^{\mathsf{NW}}(\mathsf{func}(y), \cdot) \colon \{0,1\}^t \to \{0,1\}^m$ is a PRG for S-size $\mathscr{C}$ circuits with error $1/S$.

Finally, letting $\delta = \gamma/2$. We can define our NPRG by $\{G_n\}_{n \in \mathbb{N}_{\geq 1}}$ as follows: For every $n \in \mathbb{N}_{\geq 1}$, $G_n = (G^{\mathsf{P}}_n, G^{\mathsf{W}}_n)$ in which G^{W}_n takes a $\{0,1\}^{2^n}$-bit input y and output $V(1^n, y)$, and G^{P}_n takes $y \in \{0,1\}^{2^n}$ and $s \in \{0,1\}^{t(n)}$ and output $G^{\mathsf{NW}}(y, s)$. By the above discussions, it follows immediately that G is an i.o. NPRG fooling $S = 2^{n^\delta}$-size $\mathscr{C}$ circuits with error 2^{-n^δ} and seed length $\mathrm{poly}(n)$. Also, the running time of G^{W}_n and G^{P}_n are both bounded by $\mathrm{poly}(2^t) = 2^{\mathrm{poly}(n)}$.

To see the "consequently" part, we construct a new NPRG $\{G'_m\}_{m \in \mathbb{N}_{\geq 1}}$ such that $G'_m = G_{(\log m)^{2/\delta}}$. One can verify that G' is an i.o. NPRG fooling $2^{\log^2 m}$-size $\mathscr{C}$ circuits with error $1/10$ and seed-length $\mathrm{polylog}(m)$. Therefore, the proof is completed by applying Lemma 3.5.1.

■

Recall that [Wil11, Wil14a] proved that for every $d, m \in \mathbb{N}_{\geq 1}$, there is an $\varepsilon \in (0,1)$ such that #SAT of $\mathsf{AC}^0_d[m]$ circuits can be solved in 2^{n-n^ε} time. Combining this algorithm with Theorem 1.5.6 (and note that $\mathsf{AC}_d[m] \circ \mathsf{AC}^0_2 \subseteq \mathsf{AC}_{d+2}[m]$), we immediately have the following.

Reminder of Corollary 1.5.7. *For every $d, m \in \mathbb{N}_{\geq 1}$, there exists a $\delta \in (0,1)$ such that there is an i.o. NPRG for 2^{n^δ}-size $\mathsf{AC}^0_d[m]$ circuits with error 2^{-n^δ}, seed-length $\mathrm{poly}(n)$, and $2^{\mathrm{poly}(n)}$ running time. Consequently, $\mathsf{MA}_{\mathsf{AC}^0_d[m]} \subseteq$ i.o.-$\mathsf{NTIME}[2^{\log^\beta n}]$ for some $\beta \in \mathbb{N}_{\geq 1}$.*

Chapter 8

Lower Bounds for Nondeterministic Time Classes from Non-trivial Derandomization

In this chapter we prove the following two theorems.

Theorem 8.0.1 (Weakly average-case lower bound for NQP via direct derandomization; a stronger version of Theorem 4.3.7)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. There are two universal constants $d, \tau \in \mathbb{N}_{\geq 1}$ such that the following holds. Suppose that for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AC}^0_d[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time. Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that neither*

$$\mathsf{NTIME}[2^{\log^\beta n}] \ \textit{nor} \ (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/1}$$

can be $(1 - n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

Theorem 8.0.2 (Strongly average-case lower bound for NQP via an additional win-win argument; a stronger version of Theorem 4.3.9)**.** *Let $\mathscr{C}$ be a typical concrete circuit class that is weaker than* Formula*. Suppose that for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time. Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that neither*

$$\mathsf{NTIME}[2^{\log^\beta n}] \ \textit{nor} \ (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/1}$$

can be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

We remark that while Theorem 8.0.2 is stronger than Theorem 8.0.1 since it requires an $\widetilde{\mathsf{CAPP}}$ for a weaker circuit classes ($\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ vs. $\mathsf{AC}^0_d[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$) and gives stronger average-case lower bounds ($1/2 + 1/\mathrm{poly}(n)$ vs. $1/2 + n^{-\tau}$),[1] the proof of Theorem 8.0.2 needs an additional win-win argument, while the proof of Theorem 8.0.1 is a straightforward direct derandomization, which we believe is easier to understand.

We will prove Theorem 8.0.1 and Theorem 8.0.2 in Section 8.1, assuming some technical ingredients: $\mathsf{MA} \cap \mathsf{coMA}_{\mathscr{C}}$ lower bounds (Theorem 8.1.1), a win-win lemma (Lemma 4.3.8), and a careful derandomization theorem (Theorem 4.3.2).

Next, in Section 8.2, we prove the win-win theorem, using ideas from randomized encodings [IK02, AIK06]. Then, in Section 8.3, we prove the required $\mathsf{MA} \cap \mathsf{coMA}_{\mathscr{C}}$ lower bounds, which in addition requires a PSPACE-complete language with many sophisticated reducibility properties. The construction of the required PSPACE-complete language will be given in Chapter 9. Finally in Section 8.4, we prove the careful derandomization theorem and also provide other missing proofs.

8.1 Circuit Lower Bounds for NQP via Derandomization

8.1.1 Technical Ingredients

We will need several technical ingredients, some of them are already discussed in Chapter 4.

Theorem 8.1.1 (Weakly average-case lower bounds for $\mathsf{MA} \cap \mathsf{coMA}$; a stronger version of Theorem 4.3.6)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. There are universal constants $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, there is a constant $c \in \mathbb{N}_{\geq 1}$ and a language*

$$L \in \left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2] \circ \mathscr{C}} \right)_{/1}$$

such that, for all large enough $n \in \mathbb{N}_{\geq 1}$, there exists $m \in [n, n^c]$ such that L_m cannot be $(1 - m^{-\tau})$-approximated by m^a-size $\mathscr{C}$ circuits.

Reminder of Lemma 4.3.8. *Let $\mathscr{C}$ be a typical concrete circuit class. One of the following holds:*

1. *There is a language $L \in \mathsf{P}$ such that for every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*

[1] We remark that Theorem 8.0.2 also requires $\mathscr{C}$ to be weaker than Formula, while Theorem 8.0.1 do not. So strictly speaking they are incomparable.

2. *There is a constant $\gamma \in \mathbb{N}_{\geq 1}$ such that every S-size formula admits a $\widetilde{\mathsf{Sum}}_{0.01} \circ \mathscr{C}$ circuit of complexity S^{γ}.*

Theorem 8.1.2 (A variant of Theorem 4.3.2)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. Suppose that there is a constant $\varepsilon \in (0,1)$ and an infinity often nondeterministic PRG for $2^{n^{\varepsilon}}$-size $\mathscr{C}$ circuits with error $1/10$, $\mathrm{poly}(n)$ seed-length, and $2^{\mathrm{poly}(n)}$ running time.*

Then, there is a constant $\beta \in \mathbb{N}_{\geq 1}$ that only depends on ε such that for every $L \in (\mathsf{MA} \cap \mathsf{coMA}_{\mathscr{C}})_{/1}$ and $c \in \mathbb{N}_{\geq 1}$, there is an $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{\beta} n}]_{/O(\log \log n)}$ such that for infinitely many $n \in \mathbb{N}$, for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

The proofs of Theorem 8.1.1, Lemma 4.3.8, and Theorem 8.1.2 will be provided in Section 8.3, Section 8.2, and Section 8.4, respectively.

8.1.2 Derandomization Condition and MA Lower Bound Condition

To simplify our presentation, we define the following two conditions, capturing the derandomization consequences of Theorem 8.1.2 and the lower bounds from Theorem 8.1.1, respectively.

Definition 8.1.3 (derandomization condition for $\mathscr{C}$)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. We say that the* derandomization condition *holds for $\mathscr{C}$, if the following holds:*

- *There is universal constant $\beta \in \mathbb{N}_{\geq 1}$ such that for every $L \in ((\mathsf{MA} \cap \mathsf{coMA})_{\mathscr{C}})_{/1}$ and every $c \in \mathbb{N}_{\geq 1}$, there is an*

$$L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{\beta} n}]_{/O(\log \log n)}$$

such that for infinitely many $n \in \mathbb{N}$ and for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

The following is a direct corollary of Theorem 1.5.6 and Theorem 8.1.2.

Corollary 8.1.4. *Let $\mathscr{C}$ be a typical concrete circuit class. If for some $\eta \in (0,1)$, CAPP of $2^{n^{\eta}}$-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in $2^{n-n^{\eta}}$ time, then* derandomization condition *holds for $\mathscr{C}$.*

Definition 8.1.5 (MA lower bound condition for $\mathscr{C}$ and $\mathscr{D}$)**.** *Let $\mathscr{C}$ and $\mathscr{D}$ be two typical concrete circuit class. We say that the* MA lower bound condition *holds for $\mathscr{C}$ and $\mathscr{D}$, if the following holds:*

- *Let $\tau \in \mathbb{N}_{\geq 1}$ be a universal constant. For all $a \in \mathbb{N}_{\geq 1}$, there is constant $c \in \mathbb{N}_{\geq 1}$ and a language*

$$L \in ((\mathsf{MA} \cap \mathsf{coMA})_{\mathscr{D}})_{/1}$$

such that for every $n \in \mathbb{N}$, there exists $m \in [n, n^c]$ such that $\mathsf{heur}_{(1-m^{-\tau})}$-$\mathscr{C}$-$\mathsf{SIZE}(L_m) > m^a$.

For simplicity, when $\mathscr{C} = \mathscr{D}$, we simply say that MA lower bound condition *holds for $\mathscr{C}$.*

The following is a direct corollary of Theorem 8.1.1.

Corollary 8.1.6. *Let $\mathscr{C}$ be a typical concrete circuit class and d_{v} be the constant from Theorem 8.1.1. The* MA lower bound condition *holds for $\mathscr{C}$ and $\mathsf{AC}^0_{d_{\mathsf{v}}}[2] \circ \mathscr{C}$.*

8.1.3 Weak average-case lower bounds for NQP via Direct Derandomization

We will first prove the following weaker version of Theorem 8.0.1, which allows for $O(\log \log n)$ bits of advice. Later, these advice will be eliminated in Section 8.1.5.

Theorem 8.1.7. *Let $\mathscr{C}$ be a typical concrete circuit class. There are two universal constants $d, \tau \in \mathbb{N}_{\geq 1}$ such that the following holds. Suppose that for some $\eta \in (0,1)$,* CAPP *of 2^{n^η}-size $\mathsf{AC}^0_d[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time.*

Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log \log n)}$ cannot be $(1-n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

We will indeed prove a more general result, showing that weak average-case lower bounds for $\mathscr{C}$ follows from appropriate derandomization condition and MA lower bound condition.

Lemma 8.1.8. *Let $\mathscr{C}$ and $\mathscr{D}$ be two typical concrete circuit class. Suppose that the* derandomization condition *holds for $\mathscr{D}$, and the* MA lower bound condition *holds for $\mathscr{C}$ and $\mathscr{D}$.*

Then, there are $\beta, \tau \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log \log n)}$ cannot be $(1-n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

Proof. Let $a \in \mathbb{N}_{\geq 1}$. Let $c \in \mathbb{N}_{\geq 1}$ and $L \in ((\mathsf{MA} \cap \mathsf{coMA})_{\mathscr{D}})_{/1}$ be the constant and hard language guaranteed by the MA lower bound condition for $\mathscr{C}$ and $\mathscr{D}$. Next, from the derandomization condition for $\mathscr{D}$, there is a universal constant $\beta \in \mathbb{N}_{\geq 1}$ and a language $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log \log n)}$ such that for infinitely many n and for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

Now, from the MA lower bound condition for $\mathscr{C}$ and $\mathscr{D}$, for every $n \in \mathbb{N}_{\geq 1}$, there there exists $m \in [n, n^c]$ such that $\mathsf{heur}_{(1-m^{-\tau})}$-$\mathscr{C}$-$\mathsf{SIZE}(L_m) > m^a$. Putting together with our guarantee on L', it follows that for infinitely many $m \in \mathbb{N}_{\geq 1}$, $\mathsf{heur}_{(1-m^{-\tau})}$-$\mathscr{C}$-$\mathsf{SIZE}(L'_m) > m^a$.

Since a is arbitrary, it follows that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log \log n)}$ cannot be $(1-n^{-\tau})$-approximated by n^a-size $\mathscr{C}$ circuits for every $a \in \mathbb{N}_{\geq 1}$. ∎

Now, Theorem 8.1.7 follows directly from Corollary 8.1.4, Corollary 8.1.6 and Lemma 8.1.8.

8.1.4 Strongly Average-case Lower Bounds for NQP via a Win-Win Argument

Next, we move to prove strongly average-case lower bounds. We will first prove the following weaker version of Theorem 8.0.2, which again allows for $O(\log\log n)$ bits of advice.

Theorem 8.1.9. *Let $\mathscr{C}$ be a typical concrete circuit class that is weaker than* Formula*. Suppose that for some $\eta \in (0,1)$,* CAPP *of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time.*

Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^\beta(n)}]_{/O(\log\log n)}$ cannot be $(1/2+\mathrm{poly}(n))$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

We need the following lemma first.

Lemma 8.1.10 (Approximate linear sums preserves PRGs)**.** *Let $\mathscr{C}$ be a typical concrete circuit class, $s \in \mathbb{N}_{\geq 1}$, and $\delta \in (0,1)$. Let $\mathcal{H} \subseteq \mathcal{F}_{s,1}$ be the set of all functions that admit $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits with complexity at most $\sqrt{s}$. If G is a PRG for s-size $\mathscr{C}$ circuits with error s^{-1}, then G is also a PRG for $\mathcal{H}$ with error $1/\sqrt{s}+2\delta$.*

Proof. Let $r \in \mathbb{N}$ be the seed length of G, and let $f \in \mathcal{H}$. From the definition of $\mathcal{H}$, there are $m \leq \sqrt{s}$ many $\mathscr{C}$ circuits $\{C_i\}_{i\in[m]}$, together with m coefficients $\{\alpha_i\}_{i\in[m]}$ such that for every $x \in \{0,1\}^s$, we have

$$\left| f(x) - \sum_{i\in[m]} \alpha_i \cdot C_i(x) \right| \leq \delta.$$

We also have $|C_i| \leq s$ for all $i \in [m]$ and $\sum_{i\in[m]} |\alpha_i| \leq \sqrt{s}$. We also let $L(x) = \sum_{i\in[m]} \alpha_i \cdot C_i(x)$ for notational convenience.

Then, we show G is also a PRG fooling f with the desired error as follows:

$$\begin{aligned}
&\left| \mathop{\mathbb{E}}_{z\in_R\{0,1\}^s}[f(z)] - \mathop{\mathbb{E}}_{r\in_R\{0,1\}^s}[f(G(r))] \right| \\
\leq &\left| \mathop{\mathbb{E}}_{z\in_R\{0,1\}^s}[L(z)] - \mathop{\mathbb{E}}_{r\in_R\{0,1\}^s}[L(G(r))] \right| + 2\delta && (\|f-L\|_\infty \leq \delta) \\
\leq &\sum_{i\in[m]} \alpha_i \left| \mathop{\mathbb{E}}_{z\in_R\{0,1\}^s}[C_i(z)] - \mathop{\mathbb{E}}_{r\in_R\{0,1\}^s}[C_i(G(r))] \right| + 2\delta && (\text{the definition of } L) \\
\leq &\sum_{i\in[m]} |\alpha_i| \cdot s^{-1} + 2\delta && (C_i \text{ has size at most } s) \\
\leq &1/\sqrt{s} + 2\delta. && (\textstyle\sum_{i\in[m]} |\alpha_i| \leq \sqrt{s})
\end{aligned}$$

Hence, G is a PRG fooling $\mathcal{H}$ with error $1/\sqrt{s}+2\delta$ as well. ■

The following corollary follows immediately from Lemma 8.1.10.

Corollary 8.1.11. *Let $\mathscr{C}$ be a typical concrete circuit class, $s \in \mathbb{N}_{\geq 1}$ and $\delta \in (0,1)$. Let $\mathcal{H} \subseteq \mathcal{F}_{s,1}$ be the set of all functions that admit $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits with complexity at most $\sqrt{s}$. If $G = (G^{\mathsf{P}}, G^{\mathsf{W}})$ is an NPRG for s-size $\mathscr{C}$ circuits with error s^{-1}, then G is also an NPRG for $\mathcal{H}$ with error $1/\sqrt{s} + 2\delta$.*

We will need the following lemma, which is a simple corollary of Lemma 4.3.8 and Corollary 8.1.11.

Lemma 8.1.12. *Let $\mathscr{C}$ be a typical concrete circuit class that is weaker than* Formula. *One of the following holds:*

1. *There is a language $L \in \mathsf{P}$ such that for every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*
2. *If for some $\eta \in (0,1)$, CAPP of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time, then* derandomization condition *holds for* Formula.

Proof. We first assume Item (1) does not hold, then by Lemma 4.3.8, every S-size formula admits a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity at most S^γ for some $\gamma \in \mathbb{N}_{\geq 1}$.

From the assumed algorithm for $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits from Item (2) and Theorem 1.5.6, it follows that there is a constant $\varepsilon \in (0,1)$, size parameter $S(n) = 2^{\Omega(n^\varepsilon)}$, and an infinitely often nondeterministic PRG for $S(n)$-size $\mathscr{C}$ circuits with $S(n)^{-1}$-error, $\mathrm{poly}(n)$-seed-length, and $2^{\mathrm{poly}(n)}$ running time.

Now, combining Corollary 8.1.11, the above i.o. NPRG for $\mathscr{C}$, and the simulation of formulas by $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$, it follows that there is another size parameter $S_0(n) = 2^{\Omega(n^\varepsilon)}$ and an infinitely often nondeterministic PRG for $S_0(n)$-size $\mathscr{C}$ circuits, $\mathrm{poly}(n)$-seed-length, and $2^{\mathrm{poly}(n)}$ running time. The second item then follows from Theorem 8.1.2. ∎

Since $\mathsf{AC}^0_{d_\mathsf{v}}[2] \circ$ Formula is weaker than Formula, we have that the MA lower bound condition holds for Formula from Corollary 8.1.6. Hence, the following lemma follows immediately from Lemma 8.1.12 and Lemma 8.1.8.

Lemma 8.1.13. *Let $\mathscr{C}$ be a typical concrete circuit class that is weaker than* Formula. *One of the following holds:*

- *There is a language $L \in \mathsf{P}$ such that for every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*

- *There is $\beta \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log\log n)}$ cannot be $(1 - n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size formulas.*

Finally, we need to perform some mild-to-strong hardness amplification to prove Theorem 8.1.9. The following Lemma follows from a careful analysis of Levin's proof of Yao's XOR Lemma [Lev87, GNW11]. We provide a proof in Section 8.4 for completeness.

Lemma 8.1.14. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a universal constant $c \geq 1$ such that, for every $n \in \mathbb{N}$, $f \in \mathcal{F}_{n,1}$, $\delta \in (0, 0.01)$, $k \in \mathbb{N}$, $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right)$ and $\ell = c \cdot \frac{\log \delta^{-1}}{\varepsilon_k^2}$, if f cannot be $(1 - 5\delta)$-approximated by $\mathsf{MAJ}_\ell \circ \mathscr{C}$ circuits of size $s \cdot \ell + 1$, then $f^{\oplus k}$ cannot be $(\frac{1}{2} + \varepsilon_k)$-approximated by $\mathscr{C}$ circuits of size s.*

Lemma 8.1.15. *Let $\mathscr{C}$ be a typical concrete circuit class, $\tau \in \mathbb{N}_{\geq 1}$. For every $\beta \geq 2$ and every language $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log\log n)}$, there is a language $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log\log n)}$ such that, for two nondecreasing unbounded functions $S, \ell \colon \mathbb{N} \to \mathbb{N}$ such that $n \leq \ell(n) \leq 2^{o(n)}$, the following holds:*

- *If L cannot be $(1 - n^{-\tau})$-approximated by $O(\ell(n)S(n))$-size $\mathsf{MAJ}_{\ell(n)} \circ \mathscr{C}$ circuits, then L' cannot be $(1/2 + \ell(n^{1/(\tau+3)})^{-1/3})$-approximated by $S(n^{1/(\tau+3)})$-size $\mathscr{C}$ circuits.*

Proof. We first define L' as follows: Given an input $x \in \{0,1\}^n$ for some $n \in \mathbb{N}$. Letting m be the largest integer such that $m^{\tau+2} \leq n$, and $k = \min(n - m^{\tau+2}, m^{\tau+1})$, we set $L'(x) = L_m^{\oplus k}(x_{\leq km})$, where $x_{\leq km}$ denotes the first km bits of x. Using the straightforward algorithm for computing L', it follows that $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log\log n)}$.[2]

Now, there are infinitely many $n \in \mathbb{N}_{\geq 1}$ such that L_n cannot be $(1 - n^{-\tau})$-approximated by $\ell(n)S(n)$-size $\mathsf{MAJ}_{\ell(n)} \circ \mathscr{C}$ circuits. We call these n good.

For every sufficiently large good $n \in \mathbb{N}_{\geq 1}$, we set $\delta = n^{-\tau}/5$ and $k = k(n)$ be the smallest integer so that $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right) \geq \ell(n)^{1/3}$. Let c_1 be the universal constant in Lemma 8.1.14. Since n is sufficiently large and $\ell(n) \geq n$, $\ell_0 = c_1 \frac{\log \delta^{-1}}{\varepsilon_k^2} < \ell(n)$. Now, by Lemma 8.1.14 and the fact that L_n cannot be $(1 - 5\delta)$-approximated by $\ell_0 \cdot S(n) + 1 \leq \ell(n) \cdot S(n)$-size $\mathsf{MAJ}_{\ell_0} \circ \mathscr{C}$ circuits, it follows that $(L_n)^{\oplus k}$ cannot be $(1/2 + \ell(n)^{-1/3})$-approximated (note that $\varepsilon_k \leq \ell(n)^{-1/3}$ from our choice) by $S(n)$-size $\mathscr{C}$ circuits.

From our definition of L', it follows that for infinitely many $n \in \mathbb{N}_{\geq 1}$, $L'_{n^{\tau+2}+k(n)}$ (from our choice of k and the assumption that $\ell(n) = 2^{o(n)}$, we have that $k \leq n^{\tau+2}$) cannot be $(1/2 +$

[2] We remark that this step crucially uses the fact that L is in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ instead of $\mathsf{NTIME}[2^{\log^b n}]_{/O(\log\log n)}$.

$\ell(n)^{-1/3}$)-approximated by $S(n)$-size $\mathscr{C}$ circuits, which completes the proof since both S and ℓ are nondecreasing. ■

Now, Theorem 8.1.9 follows immediately from Lemma 8.1.13 and Lemma 8.1.14.

8.1.5 Eliminating or Reducing the Advice

Finally, we will apply the following lemma to get rid or reduce the advice in Theorem 8.1.7 and Theorem 8.1.9. The same trick was used in [COS18] as well.

Lemma 8.1.16. *For every $\beta \geq 2$ and every language $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/O(\log\log n)}$, there are $\tau \in \mathbb{N}_{\geq 1}$ and languages $L_1 \in \mathsf{NTIME}[2^{\log^\beta n}]$ and $L_2 \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/1}$ such that the following holds:*

- *For every typical circuit class $\mathscr{C}$, $S\colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \to (0,1/2)$ such that S, if L cannot be $1/2+\varepsilon(n)$-approximated by $S(n)$-size $\mathscr{C}$ circuits, then neither L_1 nor L_2 can be $1/2+\varepsilon(m(n))$-approximated by $S(m(n))$-size $\mathscr{C}$ circuits, where $m(n)$ is the largest integer such that $m \cdot 2^{\tau\cdot\log\log m} \leq n$.*

Proof. Let $\tau \in \mathbb{N}_{\geq 1}$ be the constant such that $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/\tau\log\log n}$.

For every $n \in \mathbb{N}_{\geq 1}$, let $\ell_n = \tau\log\log n$, and $\{w_i^{(n)}\}_{i\in[2^{\ell_n}]}$ be an enumeration of the set $\{0,1\}^{\ell_n}$. We will prove the lemma for L_1 and L_2 separately.

NTIME lower bounds We first prove the case for $L_1 \in \mathsf{NTIME}[2^{\log^\beta n}]$. We define $L_1 \in \mathsf{NTIME}[2^{\log^\beta n}]$ by the following algorithm A_1: on an input of length n, let m be the largest integer such that $m\cdot 2^{\ell_m} \leq n$, and $k = n - m\cdot 2^{\ell_m} + 1$; A_1 simulates the non-deterministic algorithm for L'_m with the advice w_k on the first m bits of the input. (If $k > 2^{\ell_m}$, A_1 simply outputs 0.)

Since L cannot be $1/2+\varepsilon(n)$-approximated by $S(n)$-size $\mathscr{C}$ circuits, there are infinitely many pairs $(m_i, a_i) \in \mathbb{N} \times \left[2^{\ell_{m_i}}\right]$ such that the non-deterministic algorithm for L_{m_i} with advice w_{a_i} computes a function that cannot be $(1/2+\varepsilon(m_i))$-approximated by $S(m_i)$-size $\mathscr{C}$ circuits.

By the construction of L_1, for infinitely many $(m_i, a_i) \in \mathbb{N} \times \left[2^{\ell_{m_i}}\right]$, $(L_1)_{n_i}$ cannot be $(1/2+\varepsilon(m_i))$-approximated by $S(m_i)$-size $\mathscr{C}$ circuits, where $n_i = m_i\cdot 2^{\ell_{m_i}} + a_i - 1$.

$(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}_{/1}$ lower bounds Now we define $L_2 \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^\beta n}]_{/1}$ by the following algorithm A_2: on an input of length n, let m be the largest integer such that $m\cdot 2^{\ell_m} \leq n$, and $k = n - m\cdot 2^{\ell_m} + 1$; we set the advice bit $\alpha_n = 1$ if and only if $k \leq 2^{\ell_m}$ and w_k is the correct advice for input length m of language L; when $\alpha_n = 1$, A_2 simulates L_m with the advice w_k on the first m

bits of the input; otherwise, A_2 simply outputs 0. A similar argument as that of the previous case completes the proof. ■

Finally, applying Lemma 8.1.16, Theorem 8.0.1 and Theorem 8.0.2 follow immediately from Theorem 8.1.7 and Theorem 8.1.9, respectively.

8.2 Proof of the Win-win Lemma

In this section we prove Lemma 4.3.8. To do so, we first introduce a pair of $\oplus$L-complete problems CMD and DCMD, and state some of their properties that help us to establish Lemma 4.3.8.

8.2.1 $\oplus$L-Complete Problems with Good Properties

Recall that $\oplus$L is the class of problems solvable by a nondeterministic logspace Turing machine that accepts the input if the number of accepting paths is odd.) We define the following two problems, called Connected Matrix Determinant (CMD) and Decomposed Connected Matrix Determinant (DCMD):

Definition 8.2.1. *An instance of* CMD *is an $n \times n$ matrix over* $\mathsf{GF}(2)$ *where the main diagonal and above may contain either* 0 *or* 1*, the second diagonal (i.e. the one below the main diagonal) contains* 1*, and other entries are* 0*. In other words, the matrix is of the following form (where $*$ represents any element in* $\mathsf{GF}(2)$*):*

$$\begin{pmatrix} * & * & * & \cdots & * & * \\ 1 & * & * & \cdots & * & * \\ 0 & 1 & * & \cdots & * & * \\ 0 & 0 & 1 & \cdots & * & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & * \end{pmatrix}.$$

The instance is an $(n(n+1)/2)$-bit string specifying elements on and above the main diagonal. We define $x \in$ CMD if and only if the determinant (over $\mathsf{GF}(2)$*) of the matrix corresponding to x is* 1*.*

An instance of DCMD *is a string of length $n^3(n+1)/2$. For an input x,* $\mathsf{DCMD}(x)$ *is computed as follows: we partition x into blocks of length n^2, let $y_i (i \in [n(n+1)/2])$ be the parity of the i-th block, and define* $\mathsf{DCMD}(x) \coloneqq \mathsf{CMD}(y_1 \circ y_2 \circ \cdots \circ y_{n(n+1)/2})$.

The precise definitions of CMD and DCMD are not important here, as we only need the following two important facts about them.

Theorem 8.2.2 ([AIK06, GGH+07]). *For some $w = w(n) = O(n^4)$, there is a function $P : \{0,1\}^{n(n+1)/2} \times \{0,1\}^w \to \{0,1\}^{n^3(n+1)/2}$ such that the following hold.*

- *For any input $x \in \{0,1\}^{n(n+1)/2}$, the random variable $P(x, \mathcal{D}_w)$ is uniformly distributed in $\{0,1\}^{n^3(n+1)/2}$.*
- *For any $x \in \{0,1\}^{n(n+1)/2}$ and $r \in \{0,1\}^w$, let $P(x,r) = y$, then $\mathsf{CMD}(x) = \mathsf{DCMD}(y) \oplus r_1$, where r_1 is the first bit of r.*
- *For each fixed randomness r, $P(x,r)$ is a projection over x, computable in polynomial time given r.*

Theorem 8.2.3 ([IK02]). CMD *is $\oplus\mathsf{L}$-complete under projections.*

We refer the interested readers to Section 4 of [Vio09c] that contains an excellent exposition of these theorems.

Remark 8.2.4. *Theorem 8.2.2 is essentially a* decomposable randomized encoding *(see, e.g. [App17]) of $\oplus\mathsf{L}$. Such a randomized encoding of NC^1 can also be obtained via Yao's garbled circuit [Yao86, BHR12], which is also enough for our application.*

8.2.2 Proof of Lemma 4.3.8

Now we are ready to prove Lemma 4.3.8. We will indeed prove the following stronger lemma, from which Lemma 4.3.8 follows immediately (from the "consequently" part).

Lemma 8.2.5. *Let $\mathscr{C}$ be a typical circuit class, $\varepsilon\colon \mathbb{N} \to (0, 1/2]$ be an error parameter, and $S\colon \mathbb{N} \to \mathbb{N}$ be a size parameter. Suppose that for every $n \in \mathbb{N}_{\geq 1}$, on input length $n^3(n+1)/2$, DCMD can be $(1/2 + \varepsilon(n))$-approximated by an $S(n)$-size $\mathscr{C}$ circuit.*

Then, for every $\delta > 0$ and $n \in \mathbb{N}_{\geq 1}$, CMD on input length $n(n+1)/2$ has a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of sparsity $O(\varepsilon(n)^{-2}\delta^{-2}n^2)$. Moreover, the sum of absolute values of all coefficients of the $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit is $O(\varepsilon(n)^{-1})$. Consequently, there is a universal constant $\kappa \in \mathbb{N}_{\geq 1}$ such that for every $d \in \mathbb{N}_{\geq 1}$, a depth-$d$ circuit C has an equivalent $\widetilde{\mathsf{Sum}}_{0.01} \circ \mathscr{C}$ circuit of complexity at most $O(S(\mu) \cdot \varepsilon(\mu) \cdot \mu^2)$, where $\mu = 2^{\kappa d}$.

Proof. Let C be the $\mathscr{C}$ circuit that $(1/2 + \varepsilon(n))$-approximates DCMD and let

$$\varepsilon'(n) = \Pr_{z \in_R \{0,1\}^{n^3(n+1)/2}}[C(z) = \mathsf{DCMD}(z)] - 1/2.$$

Note that $\varepsilon'(n) \geq \varepsilon(n)$.

On input length $|x| = n(n+1)/2$, let $P(x,r)$ and $w = w(n)$ be the randomized projection given and randomness parameter in Theorem 8.2.2 such that $\mathsf{CMD}(x) = \mathsf{DCMD}(P(x,r)) \oplus r_1$ for all $r \in \{0,1\}^w$, and $P(x,\mathbf{r})$ is uniformly distributed over $\{0,1\}^{n^3(n+1)/2}$ if $\mathbf{r} \in_R \{0,1\}^w$.

Let $\mathcal{D}$ be a distribution of $\mathscr{C}$ circuits such that a sample D from $\mathcal{D}$ is generated by setting $D(x) := C(P(x,\mathbf{r})) \oplus \mathbf{r}_1$ where $\mathbf{r} \in_R \{0,1\}^w$. Let $\mathbf{D} \sim \mathcal{D}$. Then for any $x \in \{0,1\}^{n(n+1)/2}$, we have

$$\Pr[\mathbf{D}(x) = \mathsf{CMD}(x)] = 1/2 + \varepsilon'(n).$$

The reason is that $\mathbf{D}(x) = \mathsf{CMD}(x)$ if and only if $C(P(x,\mathbf{r})) = \mathsf{DCMD}(P(x,\mathbf{r}))$, which happens with probability exactly $1/2 + \varepsilon'(n)$ since $P(x,\mathbf{r})$ is uniformly distributed.

Now, we draw $t = \Theta(\varepsilon'(n)^{-2}\delta^{-2}n^2)$ i.i.d. random samples $C_1, C_2, \ldots, C_t$ from $\mathcal{D}$, and by a Chernoff bound, for any particular $x \in \{0,1\}^{n(n+1)/2}$, we have

$$\Pr_{C_1,\ldots,C_t \leftarrow \mathcal{D}}\left[\Pr_{i\in[t]}[C_i(x) = \mathsf{CMD}(x)] \in \left(1/2 + (1-2\delta)\varepsilon'(n), 1/2 + (1+2\delta)\varepsilon'(n)\right)\right] > 1 - 2^{-2n^2}.$$

By a union bound, we can select $C_1, C_2, \ldots, C_t$ such that for every $x \in \{0,1\}^{n(n+1)/2}$,

$$\Pr_{i\in[t]}[C_i(x) = \mathsf{CMD}(x)] \in \left(1/2 + (1-2\delta)\varepsilon'(n), 1/2 + (1+2\delta)\varepsilon'(n)\right).$$

To obtain a valid $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ for CMD, we can simply scale the coefficients accordingly as follows.

Consider the following linear combination of $\mathscr{C}$ circuits, which has sparsity $t+1$:

$$E(x) = -\frac{1}{4\varepsilon'(n)} + \frac{1}{2} + \sum_{i=1}^{t} \frac{1}{2\varepsilon'(n)\cdot t} \cdot C_i(x). \tag{8.1}$$

If $\mathsf{CMD}(x) = 1$, it follows that

$$\sum_{i=1}^{t} \frac{1}{2\varepsilon'(n)\cdot t} \cdot C_i(x) \in \left(\frac{1}{4\varepsilon'(n)} + \frac{1}{2} - \delta, \frac{1}{4\varepsilon'(n)} + \frac{1}{2} + \delta\right)$$

and therefore $E(x) \in (1-\delta, 1+\delta)$.

Otherwise, $\mathsf{CMD}(x) = 0$, then

$$\sum_{i=1}^{t} \frac{1}{2\varepsilon'(n)\cdot t} \cdot C_i(x) \in \left(\frac{1}{4\varepsilon'(n)} - \frac{1}{2} - \delta, \frac{1}{4\varepsilon'(n)} - \frac{1}{2} + \delta\right)$$

and therefore $E(x) \in (-\delta, \delta)$. Hence, E is a valid $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit that computes CMD.

It can be seen from (8.1) that the sum of absolute values of all coefficients in E is at most $\frac{t}{2\varepsilon'(n)\cdot t} + O(1) = O(\varepsilon(n)^{-1})$.

Finally, the consequently part follows from the fact that CMD is complete for $\oplus\mathsf{L}$ under projections (Theorem 8.2.3). ■

8.3 Circuit Lower Bounds for $\mathsf{MA}_{\mathsf{AC}^0[2]\circ\mathscr{C}}$ against $\mathscr{C}$

In this section we prove Theorem 8.1.1. We will need to use a PSPACE-complete language with very nice reducibility properties, which are introduced below.

8.3.1 A PSPACE-Complete Language with $\mathsf{AC}^0[2]$ Reducibility Properties

We say a circuit family $\{C_n\}_{n\in\mathbb{N}_{\geq 1}}$ is *uniform*, if there is an uniform algorithm A that outputs the description of C_n given input 1^n, in time polynomial of the size of C_n (*i.e.*, we will only consider P-uniformity). We first define the desired reducibility properties below.

Definition 8.3.1. *Let $L\colon \{0,1\}^* \to \{0,1\}$ be a language, we define the following properties:*

1. *L is $\mathscr{C}$* downward self-reducible *if there is a uniform $\mathscr{C}$ oracle circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every large enough $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, $A^{L_{n-1}}(x) = L_n(x)$.*
2. *L is* paddable, *if there is a polynomial time computable projection* Pad *(i.e., each output bit is either a constant or only depends on 1 input bit), such that for all integers $1 \leq n < m$ and $x \in \{0,1\}^n$, we have $x \in L$ if and only if $\mathsf{Pad}(x, 1^m) \in L$, where $\mathsf{Pad}(x, 1^m)$ always has length m.*
3. *L is* same-length checkable, *if there is a randomized oracle algorithm M with output in $\{0, 1, \perp\}$ such that, for every input $x \in \{0,1\}^*$,*
   (a) M asks its oracle queries only of length $|x|$ and runs in $\mathrm{poly}(|x|)$ time.
   (b) M^{L_n} outputs $L_n(x)$ with probability 1.
   (c) M^O outputs an element in $\{L(x), \perp\}$ with probability at least $2/3$ for every oracle $O\colon \{0,1\}^n \to \{0,1\}$.

   We call M an instance checker *for L. Moreover, we say that L is $\mathscr{C}$ same-length checkable, if there is an instance checker M that can be implemented by uniform $\mathscr{C}$ oracle circuits.*[3]

[3]Formally, suppose M uses at most $p(n) \leq \mathrm{poly}(n)$ bits of randomness on inputs of length n, and let $M(x;r)^O$ be the output of M given input x and randomness r. There is a polynomial-time algorithm A such that, for every $n \in \mathbb{N}$, $A(1^n)$ outputs a $\mathscr{C}$ oracle circuit C_n such that (1) C_n takes $n + p(n)$ bits as input and (2) $C_n(x;r)^O = M(x;r)^O$ for every $(x,r) \in \{0,1\}^n \times \{0,1\}^{p(n)}$ and oracle $O\colon \{0,1\}^n \to \{0,1\}$.

4. L *is* $\mathscr{C}$ weakly error correctable, *if there is a constant* $\tau_{\mathsf{wc2ac}} \geq 1$ *such that for every large enough* $n \in \mathbb{N}$ *and for every function* $\tilde{f}\colon \{0,1\}^n \to \{0,1\}$ *such that* $\tilde{f}$ $(1 - n^{-\tau_{\mathsf{wc2ac}}})$*-approximates* L_n*, there exists an* $n^{\tau_{\mathsf{wc2ac}}}$*-size* $\mathscr{C}$ *circuit* C_n *such that* $C_n^{\tilde{f}}(x) = L_n(x)$ *for every* $x \in \{0,1\}^n$.

Additionally, we say that L *is non-adaptive* $\mathscr{C}$ *downward self-reducible, same-length checkable, or weakly error correctable, if the corresponding* $\mathscr{C}$ *oracle circuits are non-adaptive.*

The following PSPACE-complete language was given by [San09] (modifying a construction of Trevisan and Vadhan [TV07]).

Theorem 8.3.2 ([TV07, San09])**.** *There is a* PSPACE*-complete language* L^{TV} *that is paddable,* TC^0 *downward self-reducible, and same-length checkable.*[4]

Later, [Che19] gave a modification of the language L^{TV} that is also non-adaptive TC^0 same-length checkable, which is further modified by [CR20].

Theorem 8.3.3 ([Che19, CR20])**.** *There is a* PSPACE*-complete language* L^{Chen} *that is paddable, non-adaptive* TC^0 *downward self-reducible, non-adaptive* TC^0 *same-length checkable, and non-adaptive* NC^1 *weakly error correctable.*

In this work, we further improve the complexity of these reducibility properties to $\mathsf{AC}^0[2]$. For convenience, we sometimes allow an algorithm A to take some integers $\alpha_1, \ldots, \alpha_k$ as *input parameters*, and a Boolean string β with length at most $\mathrm{poly}(\sum_{i\in[k]} \alpha_i)$ as *input*. For simplicity we require that $|\beta|$ only depends on $\alpha_1, \ldots, \alpha_k$. We will use $A_{\alpha_1,\ldots,\alpha_k}$ to denote the restriction of A when its input parameters are set to $\alpha_1, \ldots, \alpha_k$.

We say that A can be implemented by a uniform $\mathscr{C}$ circuit family, if there is an algorithm B such that for every $\alpha_1, \ldots, \alpha_k \in \mathbb{N}$, $B(\alpha_1, \ldots, \alpha_k)$ outputs a $\mathrm{poly}(\sum_{i\in[k]} \alpha_i)$-size $\mathscr{C}$ circuits that computes $A_{\alpha_1,\ldots,\alpha_k}$.

Theorem 8.3.4. *There is a* PSPACE*-complete language* L^{PSPACE} *that is paddable, non-adaptive* $\mathsf{AC}^0[2]$ *downward self-reducible, non-adaptive* $\mathsf{AC}^0[2]$ *same-length checkable and non-adaptive* $\mathsf{AC}^0[2]$ *weakly error correctable.*[5]

Moreover, There are two algorithms DSR *and* Aux *satisfying the following*[6]*:*

[4] [TV07] does not explicitly state the TC^0 downward self-reducible property, but it is evident from their proof.

[5] When we say L^{PSPACE} is non-adaptive $\mathsf{AC}^0[2]$ weakly error correctable, we mean it is non-adaptive $\mathsf{AC}^0_d[2]$ weakly error correctable for a universal constant $d \in \mathbb{N}$.

[6] The conditions below are stronger than only being $\mathsf{AC}^0[2]$ downward self-reducible, and will become useful role in our proofs in Section 8.3; see Lemma 8.3.7.

1. Aux *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and output a value from* $\{0,1\}$.
2. Aux *can be implemented by a uniform* $\mathsf{AC}^0[2]$ *circuit.*
3. DSR *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and functions* $h_1 \colon \{0,1\}^{n-1} \to \{0,1\}$ *and* $h_2 \colon \{0,1\}^n \to \{0,1\}$ *as oracles.*
4. *For every* $n \in \mathbb{N}_{\geq 1}$, $\mathsf{DSR}_n^{L_{n-1}^{\mathsf{PSPACE}},\mathsf{Aux}_n}$ *computes* L_n^{PSPACE}.
5. DSR *can be implemented by a uniform non-adaptive* $\mathsf{XOR} \circ \mathsf{AND}_3$ *oracle circuit family. In more details,* DSR *first queries its oracles on some* projections *of the input* $\vec{x}$ *to obtain some intermediate values, and then applies an* $\mathsf{XOR} \circ \mathsf{AND}_3$ *circuit on those intermediate values and the input* $\vec{x}$ *to obtain the output.*

See Chapter 9 for a proof of Theorem 8.3.4. The following corollary follows immediately from the paddable property of L^{PSPACE} in Theorem 8.3.4.

Corollary 8.3.5. *For any circuit class* $\mathscr{C}$, $\mathscr{C}$-$\mathsf{SIZE}(L_n)$ *is non-decreasing.*

8.3.2 Technical Ingredients

We begin with some technical ingredients.

Theorem 8.3.6. *Let* $s \colon \mathbb{N} \to \mathbb{N}$ *be a space-constructible function such that* $s(n) \leq 2^{o(n)}$ *and* $s(n) \geq n$ *for every* n. *There is a universal constant* c *and a language* $L \in \mathsf{SPACE}[s(n)^c]$ *such that* $\mathsf{heur}_{0.99}$-$\mathsf{SIZE}(L_n) > s(n)$ *for all sufficiently large* n.

Proof. In the following we always assume that n is large enough. Let $c_1 \geq 1$ be a large enough constant and let $\ell = c_1 \log s(n)$. There are $2^{2^\ell} = 2^{s(n)^{c_1}}$ many functions in $\mathcal{F}_{\ell,1}$. Also, there are at most $2^{s(n)^2}$ many ℓ-input $s(n)$-size circuits. We claim that there exists a function $f \in \mathcal{F}_{\ell,1}$ that cannot be 0.99-approximated by $s(n)$-size circuits.

To see the claim. Fix an ℓ-input $s(n)$-size circuit C. We draw a random function $f \in \mathcal{F}_{\ell,1}$. By a Chernoff bound, C 0.99-approximates f with probability at most $2^{-\Omega(2^\ell)} \leq 2^{-\Omega(s(n)^{c_1})} \leq 2^{-s(n)^3}$, the last inequality follows from the fact that c_1 and n are large enough. Our claim then follows from a union bound over all $2^{s(n)^2}$ many ℓ-input $s(n)$-size circuits.

Now, letting $c = 2c_1$, our algorithm for L first enumerates all ℓ-bit functions to find the lexicographically first $f_0 \in \mathcal{F}_{\ell,1}$ that cannot be 0.99-approximated by all $s(n)$-size circuits. Note that by our claim above, such f_0 exists for a sufficiently large n. Then our algorithm computes f_0 on the first ℓ bits of the input, and ignores the rest of the input. (Note that here we use the fact that

$\ell \leq O(\log s(n)) \leq o(n)$.) This algorithm can be implemented in $s(n)^c$ space in a straightforward way, and the average-case hardness for L follows from our construction of f_0. ■

The concrete circuit class $\widetilde{\mathsf{AC}}^0_d[2]$. For $d \in \mathbb{N}_{\geq 1}$, we define $\widetilde{\mathsf{AC}}^0_d[2]$ as a sub-class of $\mathsf{AC}^0_d[2]$ such that the top-gate must be an XOR gate and the second layer (counting from the top layer) must all be AND gates. An S-size $\widetilde{\mathsf{AC}}^0_d[2]$ is by definition also an S-size $\mathsf{AC}^0_d[2]$ circuit and an S-size $\mathsf{AC}^0_d[2]$ circuits has an equivalent $O(S)$-size $\widetilde{\mathsf{AC}}^0_{d+2}[2]$ circuit by adding two dummy layers at the top.

The reason we will work with $\widetilde{\mathsf{AC}}^0_d[2]$ instead of $\mathsf{AC}^0_d[2]$ is due to the following lemma.

Lemma 8.3.7. *Let $\mathscr{C}$ be a typical concrete circuit class. There are two universal constants $d_0, c_0 \in \mathbb{N}$ such that the following holds. For every $d_\star \in \mathbb{N}$ such that $d_\star \geq d_0$, letting $\mathscr{D} = \widetilde{\mathsf{AC}}^0_{d_\star}[2] \circ \mathscr{C}$, for all sufficiently large $n \in \mathbb{N}$, we have*

$$\mathscr{D}\text{-}\mathsf{SIZE}(L^{\mathsf{PSPACE}}_n) \leq \left(\mathscr{D}\text{-}\mathsf{SIZE}(L^{\mathsf{PSPACE}}_{n-1}) + n^{c_0}\right)^{c_0},$$

where L^{PSPACE} is the PSPACE*-complete language from Theorem 8.3.4.*

Proof. Let DSR and Aux be the algorithms from Theorem 8.3.4. We set $d_0 \in \mathbb{N}$ so that the Aux_n can be implemented by $\mathrm{poly}(n)$-size $\widetilde{\mathsf{AC}}^0_{d_0}[2]$ circuits. The lemma then follows from the properties of DSR and Aux, and the observation that a size-S $(\mathsf{XOR} \circ \mathsf{AND}_3) \circ \widetilde{\mathsf{AC}}_{d_0}[m_\star]$ circuit can be converted into an $\widetilde{\mathsf{AC}}_{d_0}[m_\star]$ circuit of size $\mathrm{poly}(S)$, since an $\mathsf{AND}_3 \circ \mathsf{XOR}_t$ circuit can be converted into an $\mathsf{XOR}_{O(t^3)} \circ \mathsf{AND}_3$ circuit. ■

8.3.3 Lower Bounds for $\mathsf{MA}_{\mathsf{AC}^0[2] \circ \mathscr{C}}$ against $\mathscr{C}$ Circuits

We will indeed prove the following theorem, from which Theorem 8.1.1 follows immediately.

Theorem 8.3.8 (Formal version of Theorem 8.1.1)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. There are universal constants $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ such that the following holds. For all $a \in \mathbb{N}_{\geq 1}$, there is constant $c \in \mathbb{N}_{\geq 1}$ and a language*

$$L \in \left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2] \circ \mathscr{C}}\right)_{/1}$$

such that the following hold:

1. *For all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*
   - $\mathsf{heur}_{(1-n^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) > n^a$, *or*
   - $\mathsf{heur}_{(1-m^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_m) > m^a$, *for an $m \in (n^c, 2 \cdot n^c) \cap \mathbb{N}$.*
2. *For every $n \in \mathbb{N}$, if the advice for L on n-bit inputs is 0, then L_n is the all-zero function.*

Algorithm 8.1: The $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm A_L

Given an input x with length $n = |x|$;
Given an advice integer $\alpha = \alpha_n \in \{0,1\}$;
if $\alpha = 0$ **then**
    Output 0 and terminate
Let $m = n^c$;
Let $n_0 = n_0(n)$ be the largest integer such that $n_0^c \leq n$;
Let $m_0 = n_0^c$;
Let $\ell = n - m_0$;
if *n is a power of* 2 **then**
    /* We are in the case that $\mathscr{D}$-$\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$. */
    Compute a z in $O(n^c)$ time such that $L_n^{\mathsf{diag}}(x) = L_m^{\mathsf{PSPACE}}(z)$;
    Guess a $\mathscr{D}$ circuit C of size at most n^b;
    Compute in $\mathrm{poly}(m) \leq \mathrm{poly}(n)$ time a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit IC_m that implements the instance checker for L_m^{PSPACE};
    Let $\mathsf{rnd}_m \leq \mathrm{poly}(m)$ be the number of random coins used by IC_m, draw $r \in_{\mathsf{R}} \{0,1\}^{\mathsf{rnd}_m}$;
    Output $\mathsf{IC}_m^C(z,r)$;
else
    /* We are in the case that $\mathscr{D}$-$\mathsf{SIZE}(L_{m_0}^{\mathsf{PSPACE}}) > n_0^b$ and ℓ is the largest integer such that $\mathscr{D}$-$\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n_0^b$. */
    Let z be the first ℓ bits of x;
    Guess a $\mathscr{D}$ circuit C of size at most n_0^b;
    Compute in $\mathrm{poly}(\ell) \leq \mathrm{poly}(n)$ time a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit IC_ℓ that implements the instance checker for L_ℓ^{PSPACE};
    Let $\mathsf{rnd}_\ell \leq \mathrm{poly}(\ell)$ be the number of random coins used by IC_ℓ, draw $r \in_{\mathsf{R}} \{0,1\}^{\mathsf{rnd}_\ell}$;
    Output $\mathsf{IC}_\ell^C(z,r)$;

Algorithm 8.2: The algorithm A_{adv} for setting advice bits in Algorithm 8.1

All the α_n are set to 0 by default;
for $\tau = 1 \to \infty$ **do**
    Let $n = 2^\tau$;
    Let $m = n^c$;
    if $\mathscr{D}$-$\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$ **then**
        Set $\alpha_n = 1$;
    else
        Let $\ell = \max\{\ell : \mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n^b\}$;
        Set $\alpha_{m+\ell} = 1$;

Proof of Theorem 8.3.8. Let L^{PSPACE} be the language from Theorem 8.3.4 and $\beta \in \mathbb{N}_{\geq 1}$ be the constant such that for every $n, s \in \mathbb{N}_{\geq 1}$ such that $s \geq n$, an s-size $\mathscr{C}$ circuit on n-bit inputs has an equivalent s^{β}-size general fan-in 2 Boolean circuits (such β exists since $\mathscr{C}$ is a typical concrete circuit class).

Applying Theorem 8.3.6 with size function $S(n) = n^{\beta \cdot a}$, there is a language $L^{\mathsf{diag}} \in \mathsf{PSPACE}$ such that $\mathsf{heur}_{0.99}\text{-}\mathsf{SIZE}(L^{\mathsf{diag}}_n) > n^{\beta \cdot a}$ for all sufficiently large $n \in \mathbb{N}$. Since L^{PSPACE} is PSPACE-complete and paddable, there is $c_1 \in \mathbb{N}$ such that L^{diag}_n can be reduced to L^{PSPACE} on input length n^{c_1} in $O(n^{c_1})$ time. Let d_0, c_0 be the constants from Lemma 8.3.7.

Let d_{wc2ac} be such that L^{PSPACE} is $\mathsf{AC}^0_{d_{\mathsf{wc2ac}}}[2]$ weakly error correctable, and let τ be the corresponding constant τ_{wc2ac}. We then set $c = c_1$ and also let $d_{\mathsf{g}} = d_0 + d_{\mathsf{wc2ac}} + 2$. We then set $\mathscr{D} = \widetilde{\mathsf{AC}}^0_{d_{\mathsf{g}}}[2] \circ \mathscr{C}$. Let $b \in \mathbb{N}$ be a sufficiently large constant to be chosen later.

The algorithm Let $\tau \in \mathbb{N}_{\geq 1}$ be sufficiently large, $n = 2^{\tau}$, and $m = n^c$. We first provide an informal description of the $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm A_L that computes the hard language L. There are two cases:

1. When $\mathscr{D}\text{-}\mathsf{SIZE}(L^{\mathsf{PSPACE}}_m) \leq n^b$. That is, when L^{PSPACE}_m is *easy*. In this case, on inputs of length n, we guess-and-verify a $\mathscr{D}$ circuit for L^{PSPACE}_m of size at most n^b, and use that to compute L^{diag}_n.
2. Otherwise, we know that L^{PSPACE}_m is *hard*. Let ℓ be the largest integer such that

$$\mathscr{D}\text{-}\mathsf{SIZE}(L^{\mathsf{PSPACE}}_{\ell}) \leq n^b.$$

   On inputs of length $m_1 = m + \ell$, we guess-and-verify a $\mathscr{D}$ circuit for $L^{\mathsf{PSPACE}}_{\ell}$, and compute $L^{\mathsf{PSPACE}}_{\ell}$ on the first ℓ input bits. Note that by Corollary 8.3.5, we have $0 < \ell < m$ and therefore $m + \ell$ is not a power of 2.

Intuitively, the A_L computes a hard function because either it computes the hard language L^{diag}_n on inputs of length n, or it computes the hard language $L^{\mathsf{PSPACE}}_{\ell}$ on inputs of length m. A formal description of A_L is given in Algorithm 8.1, and the algorithm A_{adv} for setting the advice bits of A_L is given in Algorithm 8.2. Since $m + \ell$ at Line 9 is never a power of 2, α_n can only be set once in Algorithm 8.2.

Now we verify that the algorithm above computes a language satisfying our requirements.

A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise We first show that A_L satisfies the MA promise (see Definition 3.4.4). The intuition is that it only tries to guess-and-verify a circuit for L^{PSPACE} when it exists,

and the properties of the instance checker (see Definition 8.3.1) ensure that in this case A_L satisfies the MA promise.

Formally, there are three cases:

1. $\alpha_n = 0$. In this case, A_L computes the all zero function, and clearly satisfies the promise.
2. $\alpha_n = 1$ and n is a power of 2. In this case, from Algorithm 8.2, we have $\mathscr{D}\text{-}\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$ for $m = n^c$. Therefore, at least one guess of the circuit C is the correct circuit for L_m^{PSPACE}, and on that guess, A_L outputs $L_m^{\mathsf{PSPACE}}(z) = L_n^{\mathsf{diag}}(x)$ with probability 1, by the property of the instance checker (see Definition 8.3.1). Again by the property of the instance checker, on all guesses of C, A_L outputs $1 - L_m^{\mathsf{PSPACE}}(z) = 1 - L_n^{\mathsf{diag}}(x)$ with probability at most $1/3$.
3. $\alpha_n = 1$ and n is not a power of 2. In this case, from Algorithm 8.2, we have $\mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n_0^b$. Therefore, at least one guess of the circuit C is the correct circuit for L_ℓ^{PSPACE}, and on that guess, A_L outputs $L_\ell^{\mathsf{PSPACE}}(z)$ with probability 1, again by the property of the instance checker. Similar to the previous case, on all possible guesses of C, A_L outputs $1 - L_\ell^{\mathsf{PSPACE}}(z)$ with probability at most $1/3$.

The following claim summarizes what we have.

Claim 8.3.9. *A_L with advice set by A_{adv} is an $\mathsf{MA}_{/1}$ algorithm for a language L such that, for every $n \in \mathbb{N}$, L_n is defined as below:*

1. *If $\alpha_n = 0$, then L_n is the all-zero function.*
2. *If $\alpha_n = 1$ and n is a power of 2, then L_n is the same function as L_n^{diag}.*
3. *If $\alpha_n = 1$ and n is not a power of 2, then L_n is the n-bit function that computes L_ℓ^{PSPACE} on the first ℓ bits and ignores the rest of the input.*

Also, note that Item (2) of the theorem follows directly from Item (1) of Claim 8.3.9.

The verifier complexity of A_L. We also need to show that A_L is an $\left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_\star + d_\mathsf{v}}[m_\star]}\right)_{/1}$ algorithm. Let $d_{\mathsf{ic}} \in \mathbb{N}_{\geq 1}$ be such that the instance checker for L^{PSPACE} can be implemented by non-adaptive $\widetilde{\mathsf{AC}}^0_{d_{\mathsf{ic}}}[2]$ circuits, then we can see that for every possible guess C in Algorithm 8.1, $\mathsf{IC}^C_m(z,\cdot)$ (resp. $\mathsf{IC}^C_\ell(z,\cdot)$) is a polynomial-size $\widetilde{\mathsf{AC}}^0_{d_\mathsf{g}+d_{\mathsf{ic}}}[m_\mathsf{g}] \circ \mathscr{C}$ circuit (XOR gates can be simulated by $\mathsf{MOD}_{m_\mathsf{g}}$ gates with a linear-size blow-up, since m_g is even). We now set $d_\mathsf{v} = d_\mathsf{g} + d_{\mathsf{ic}}$.

A_L computes a hard language. Finally, we show that the algorithm A_L indeed computes a hard language as stated. Let τ be a sufficiently large integer, $n = 2^\tau$, and $m = n^c$. There are two cases:

1. $\mathscr{D}\text{-}\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$. In this case, we have $\alpha_n = 1$ by Algorithm 8.2. By Item (2) of Claim 8.3.9, we have that L_n is the same function as L_n^{diag}, and therefore $\mathsf{heur}_{0.99}\text{-}\mathsf{SIZE}(L_n) >$

$n^{\beta \cdot a}$, which implies $\mathsf{heur}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) > n^a$, by the definition of β.

2. $\mathscr{D}\text{-}\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) > n^b$. Let ℓ be the largest integer such that $\mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n^b$. By Corollary 8.3.5, we have $0 < \ell < m$.

   Note that $\mathscr{D}\text{-}\mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}}) \leq ((\ell+1)^{c_0} + \mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}))^{c_0}$ from Lemma 8.3.7. Therefore,

$$\mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \geq \left(\mathscr{D}\text{-}\mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}})\right)^{1/c_0} - (\ell+1)^{c_0} \geq n^{b/c_0} - m^{c_0} = n^{b/c_0} - n^{c \cdot c_0}.$$

   Furthermore, recall that $\mathscr{D} = \widetilde{\mathsf{AC}}^0_{d_0 + d_{\mathsf{wc2ac}}+2}[2]$ and L^{PSPACE} is $\mathsf{AC}^0_{d_{\mathsf{wc2ac}}}[2]$ weakly error correctable with constant $\tau_{\mathsf{wc2ac}} = \tau$.[7] We also have

$$\mathsf{heur}_{(1-\ell^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \geq \left[n^{b/c_0} - n^{c \cdot c_0}\right] / O(\ell^\tau).$$

   Now, on inputs of length $m_1 = m + \ell$, we have $\alpha_{m_1} = 1$ by Algorithm 8.2 (note that $m_1 \in (m, 2m)$ as $\ell \in (0, m)$). We set b large enough so that

$$\left[n^{b/c_0} - n^{c \cdot c_0}\right] / O(\ell^\tau) \geq (2n^c)^{a+1} = (2m)^{a+1} \geq m_1^{a+1},$$

   hence we also have $\mathsf{heur}_{(1-\ell^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) > m_1^a$.

   Then by Item (3) of Claim 8.3.9, we have that L_{m_1} is the m_1-input function that computes L_ℓ^{PSPACE} on the first ℓ bits and ignores the last m input bits. Hence, we have

$$\mathsf{heur}_{(1-m_1^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_{m_1}) \geq \mathsf{heur}_{(1-\ell^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_{m_1}) > m_1^a,$$

   which completes the proof.

■

8.4 Missing Proofs

We first prove Theorem 8.1.2, which is restated below.

Reminder of Theorem 8.1.2. *Let $\mathscr{C}$ be a typical concrete circuit class. Suppose that there is a constant $\varepsilon \in (0,1)$ and an infinity often nondeterministic PRG for 2^{n^ε}-size $\mathscr{C}$ circuits with error $1/10$, $\mathrm{poly}(n)$ seed-length, and $2^{\mathrm{poly}(n)}$ running time.*

[7] The $+2$ in the depth of $\mathscr{D}$ corresponds to the fact that $\mathsf{AC}^0_{d_{\mathsf{wc2ac}}}[2]$ can be simulated by $\widetilde{\mathsf{AC}}^0_{d_{\mathsf{wc2ac}}+2}[2]$ with a linear blow-up in size.

Then, there is a constant $\beta \in \mathbb{N}_{\geq 1}$ that only depends on ε such that for every $L \in (\mathsf{MA} \cap \mathsf{coMA}_{\mathscr{C}})_{/1}$ and $c \in \mathbb{N}_{\geq 1}$, there is an $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{\beta} n}]_{/O(\log\log n)}$ such that for infinitely many $n \in \mathbb{N}$, for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

Proof. Let $G = \{G_n\}$ and $\varepsilon \in (0,1)$ be the assumed i.o. NPRG and the corresponding constant. We will first show the desired derandomization for all $L \in (\mathsf{MA}_{\mathscr{C}})_{/1}$, and then generalize it to $(\mathsf{MA} \cap \mathsf{coMA}_{\mathscr{C}})_{/1}$.

Let $k \in \mathbb{N}_{\geq 1}$ be such that $L \in \mathsf{MA}_{\mathscr{C}}\mathsf{TIME}[n^k]_{/1}$. We now construct another i.o. NPRG $\bar{G} = \{\bar{G}_n\}$ that requires $O(\log\log n)$ bits of advice. Jumping ahead, $\bar{G}_n$ simply interprets its advice adv_n as an integer $\ell_n \leq \mathrm{polylog}(n)$, and then computes G_{ℓ_n}. So from now we just assume $\bar{G}_n$ takes an integer $\ell_n \leq \mathrm{polylog}(n)$ as advice.

Let $\mathcal{I}_G$ be the range of G (*i.e.*, $n \in \mathbb{N}_{\geq 1}$ such that G_n is an NPRG for 2^{n^ε}-size $\mathscr{C}$ circuits with the specified parameters). The advice for $\bar{G}$ is set by the following infinite procedure:

1. Initially all the ℓ_n are 0.
2. For every $\alpha \in \mathcal{I}_G$, in increasing order:
   (a) Let n be the largest integer such that $n^{kc} \leq 2^{\alpha^\varepsilon}$.
   (b) If all of $m \in [n, n^c]$ satisfying $\ell_m = 0$ (*i.e.*, all of them are not set yet), then we set $\ell_m = \alpha$ for every $m \in [n, n^c]$.

From the algorithm above, we can verify (1) $\ell_n \leq (\log n)^{2/\varepsilon}$ for all sufficiently large n, meaning that $\bar{G}$ indeed only requires $O(\log\log n)$ bits of advice; (2) $\bar{G}$ is an NPRG for n^k-size $\mathscr{C}$ circuits with $\mathrm{polylog}(n)$-seed-length and $2^{\mathrm{polylog}(n)}$ running time; (3) for infinitely many n, for every $m \in [n, n^c]$, $\bar{G}_m$ works. Since the bound on ℓ_n from (1) only depends on ε, we can show that there exists a constant $\beta \in \mathbb{N}_{\geq 1}$ (that only depends on ε) such that the seed-length and the running time of $\bar{G}_n$ can be bounded by $\log^{\beta} n$ and $2^{\log^{\beta} n}$ for all sufficiently large $n \in \mathbb{N}_{\geq 1}$.

Now, let $\mathcal{I}_{\bar{G}}$ be the range of $\bar{G}$. From (3) above, we know that for infinitely many n, $(\mathbb{N}_{\geq 1} \cap [n, n^c]) \subseteq \mathcal{I}_{\bar{G}}$. Applying Lemma 3.5.1, it follows that there is an $L' \in \mathsf{NTIME}[2^{\log^{\beta} n}]_{/O(\log\log n)}$ such that L and L' agree on all m-bit inputs for every $n \in \mathcal{I}_{\bar{G}}$.

Finally, to finish the proof for $L \in (\mathsf{MA} \cap \mathsf{coMA}_{\mathscr{C}})_{/1}$, we apply the derandomization of $(\mathsf{MA}_{\mathscr{C}})_{/1}$ above for both L and the complement of L. ■

Next, we provide a self-contained proof of Lemma 8.1.14. Our proof below follows a similar structure of the proof of Lemma 1.5.10 (see Section 7.1, which is from [CLW20]).

Reminder of Lemma 8.1.14. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a universal constant $c \geq 1$ such that, for every $n \in \mathbb{N}$, $f \in \mathcal{F}_{n,1}$, $\delta \in (0, 0.01)$, $k \in \mathbb{N}$, $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right)$ and $\ell = c \cdot \frac{\log \delta^{-1}}{\varepsilon_k^2}$, if f cannot be $(1-5\delta)$-approximated by $\mathsf{MAJ}_\ell \circ \mathscr{C}$ circuits of size $s \cdot \ell + 1$, then $f^{\oplus k}$ cannot be $(\frac{1}{2} + \varepsilon_k)$-approximated by $\mathscr{C}$ circuits of size s.*

Proof. Let $c \geq 1$ be a large enough constant. Fix $n \in \mathbb{N}$, $f \in \mathcal{F}_{n,1}$, and $\delta \in (0, 0.01)$. We will prove the following contrapositive of the lemma.

Claim 8.4.1. *For every $k \in \mathbb{N}$, $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right)$, and $\ell_k = c \cdot \frac{\log \delta^{-1}}{\varepsilon_k^2}$, if $f^{\oplus k}$ can be $(\frac{1}{2} + \varepsilon_k)$-approximated by a $\mathscr{C}$ circuit of size s, then f can be $(1-5\delta)$-approximated by an $\mathsf{MAJ}_{\ell_k} \circ \mathscr{C}$ circuit of size $s \cdot \ell_k + 1$.*

Note that Claim 8.4.1 holds trivially when $k = 1$. In the following we prove Claim 8.4.1 by an induction on k.

Let $k \in \mathbb{N}$ be such that $k \geq 2$. For an input x to $f^{\oplus k}$, we write $x = yz$ such that $|y| = n, |z| = (k-1)n$. Letting C be a size-s $\mathscr{C}$ circuit that $(1/2 + \varepsilon_k)$-approximates $f^{\oplus k}$ and assuming that Claim 8.4.1 holds for $k-1$, we consider the following two cases.

Case 1 Suppose for some $y \in \{0,1\}^n$, we have

$$\left|\Pr_z[f^{\oplus k}(y,z) = C(y,z)] - \frac{1}{2}\right| > \frac{\varepsilon_k}{1-\delta} = (1-\delta)^{k-2} \cdot \left(\frac{1}{2} - \delta\right) = \varepsilon_{k-1}.$$

Then, we fix one such y, and note that since $\mathscr{C}$ is typical, either circuit $C'(z) := C(y,z)$ or $\neg C'(z)$ is a size-s $\mathscr{C}$ circuit that $(1/2 + \varepsilon_{k-1})$-approximates $f^{\oplus(k-1)}$. Hence, from our induction hypothesis, f can be $(1-5\delta)$-approximated by an $\mathsf{MAJ}_{\ell_{k-1}} \circ \mathscr{C}$ circuit of size $s \cdot \ell_{k-1} + 1$. This proves Claim 8.4.1 for k since $\ell_{k-1} \leq \ell_k$.

Case 2 Otherwise, for all $y \in \{0,1\}^n$, it holds that

$$\left|\Pr_z[f^{\oplus k}(y,z) = C(y,z)] - \frac{1}{2}\right| \leq \frac{\varepsilon_k}{1-\delta}. \tag{8.2}$$

From now on, we will use ε to denote ε_k for simplicity. We define

$$\begin{aligned} T_y &:= \Pr_z[C(y,z) = f^{\oplus k}(y,z)] \\ &= \Pr_z[C(y,z) = f(y) \oplus f^{\oplus(k-1)}(z)] \\ &= \Pr_z[f(y) = C(y,z) \oplus f^{\oplus(k-1)}(z)]. \end{aligned}$$

From the definition of T_y and (8.2), it follows that for every $y \in \{0,1\}^n$, we have

$$\left| T_y - \frac{1}{2} \right| \leq \frac{\varepsilon}{1-\delta}. \tag{8.3}$$

Also, since C $(\frac{1}{2} + \varepsilon)$-approximates $f^{\oplus k}$, we have

$$\mathop{\mathbb{E}}_y[T_y] \geq 1/2 + \varepsilon. \tag{8.4}$$

We need the following claim first.

Claim 8.4.2. *For at least a $1 - 4\delta$ fraction of $y \in \{0,1\}^n$, it holds that $T_y > 1/2 + \varepsilon/2$.*

Proof. For every $y \in \{0,1\}^n$, we set

$$U_y = \frac{\varepsilon}{1-\delta} - \left(T_y - \frac{1}{2} \right).$$

From (8.3) and (8.4), we have $U_y \geq 0$ for all y and $\mathbb{E}_y[U_y] \leq \frac{\varepsilon}{1-\delta} - \varepsilon \leq 2\delta\varepsilon$, where the last inequality follows from our assumption that $\delta \in (0, 0.01)$.

By a Markov inequality, we have

$$\Pr_y[U_y \geq 1/2\varepsilon] \leq \frac{\mathbb{E}_y[U_y]}{1/2\varepsilon} \leq 4\delta.$$

The claim then follows from the fact that $U_y < 1/2\varepsilon$ implies $T_y > 1/2 + \varepsilon/2$. ∎

Recall that $\ell_k = c \cdot \frac{\log \delta^{-1}}{\varepsilon_k^2}$, where c is sufficiently large universal constant. In the following we will use ℓ_k to denote ℓ for simplicity.

Now for each $i \in [\ell]$, we draw $Z_i \in_R \{0,1\}^{n(k-1)}$ independently. We then define

$$\widetilde{T}_y := \mathop{\mathbb{E}}_{i \leftarrow [\ell]} \left[f(y) = C(y, Z_i) \oplus f^{\oplus(k-1)}(Z_i) \right]. \tag{8.5}$$

Since c is large enough, by a Chernoff bound, it follows that for every $y \in \{0,1\}^n$,

$$\Pr_{\{Z_i\}}\left[\left|T_y - \widetilde{T}_y\right| \geq \varepsilon/6\right] \leq \delta.$$

By an averaging principle, we can fix an assignment to all the Z_i's so that

$$\left|T_y - \widetilde{T}_y\right| < \varepsilon/6 \tag{8.6}$$

holds for at least a $1-\delta$ fraction of $y \in \{0,1\}^n$.

Combining (8.6) and Claim 8.4.2, it follows that for at least a $1-5\delta$ fraction of $y \in \{0,1\}^n$, we have $\widetilde{T}_y > 1/2 + \varepsilon/3$. We then construct an $\mathsf{MAJ}_\ell \circ \mathscr{C}$ E by applying MAJ_ℓ to $\{C(y, Z_i) \oplus f^{\oplus(k-1)}(Z_i)\}_{i\in[\ell]}$. Note that since $f^{\oplus(k-1)}(Z_i)$ is a constant and $\mathscr{C}$ is typical, each $C(y, Z_i) \oplus f^{\oplus(k-1)}(Z_i)$ is a $\mathscr{C}$ circuit of size at most s. Also, by (8.5), $E(y) = f(y)$ if $\widetilde{T}_y > 1/2 + \varepsilon/3$.

To summarize, E is an $\mathsf{MAJ}_\ell \circ \mathscr{C}$ circuit of size at most $\ell \cdot s + 1$ that $(1-5\delta)$-approximates f. This proves Claim 8.4.1 for k. The lemma then follows from an induction on k. ■

Chapter 9

A PSPACE-complete Language with Nice Reducibility Properties

In this chapter we prove Theorem 8.3.4, which is restated below.

Reminder of Theorem 8.3.4. *There is a* PSPACE*-complete language* L^{PSPACE} *that is paddable, non-adaptive* $\mathsf{AC}^0[2]$ *downward self-reducible, non-adaptive* $\mathsf{AC}^0[2]$ *same-length checkable and non-adaptive* $\mathsf{AC}^0[2]$ *weakly error correctable.*

Moreover, There are two algorithms DSR *and* Aux *satisfying the following:*

1. Aux *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and output a value from* $\{0,1\}$.
2. Aux *can be implemented by a uniform* $\mathsf{AC}^0[2]$ *circuit.*
3. DSR *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and functions* $h_1 \colon \{0,1\}^{n-1} \to \{0,1\}$ *and* $h_2 \colon \{0,1\}^n \to \{0,1\}$ *as oracles.*
4. *For every* $n \in \mathbb{N}_{\geq 1}$, $\mathsf{DSR}_n^{L_{n-1}^{\mathsf{PSPACE}},\mathsf{Aux}_n}$ *computes* L_n^{PSPACE}.
5. DSR *can be implemented by a uniform non-adaptive* $\mathsf{XOR} \circ \mathsf{AND}_3$ *oracle circuit family. In more details,* DSR *first queries its oracles on some* projections *of the input* $\vec{x}$ *to obtain some intermediate values, and then applies an* $\mathsf{XOR} \circ \mathsf{AND}_3$ *circuit on those intermediate values and the input* $\vec{x}$ *to obtain the output.*

Notation. In this chapter, we sometimes allow an algorithm A to take some integers $\alpha_1, \ldots, \alpha_k$ as *input parameters*, and a Boolean string β with length at most $\mathrm{poly}(\sum_{i \in [k]} \alpha_i)$ as *input*. For simplicity we require that $|\beta|$ only depends on $\alpha_1, \ldots, \alpha_k$. We will use $A_{\alpha_1,\ldots,\alpha_k}$ to denote the restriction of A when its input parameters are set to $\alpha_1, \ldots, \alpha_k$.

We say that A can be implemented by a uniform $\mathscr{C}$ circuit family, if there is an algorithm B such that for every $\alpha_1, \ldots, \alpha_k \in \mathbb{N}$, $B(\alpha_1, \ldots, \alpha_k)$ outputs a $\mathrm{poly}(\sum_{i\in[k]} \alpha_i)$-size $\mathscr{C}$ circuit that computes $A_{\alpha_1,\ldots,\alpha_k}$.

9.1 Preliminaries

To avoid confusion, we often use bold letters (*e.g.*, $\mathbf{x}$ and $\mathbf{y}$) to emphasize that they are formal variables.

9.1.1 Finite Fields

Throughout this chapter, we will only consider finite fields of the form $\mathsf{GF}(2^{2\cdot 3^\ell})$ for some $\ell \in \mathbb{N}$, since they enjoy simple representations that will be useful for us. For every $\ell \in \mathbb{N}$, we set $\mathsf{pw}_\ell = 2 \cdot 3^\ell$ and use $\mathbb{F}^{(\ell)}$ to denote $\mathsf{GF}(2^{\mathsf{pw}_\ell})$.

Let $\ell \in \mathbb{N}$. We will always represent $\mathbb{F}^{(\ell)} = \mathsf{GF}(2^{\mathsf{pw}_\ell})$ as $\mathbb{F}_2[\mathbf{x}]/(\mathbf{x}^{\mathsf{pw}_\ell} + \mathbf{x}^{\mathsf{pw}_\ell/2} + 1)$.[1] That is, we identify an element of $\mathsf{GF}(2^{\mathsf{pw}_\ell})$ with an $\mathbb{F}_2[\mathbf{x}]$ polynomial with degree less than pw_ℓ. To avoid confusion, given a polynomial $P(\mathbf{x}) \in \mathbb{F}_2[\mathbf{x}]$ with degree less than pw_ℓ, we will use $(P(\mathbf{x}))_{\mathbb{F}^{(\ell)}}$ to denote the unique element in $\mathbb{F}^{(\ell)}$ identified with $P(\mathbf{x})$.

Let $\kappa^{(\ell)}$ be the natural bijection between $\{0,1\}^{\mathsf{pw}_\ell}$ and $\mathbb{F}^{(\ell)} = \mathsf{GF}(2^{\mathsf{pw}_\ell})$: for every $a \in \{0,1\}^{\mathsf{pw}_\ell}$, $\kappa^{(\ell)}(a) = \left(\sum_{i\in[\mathsf{pw}_\ell]} a_i \cdot \mathbf{x}^{i-1}\right)_{\mathbb{F}^{(\ell)}}$. We always use $\kappa^{(\ell)}$ to encode elements from $\mathbb{F}^{(\ell)}$ by Boolean strings. That is, whenever we say that an algorithm takes an input from $\mathbb{F}^{(\ell)}$, we mean it takes a string $x \in \{0,1\}^{\mathsf{pw}_\ell}$ and interprets it as an element of $\mathbb{F}^{(\ell)}$ via $\kappa^{(\ell)}$. Similarly, whenever we say that an algorithm outputs an element from $\mathbb{F}^{(\ell)}$, we mean it outputs a string $\{0,1\}^{\mathsf{pw}_\ell}$ encoding that element via $\kappa^{(\ell)}$. For simplicity, sometimes we use $(a)_{\mathbb{F}^{(\ell)}}$ to denote $\kappa^{(\ell)}(a)$. Also, when we say the i-th element in $\mathbb{F}^{(\ell)}$, we mean the element in $\mathbb{F}^{(\ell)}$ encoded by the i-th lexicographically smallest Boolean string in $\{0,1\}^{\mathsf{pw}_\ell}$.

Finally, for each $n \in \mathbb{N}$, we set ℓ_n to be the smallest integer such that $\mathsf{pw}_{\ell_n} \geq n$. We also let $\mathsf{sz}_n = \mathsf{pw}_{\ell_n} = 2 \cdot 3^{\ell_n}$, $\mathbb{F}_n = \mathbb{F}^{(\ell_n)} = \mathsf{GF}(2^{\mathsf{sz}_n})$, and $\kappa_n = \kappa^{(\ell_n)}$. Note that $2^n \leq |\mathbb{F}_n| \leq 2^{3n}$.

9.1.2 Uniform $\mathsf{AC}^0[2]$ Circuits for Arithmetic Operations over $\mathbb{F}_n$

We will need the following uniform $\mathsf{AC}^0[2]$ circuits for arithmetic operations over $\mathbb{F}_n$ in [HAB02, HV06].

[1] $\mathbf{x}^{2\cdot 3^\ell} + \mathbf{x}^{3^\ell} + 1 \in \mathbb{F}_2[x]$ is irreducible, see [VL99, Theorem 1.1.28].

Lemma 9.1.1 ([HAB02, HV06]). *Let $n, t \in \mathbb{N}$ be input parameters. There are uniform $\mathsf{AC}^0[2]$ circuits for the following two tasks:*

1. Iterated addition*: given a list $a_1, \ldots, a_t \in \mathbb{F}_n$, compute $\sum_{i \in [t]} a_i$.*
2. Iterated multiplication*: given a list $a_1, \ldots, a_t \in \mathbb{F}_n$ such that $t \leq \log n$, compute $\prod_{i \in [t]} a_i$.*

Applying Lemma 9.1.1, we will give two algorithms for polynomial interpolation. Let $\alpha_1, \ldots, \alpha_t$ be the first t non-zero elements from $\mathbb{F}_n$, and $\beta_1, \ldots, \beta_t \in \mathbb{F}_n$ be the input. Let p be the unique degree-$(t-1)$ polynomial such that $p(\alpha_i) = \beta_i$ for every $i \in [t]$. The first algorithm allows us to compute $p(x)$ for any $x \in \mathbb{F}_n$ (given as an additional input) in uniform $\mathsf{AC}^0[2]$, but only works for $t \leq \log n$. The second algorithm only allows us to compute $p(0)$ in uniform $\mathsf{AC}^0[2]$, but works for any number of t. Jumping ahead, the first algorithm will be useful in our construction of instance checkers, and the second algorithm will be useful for establishing weakly error correctability.

Corollary 9.1.2 (Interpolation in $\mathsf{AC}^0[2]$). *There are two algorithms D^{intp} and D^{intp0} satisfying the following:*

1. *(**Input**) D^{intp} and D^{intp0} both take $n, t \in \mathbb{N}$ as input parameters, a list $\beta_1, \ldots, \beta_t \in \mathbb{F}_n$ as input, and outputs an element from $\mathbb{F}_n$. For D^{intp}, we also require that $t \leq \log n$ (i.e., D^{intp} aborts immediately if $t > \log n$) and it takes an additional $x \in \mathbb{F}_n$ as input.*
2. *(**Output**) Let $p(\mathbf{x}) \colon \mathbb{F}_n \to \mathbb{F}_n$ be the unique polynomial with degree at most $t-1$ such that $p(\alpha_i) = \beta_i$ for every $i \in [t]$, where α_i is the i-th non-zero element in $\mathbb{F}_n$. D^{intp} outputs $p(x)$ and D^{intp0} outputs $p(0)$.*
3. *(**Complexity**) Both of D^{intp} and D^{intp0} can be implemented by uniform $\mathsf{AC}^0[2]$ circuit families.*

Proof. For every $i \in [t]$, we define a polynomial $e_i(\mathbf{x}) \colon \mathbb{F}_n \to \mathbb{F}_n$ as follows:

$$e_i(\mathbf{x}) = \prod_{j \in [t] \setminus \{i\}} \frac{\mathbf{x} - \alpha_j}{\alpha_i - \alpha_j}.$$

We have that $p(\mathbf{x}) = \sum_{i \in [t]} e_i(\mathbf{x}) \cdot \beta_i$. Applying Lemma 9.1.1, $p(x)$ can be computed by a uniform $\mathsf{AC}^0[2]$ circuit given x and $\{\beta_i\}_{i \in [t]}$ as input (note that the α_i are constants) when $t \leq \log n$. Also, $p(0)$ can be computed by a uniform $\mathsf{AC}^0[2]$ circuit given $\{\beta_i\}_{i \in [t]}$ as input (since all of the $e_i(0)$ can be precomputed in polynomial time). ∎

9.2 An Adaption of the Construction from [TV07]

We need the following lemma, which builds on the proof of IP = PSPACE theorem [LFKN92, Sha92]. The proof follows similar ideas from the proof of [TV07, Lemma 4.1], but requires several modifications.

Lemma 9.2.1. *There is a collection of polynomials* $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n \in \mathbb{N}_{\geq 1},\, i \in [n]}$ *with the following properties:*

1. *(Term polynomial) There is an algorithm* Term *satisfying the following:*
   (a) Term *takes* $n, i \in \mathbb{N}_{\geq 1}$ *such that* $i \in [n]$ *as input parameters, and* $\vec{x} \in \mathbb{F}_n$ *as input, and output an element from* $\mathbb{F}_n$.
   (b) Term *can be implemented by a uniform* $\mathsf{AC}^0[2]$ *circuit family.*
2. *(Self-reducibility) There is an algorithm* Red *satisfying the following:*
   (a) Red *takes* $n, i \in \mathbb{N}_{\geq 1}$ *such that* $i < n$ *as input parameters, and* $\vec{x} \in \mathbb{F}_n^n$ *as input, and functions* $h_1, h_2 \colon \mathbb{F}_n^n \to \mathbb{F}_n$ *as oracles.*
   (b) $\mathsf{Red}_{n,i}^{f_{n,i+1},\mathsf{Term}_{n,i}}$ *computes* $f_{n,i}$.
   (c) Red *can be implemented by a uniform non-adaptive* $\mathsf{XOR} \circ \mathsf{AND}_2$ *oracle circuit family. In more details,* Red *first queries its oracle on some* projections *of the input* $\vec{x}$ *to obtain some intermediate values, and then applies an* $\mathsf{XOR} \circ \mathsf{AND}_2$ *circuit on those intermediate values and the input* $\vec{x}$ *to obtain the output.*
3. *(Base case) For every* $n \in \mathbb{N}_{\geq 1}$, $f_{n,n}$ *is the constant-one polynomial.*
4. *(*PSPACE*-hardness) For every* $L \in$ PSPACE, *there is a pair of algorithm* $(A_L^{\mathsf{len}}, A_L^{\mathsf{red}})$ *satisfying the following:*
   (a) A_L^{len} *takes* $n \in \mathbb{N}_{\geq 1}$ *as input and outputs an integer in* $\mathrm{poly}(n)$ *time;* A_L^{red} *takes* $x \in \{0,1\}^*$ *as input, and outputs a vector* $\vec{z} \in \mathbb{F}_m^m$ *for* $m = A_L^{\mathsf{len}}(|x|)$.
   (b) *For every* $n \in \mathbb{N}_{\geq 1}$, $A_L^{\mathsf{len}}(n) \leq c_L \cdot n^{c_L}$ *for some constant* $c_L \in \mathbb{N}_{\geq 1}$ *that depends on* L, *and for every* $x \in \{0,1\}^n$, *it holds that* $L(x) = f_{m,1}(\vec{z})$, *where* $m = A_L^{\mathsf{len}}(|x|)$ *and* $\vec{z} = A_L^{\mathsf{red}}(x)$.[2]
5. *(Low degree) For every* $n \in \mathbb{N}_{\geq 1}$ *and* $i \in [n]$, $f_{n,i}$ *has individual degree at most* c_{deg}, *where* c_{deg} *is a universal constant.*
6. *(Instance checker) There is a randomized algorithm* IC *that takes* $n, i \in \mathbb{N}_{\geq 1}$ *such that* $i \in [n]$ *as input parameters, and* $\vec{x} \in \mathbb{F}_n$ *as input, and* $n - i + 1$ *functions* $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ *as oracles, and outputs an element in* $\mathbb{F}_n \cup \{\perp\}$. *The following properties hold for* IC*:*

[2] *i.e.*, $f_{m,1}(\vec{z}) = (L(x))_{\mathbb{F}_m}$.

(a) *If* $\tilde{f}_j = f_{n,j}$ *for every* $j \in \{i, \ldots, n\}$, *then* $\mathsf{IC}_{n,i}^{\tilde{f}_i,\ldots,\tilde{f}_n}(\vec{x})$ *outputs* $f_{n,i}(\vec{x})$ *with probability* 1 *for every* $\vec{x} \in \mathbb{F}_n^n$.

(b) *For every* $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ *and every* $\vec{x} \in \mathbb{F}_n^n$, $\mathsf{IC}_{n,i}^{\tilde{f}_i,\ldots,\tilde{f}_n}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$ *with probability* $1 - 1/2^n$, *over the internal randomness of* IC.

(c) IC *can be implemented by a randomized uniform non-adaptive* $\mathsf{AC}^0[2]$ *circuit family.*

We prove Lemma 9.2.1 together with some additional properties of $\mathscr{F}^{\mathsf{TV}}$ below.

9.2.1 A PSPACE-complete Problem TQBF$^{\mathsf{u}}$

We first introduce a variant of the standard PSPACE-complete problem TQBF (True Quantified Boolean Formula), which we call TQBF$^{\mathsf{u}}$. For $n \in \mathbb{N}_{\geq 1}$, we let $\phi_n^{\mathsf{cl\text{-}idx}}$ be a bijection from $[8 \cdot \binom{n}{3}]$ to $\binom{[n]}{3} \times \{0,1\}^3$. Here, for a set S and an integer $m \in \mathbb{N}$, we use $\binom{S}{m}$ to denote the set of all size-m subsets of S.

Definition 9.2.2. *The* TQBF$^{\mathsf{u}}$ *problem*[3] *takes a vector* $\vec{y} \in \{0,1\}^{8 \cdot \binom{n}{3}}$ *as input, and the goal is to decide whether the following quantified Boolean formula holds:*

$$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \bigwedge_{u \in [8 \cdot \binom{n}{3}]} [\neg y_u \vee \Phi_u(\vec{x})]\,. \tag{9.1}$$

where Q_i *equals* $\exists$ *for odd* i, $\forall$ *for even* i, *and* $\vec{x} = (x_1, \ldots, x_n)$.

For $u \in [8 \cdot \binom{n}{3}]$, *letting* $(S, \vec{\tau}) = \phi_n^{\mathsf{cl\text{-}idx}}(u)$, *where* $S = \{s_1, s_2, s_3\}$ *for* $s_1 < s_2 < s_3$, $\Phi_u(\vec{x})$ *is defined as*

$$\Phi_u(\vec{x}) = \bigvee_{i \in [3]} [(x_{s_i} \wedge \tau_i) \vee (\neg x_{s_i} \wedge \neg \tau_i)]\,.$$

We use TQBF$^{\mathsf{u}}_n$ *to denote the* TQBF$^{\mathsf{u}}$ *problem with parameter* n *(and input length* $8 \cdot \binom{n}{3}$*).*

We first show that TQBF$^{\mathsf{u}}$ is still PSPACE-complete.

Lemma 9.2.3. TQBF$^{\mathsf{u}}$ *is* PSPACE*-complete.*

Proof. Recall that the standard TQBF problem is defined as follows: given an n-variable m-clause 3-CNF $\phi(\vec{x})$ as input, the goal is to decide whether $Q_1 x_1 Q_2 \cdots Q_n x_n\, \phi(\vec{x})$ holds, where Q_i equals $\exists$ for odd i, and $\forall$ for even i. By adding dummy variables or dummy clauses, we can assume $n = m$.

[3] u stands for universal, since in (9.1) we have a universal formula that can simulate every n-variable 3-CNF.

Let $\vec{y} = 0^{8\cdot\binom{n}{3}}$ initially. For every $j \in [n]$, letting $C_j(\vec{x})$ be the j-th clause in $\phi(\vec{x})$, there exists an index $u \in [8 \cdot \binom{n}{3}]$ such that $C_j(\vec{x}) \equiv \Phi_u(\vec{x})$ and we set $y_u = 1$.

Now we can verify that $\mathsf{TQBF}^{\mathsf{u}}(\vec{y}) = \mathsf{TQBF}(\phi)$ from (9.1). This proves the PSPACE-hardness of $\mathsf{TQBF}^{\mathsf{u}}$ as TQBF is PSPACE-complete [SM73] (see also [AB09, Theorem 4.13]). From its definition, it is also clear that $\mathsf{TQBF}^{\mathsf{u}}$ is in PSPACE, which completes the proof. ■

9.2.2 Construction of $\mathscr{F}^{\mathsf{TV}}$

Next, we will formally define the collection of polynomials $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n\in\mathbb{N}_{\geq 1},\, i\in[n]}$.

We begin with some notation. For a vector $\vec{x} \in \mathbb{F}_n^n$, $i \in [n]$, and $z \in \mathbb{F}_n$, we use $\vec{x}^{i\leftarrow z}$ to denote the vector obtained from $\vec{x}$ by changing x_i to z. For a polynomial $p(\vec{x}) \colon \mathbb{F}^n \to \mathbb{F}$ and $i \in [n]$, we use $\deg_{x_i}(p)$ to denote the maximum degree of x_i in p.

We also state the following lemma, which gives details on how the self-reduction Red in Item (2) of Lemma 9.2.1 is implemented.

Lemma 9.2.4 (Self-reduction for $\mathscr{F}^{\mathsf{TV}}$)**.** *Let $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n\in\mathbb{N}_{\geq 1},\, i\in[n]}$ and* Term *be as in Lemma 9.2.1. For every $n, i \in \mathbb{N}_{\geq 1}$ such that $i < n$, one can compute an index $J = J_{n,i} \in [n]$ and a type $Q = Q_{n,i} \in \{\exists, \forall, \mathsf{LIN}, \mathsf{MUL}\}$ in $\mathrm{poly}(n)$ time such that the following hold for every vector $\vec{x} \in \mathbb{F}_n^n$:*

1. *If $Q = \forall$, then*
$$f_{n,i}(\vec{x}) = f_{n,i+1}(\vec{x}^{J\leftarrow 0}) \cdot f_{n,i+1}(\vec{x}^{J\leftarrow 1}).$$
2. *If $Q = \exists$, then*
$$f_{n,i}(\vec{x}) = 1 - (1 - f_{n,i+1}(\vec{x}^{J\leftarrow 0})) \cdot (1 - f_{n,i+1}(\vec{x}^{J\leftarrow 1})).$$
3. *If $Q = \mathsf{LIN}$, then*
$$f_{n,i}(\vec{x}) = x_J \cdot f_{n,i+1}(\vec{x}^{J\leftarrow 1}) + (1 - x_J) \cdot f_{n,i+1}(\vec{x}^{J\leftarrow 0}).$$
4. *If $Q = \mathsf{MUL}$, then*
$$f_{n,i}(\vec{x}) = \mathsf{Term}_{n,i}(\vec{x}) \cdot f_{n,i+1}(\vec{x}).$$

To simplify our presentation, we further define three polynomials $S_\exists, S_\forall, S_{\mathsf{LIN}}$ as

1. $S_\forall(x, y_0, y_1) = y_0 \cdot y_1$.
2. $S_\exists(x, y_0, y_1) = 1 - (1 - y_0) \cdot (1 - y_1)$.
3. $S_{\mathsf{LIN}}(x, y_0, y_1) = xy_1 + (1 - x)y_0$.

Now the first three cases in Lemma 9.2.4 can be succinctly written as

$$f_{n,i}(\vec{x}) = S_Q(x_j, f_{n,i+1}(\vec{x}^{J\leftarrow 0}), f_{n,i+1}(\vec{x}^{J\leftarrow 1})). \tag{9.2}$$

Construction of $\mathscr{F}^{\mathsf{TV}}$. Now we are ready to define $\mathscr{F}^{\mathsf{TV}}$. Let $n \in \mathbb{N}$ and let m be the largest integer such that $20m^6 \leq n$. We will use $\{f_{n,i}\}_{i\in[n]}$ to encode the problem $\mathsf{TQBF}^{\mathsf{u}}_m$. When $n < 20$, we simply set $f_{n,i}$ to be the constant-zero n-variate polynomial for all $i \in [n]$. So we can assume $m \geq 1$. We also let $m_y = 8 \cdot \binom{m}{3}$. (Note that $\mathsf{TQBF}^{\mathsf{u}}_m$ has m_y input bits.)

Recall that for $i \in [m]$, $Q_i = \exists$ for odd i and $Q_i = \forall$ for even i.

Letting $\lambda = m + m_y + 1$. For convenience, we first construct two other polynomial families $\{g^{(n)}_{i,j} : \mathbb{F}^n_n \to \mathbb{F}_n\}_{i,j\in[\lambda]}$ and $\{\mathsf{Term}^{(n)}_i : \mathbb{F}^n_n \to \mathbb{F}_n\}_i$ as follows:

1. $g^{(n)}_{\lambda,j}$ are all set to be constant-one polynomials for every $j \in [\lambda]$.
2. For every i from $m + m_y$ down to 1:
   (a) If $i \in [m]$, we set
   $$g^{(n)}_{i,\lambda}(\vec{x}) = S_{Q_i}(x_i, g^{(n)}_{i+1,1}(\vec{x}^{i\leftarrow 0}), g^{(n)}_{i+1,1}(\vec{x}^{i\leftarrow 1})) \tag{9.3}$$
   for every $\vec{x} \in \mathbb{F}^n_n$.
   (b) Otherwise $i > m$. Let $u = m + m_y + 1 - i$ and $(S, \vec{\tau}) = \phi^{\mathsf{cl\text{-}idx}}_m(u)$ such that $S = \{s_1, s_2, s_3\} \subseteq [m]$ where $s_1 < s_2 < s_3$. We define
   $$\mathsf{Term}^{(n)}_i(\vec{x}) = 1 - (1 - x_{m+u}) \cdot \prod_{\ell\in[3]} \Big[1 - [\tau_\ell \cdot x_{s_\ell} + (1 - \tau_\ell) \cdot (1 - x_{s_\ell})]\Big], \tag{9.4}$$
   and
   $$g^{(n)}_{i,\lambda}(\vec{x}) = g^{(n)}_{i+1,1}(\vec{x}) \cdot \mathsf{Term}^{(n)}_i(\vec{x}). \tag{9.5}$$
   (c) For every j from $m + m_y$ down to 1:
      i. We set
      $$g^{(n)}_{i,j}(\vec{x}) = S_{\mathsf{LIN}}(x_j, g^{(n)}_{i,j+1}(\vec{x}^{j\leftarrow 0}), g^{(n)}_{i,j+1}(\vec{x}^{j\leftarrow 1})). \tag{9.6}$$

Intuitively speaking, in the above process, we start from the base constant-one polynomial and keep multiplying the last polynomial with the arithmetization of a single term in (9.1), and then apply linearization to all variables to reduce the individual degrees of all variables. Then, we

obtain the multi-linear extension of the base formula[4]

$$\bigwedge_{u\in[8\cdot\binom{n}{3}]} [\neg x_{m+u} \vee \Phi_u(\vec{x})]\,. \tag{9.7}$$

After that, we apply the $\exists$ and $\forall$ quantifiers alternatively, each followed by LIN operators, just as in [TV07].

We will first prove some properties of the family $\{g_{i,j}^{(n)} : \mathbb{F}_n^n \to \mathbb{F}_n\}_{i,j\in[\lambda]}$, and then show how to construct $\mathscr{F}^{\mathsf{TV}}$ based on them.

Lemma 9.2.5. *For every $i \in [\lambda]$ the following hold:*

1. *The individual degree of $g_{i,1}^{(n)}$ is at most 1.*
2. *For every $j \in [\lambda]$, the individual degree of $g_{i,j}^{(n)}$ is at most 2.*
3. *$g_{i,1}^{(n)}$ agrees with $g_{i,\lambda}^{(n)}$ on all points from $\{0,1\}^n$.*

Proof. First, we observe that for every $i, j \in [\lambda]$, $g_{i,j}^{(n)}$ does not depend on any variable x_k with $k > m + m_y$ (*i.e.*, $\deg_{x_k}(g_{i,j}^{(n)}) = 0$). Next, by the definition of S_{LIN} together with (9.6), we can see that for every $i, j \in [\lambda - 1]$, we have (1) $\deg_{x_j}(g_{i,j}^{(n)}) \le 1$, (2) $\deg_{x_\ell}(g_{i,j}^{(n)}) \le \deg_{x_\ell}(g_{i,j+1}^{(n)})$ for $\ell \in [n] \setminus \{j\}$, and (3) $g_{i,j}^{(n)}$ and $g_{i,j+1}^{(n)}$ agree on all points from $\{0,1\}^n$.

From above discussions, Item (1) and (3) follow immediately (the case of $i = \lambda$ just follows from our definition). To see Item (2), again by the above discussions, it suffices to verify that the individual degree of $g_{i,\lambda}^{(n)}$ for every $i \in [\lambda - 1]$ are bounded by 2. When $i \in [m]$, this follows from the definition (9.3) and the fact that $g_{i+1,1}^{(n)}$ has individual degree at most 1. When $i > m$ it follows from the fact that $\mathsf{Term}_i^{(n)}$ has individual degree at most 1 and the definition (9.5). ∎

Lemma 9.2.6. *$g_{m+1,1}^{(n)}$ is the linear extension of (9.7).*[5]

Proof. We first note that for every $i \in [m + m_y] \setminus [m]$, letting $\mu_i = m + m_y + 1 - i$, from (9.4), $\mathsf{Term}_i^{(n)}$ is the linear extension of the clause

$$C_i := \neg x_{m+\mu_i} \vee \Phi_{\mu_i}(\vec{x}).$$

Now we will use a simple induction to prove the following claim, which easily implies this lemma (by setting $i = m + 1$).

[4]Note that here we concatenate $\vec{x}$ and $\vec{y}$, so y_u corresponds to x_{m+u}.
[5]We treat (9.7) as a Boolean function on n bits by adding $n - (m + m_y)$ dummy variables at the end.

Claim 9.2.7. *For every $i \in [m+m_y] \setminus [m]$, $g_{i,1}^{(n)}$ is the linear extension of*

$$\Psi_i := \bigwedge_{u \in [\mu_i]} [\neg x_{m+u} \vee \Phi_u(\vec{x})] .$$

The base case $i = m + m_y$ can be established by the fact that $g_{i,\lambda}^{(n)} = g_{i+1,1}^{(n)} \cdot \mathsf{Term}_i^{(n)} = \mathsf{Term}_i^{(n)}$ and Item (3) of Lemma 9.2.5. Now, assuming that the claim holds for $i+1 \in \{m+2, \ldots, m+m_y\}$, we show it holds for i as well. Since $g_{i+1,1}^{(n)}$ is the linear extension of Ψ_{i+1}, $\Psi_i = \Psi_{i+1} \wedge C_i$, and $\mathsf{Term}_i^{(n)}$ is the linear extension of C_i, we know that $g_{i,\lambda}^{(n)} = g_{i+1,1}^{(n)} \cdot \mathsf{Term}_i^{(n)}$ agrees with Ψ_i on all points from $\{0,1\}^n$. Now, from Item (1) and Item (3) of Lemma 9.2.5, we know that $g_{i,1}^{(n)}$ is both multi-linear and agrees with Ψ_i on all points from $\{0,1\}^n$. Therefore, $g_{i,1}^{(n)}$ is the linear extension of Ψ_i. ■

Finally, we show that $g_{1,1}^{(n)}$ correctly encodes $\mathsf{TQBF}_m^{\mathsf{u}}$.

Lemma 9.2.8. *For every $\vec{x} \in \{0,1\}^m$, $\vec{z} \in \{0,1\}^{n-m-m_y}$, and $\vec{y} \in \{0,1\}^{m_y}$, $g_{1,1}^{(n)}(\vec{x},\vec{y},\vec{z}) = \mathsf{TQBF}_m^{\mathsf{u}}(\vec{y})$.*

Proof. For every $i \in [m+1]$, we define the following quantified formula

$$\Lambda_i(\vec{x},\vec{y}) := Q_i x_i Q_{i+1} x_{i+1} \cdots Q_m x_m \bigwedge_{u \in [8 \cdot \binom{m}{3}]} [\neg y_u \vee \Phi_u(\vec{x})] ,$$

where $\vec{x} \in \{0,1\}^{i-1}$ and $\vec{y} \in \{0,1\}^{m_y}$.

In particular, $\Lambda_{m+1}(\vec{x},\vec{y}) = \bigwedge_{u \in [8 \cdot \binom{m}{3}]} [\neg y_u \vee \Phi_u(\vec{x})]$ and $\Lambda_1(\vec{y}) = \mathsf{TQBF}_m^{\mathsf{u}}(\vec{y})$.

We will again use a simple induction to prove the following claim, which easily implies this lemma (by setting $i = 1$).

Claim 9.2.9. *For every $i \in [m+1]$, $\vec{x} \in \{0,1\}^{i-1}$, $\vec{w} \in \{0,1\}^{m-(i-1)}$, $\vec{z} \in \{0,1\}^{n-m-m_y}$, and $\vec{y} \in \{0,1\}^{m_y}$, $g_{i,1}^{(n)}(\vec{x},\vec{w},\vec{y},\vec{z}) = \Lambda_i(\vec{x},\vec{y})$.*

The base case $i = m+1$ is exactly Lemma 9.2.6. Now, assuming that the claim holds for $i+1 \in \{2, \ldots, m+1\}$, we show it holds for i as well. Note that $\Lambda_i(\vec{x},\vec{y}) = Q_i x_{i+1}\, \Lambda_{i+1}(\vec{x}, x_{i+1}, \vec{y})$, the claim then follows from the definition of $g_{i,\lambda}$ (see (9.3)) and Item (3) of Lemma 9.2.5. ■

Now, recall that $m_y = 8 \cdot \binom{m}{3} \leq 2m^3$ and $20m^6 \leq n$. We have $\lambda^2 = (m+m_y+1)^2 \leq 16 \cdot m^6 < n$. We are now ready to define $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i}\}_{n \in \mathbb{N}_{\geq 1},\, i \in [n]}$ as follows: for every $(i,j) \in [\lambda]$, we set $f_{n,(i-1)\cdot\lambda+j} = g_{i,j}^{(n)}$ and for every $\ell \in [n] \setminus [\lambda^2]$, we set $f_{n,\ell}$ to be the constant-one polynomial.

Now we are ready to set $Q_{n,\ell}$, $J_{n,\ell}$, and $\mathsf{Term}_{n,\ell}$ for every $\ell \in [n]$ accordingly to prove Lemma 9.2.4.

Proof of Lemma 9.2.4. For every $\ell \in [n-1] \setminus [\lambda^2]$, we set $Q_{n,\ell} = \mathsf{LIN}$ and $J_{n,\ell} = 1$. We note that when $f_{n,\ell+1}$ is the constant-one polynomial, from (9.2), $f_{n,\ell}$ is also the constant-one polynomial. Hence, Lemma 9.2.4 holds for $\ell \in [n-1] \setminus [\lambda^2]$.

Next, for every $i, j \in [\lambda]$:

1. Let $\ell = (i-1) \cdot \lambda + j$.
2. If $j < \lambda$, we set $Q_{n,\ell} = \mathsf{LIN}$ and $J_{n,\ell} = j$.
3. Otherwise,
   (a) If $i > m$, we set $Q_{n,\ell} = \mathsf{MUL}$, $J_{n,\ell} = 1$, and $\mathsf{Term}_{n,\ell} = \mathsf{Term}_i^{(n)}$.
   (b) Otherwise $i \in [m]$, we set $Q_{n,\ell} = Q_i$ and $J_{n,\ell} = i$.

For every ℓ that $\mathsf{Term}_{n,\ell}$ is not defined above, we set $\mathsf{Term}_{n,\ell}$ to be the constant-zero function. Lemma 9.2.4 then follows from the definition of $g_{i,j}^{(n)}$ and $\mathsf{Term}_i^{(n)}$. ■

9.2.3 Proof of Lemma 9.2.1

Now we are ready to prove Lemma 9.2.1. We first prove every item except for Item (6).

Proof of Lemma 9.2.1, except for Item (6). First, Item (1) follows immediately from the definition of $\mathsf{Term}_i^{(n)}$ in (9.4), and Item (3) of Lemma 9.2.1 follows immediately from the definition of $f_{n,n}$.

Item (2.b) follows immediately from Lemma 9.2.4. To see Item (2.c), we note that according to Lemma 9.2.4, (1) $f_{n,i}(\vec{x})$ can be computed by a degree-2 polynomial over $\vec{x}$ and the outputs returned by queries to $f_{n,i+1}$ and $\mathsf{Term}_{n,i}$, and (2) the queries to $f_{n,i+1}$ or $\mathsf{Term}_{n,i}$ are projections on $\vec{x}$. Item (2.c) then follows from the observation that degree-2 polynomials over $\mathbb{F}_n$ can be computed by an $\mathsf{XOR} \circ \mathsf{AND}_2$ circuit.

Item (4) follows from the fact that $f_{n,1} = g_{1,1}^{(n)}$, Lemma 9.2.8, and the PSPACE-completeness of $\mathsf{TQBF}^{\mathsf{u}}$ (Lemma 9.2.3); Item (5) follows immediately from the definition of $f_{n,i}$ and Lemma 9.2.5. ■

Finally, to prove Item (6) of Lemma 9.2.1, we give a detailed implementation of the instance checker IC in Algorithm 9.1 and show that it can indeed be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ circuit family.

Proof of Item (6) of Lemma 9.2.1. Fix $n, i \in \mathbb{N}_{\geq 1}$ such that $i \in [n]$. Let $\vec{x} \in \mathbb{F}_n^n$ be the input and $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ be the $n-i+1$ oracle functions. We first note that throughout Algorithm 9.1, we have $r_j = \tilde{f}_j(\vec{\alpha}_j)$ for every $j \in \{i, \ldots, n\}$.

Algorithm 9.1: The instance checker IC from Item (6) of Lemma 9.2.1

Given $n, i \in \mathbb{N}_{\geq 1}$ such that $i \in [n]$ as input parameters, and $\vec{x} \in \mathbb{F}_n$ as the input;
Given $n - i + 1$ functions $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ as the oracles;
Let $\vec{\alpha}_i = \vec{x}$ and $r_i = \tilde{f}_i(\vec{x})$; // The initial goal is to verify the claim $f_{n,i}(\vec{x}) = r_i$
for $j \in \{i, i+1, \ldots, n-1\}$ **do**
    /* The goal at the j-th stage is to verify the claim $f_{n,j}(\vec{\alpha}_j) = r_j$ */
    Compute $J = J_{n,j}$ and $Q = Q_{n,j}$ from Lemma 9.2.4;
    Draw $z_j \in_{\mathsf{R}} \mathbb{F}_n$;
    if $Q \in \{\exists, \forall, \mathsf{LIN}\}$ **then**
        Let $t = c_{\mathsf{deg}} + 1$ and $w_1, \ldots, w_t$ be the first t non-zero elements in $\mathbb{F}_n$; // c_{deg} is the constant from Item (5) of Lemma 9.2.1
        Set $\beta_\ell = \tilde{f}_{j+1}((\vec{\alpha}_j)^{J \leftarrow w_\ell})$ for every $\ell \in [t]$;
        Let $\mathcal{L} = \{\beta_\ell\}_{\ell \in [t]}$;
        if $r_j \neq S_Q((\vec{\alpha}_j)_J, D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 0), D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 1))$ **then**
            return $\perp$;
        /* Reduce the verification of $f_{n,j}(\vec{\alpha}_j) = r_j$ to the verification of $f_{n,j}(\vec{\alpha}_{j+1}) = r_{j+1}$ */
        Set $\vec{\alpha}_{j+1} = (\vec{\alpha}_j)^{J \leftarrow z_j}$ and $r_{j+1} = \tilde{f}_{j+1}(\vec{\alpha}_{j+1})$;
        if $r_{j+1} \neq D^{\mathsf{intp}}_{n,t}(\mathcal{L}, z_j)$ **then**
            return $\perp$;
    else
        Set $\vec{\alpha}_{j+1} = \vec{\alpha}_j$ and $r_{j+1} = \tilde{f}_{j+1}(\vec{\alpha}_{j+1})$;
        if $r_j \neq r_{j+1} \cdot \mathsf{Term}_{n,j}(\vec{\alpha}_j)$ **then**
            return $\perp$;
/* Finally, check the final claim to see if it is correct. It should equal to 1 according to Item (3) of Lemma 9.2.1 */
if $r_n = 1$ **then**
    return r_i;
else
    return $\perp$;

Completeness. We first establish Item (6.a) (*i.e.*, the completeness). Assuming that $\tilde{f}_j = f_{n,j}$ for every $j \in \{i, \ldots, n\}$, it follows that in Algorithm 9.1, $D^{\mathsf{intp}}_{n,t}(\mathcal{L}, \alpha)$ always equals to $f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow \alpha})$ for every $\alpha \in \mathbb{F}_n$.[6] Since we also have $r_j = \tilde{f}_j(\vec{\alpha}_j)$, by Lemma 9.2.4, we know that Algorithm 9.1 always passes the check on Line 11, Line 14, and Line 18. Finally, since $r_n = \tilde{f}_n(\vec{\alpha}_n) = f_{n,n}(\vec{\alpha}_n) = 1$ (by Item (3) of Lemma 9.2.1), we know that IC outputs $r_i = \tilde{f}_i(\vec{\alpha}_i) = f_{n,i}(\vec{x})$ with probability 1.

Soundness. Next we establish Item (6.b) (*i.e.*, the soundness). We will do so by establishing the following claim.

Claim 9.2.10. *For every $j \in \{i, \ldots, n\}$, if $r_j \neq f_{n,j}(\vec{\alpha}_j)$, then for every fixed randomness $z_i, \ldots, z_{j-1}$, with probability at least $1 - (n-j) \cdot c_{\mathsf{deg}}/2^n$ over the random choice of $z_j, \ldots, z_{n-1}$, we have $\mathsf{IC}^{\tilde{f}_i, \ldots, \tilde{f}_n}_{n,i}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$.*

Proof. We will prove the claim by induction. For the base case $j = n$, the claim immediately follows from the final check (Line 20) of Algorithm 9.1 since IC would always output $\perp$.

Now, assuming that the claim holds for $j+1 \in \{i+1, \ldots, n\}$, we will show it holds for j as well. The case for $Q = \mathsf{MUL}$ follows straightforwardly from the definition of $f_{n,j}$ and $f_{n,j+1}$, so in the following we assume $Q \in \{\exists, \forall, \mathsf{LIN}\}$.

Let $p \colon \mathbb{F}_n \to \mathbb{F}_n$ be the degree-c_{deg} polynomial such that $p(\alpha) = D^{\mathsf{intp}}_{n,t}(\mathcal{L}, \alpha)$ for every $\alpha \in \mathbb{F}_n$ (see Item (2) of Corollary 9.1.2), and $q \colon \mathbb{F}_n \to \mathbb{F}_n$ be the restriction of $f_{n,j+1}$ defined by $q(\alpha) = f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow \alpha})$. Note that q has degree at most c_{deg} by Item (5) of Lemma 9.2.1. We consider two cases separately, $p = q$ and $p \neq q$.

First, suppose $p = q$. In this case, we have

$$\begin{aligned} S_Q((\vec{\alpha}_j)_J, D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 0), D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 1)) &= S_Q((\vec{\alpha}_j)_J, q(0), q(1)) \\ &= S_Q((\vec{\alpha}_j)_J, f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow 0}), f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow 1})) = f_{n,j}(\vec{\alpha}_j). \end{aligned}$$

Hence, it follows that $r_j \neq S_Q((\vec{\alpha}_j)_J, D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 0), D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 1))$ and IC does not pass the test on Line 11, and outputs $\perp$ immediately.

Next, suppose $p \neq q$. Since both polynomials have degree at most c_{deg} and note that z_j is independent of p and q (they both only depend on $z_i, \ldots, z_{j-1}$, which are fixed by the assumption), it follows that $p(z_j) = q(z_j)$ with probability at most $c_{\mathsf{deg}}/|\mathbb{F}_n| \leq c_{\mathsf{deg}}/2^n$. Conditioning on the

[6] Here we crucially used the fact that the individual degree of $f_{n,i}$ is bounded by the constant c_{deg}, so that Corollary 9.1.2 can be applied.

event that $p(z_j) \neq q(z_j)$, we have that $r_{j+1} = p(z_j)$ (otherwise the check on Line 14 fails and IC outputs $\perp$ immediately) and $f_{n,j+1}(\vec{\alpha}_{j+1}) = q(z_j) \neq r_{j+1}$. Hence, by the induction hypothesis, it follows that $\mathsf{IC}_{n,i}^{\tilde{f}_i,\ldots,\tilde{f}_n}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$ happens with probability at least $1 - (n-j) \cdot c_{\mathsf{deg}}/2^n$ over the random choice of $z_j, \ldots, z_{n-1}$. ■

Applying the claim above with $j = i$, we have that $\mathsf{IC}_{n,i}^{\tilde{f}_i,\ldots,\tilde{f}_n}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$ with probability at least $1 - n \cdot c_{\mathsf{deg}}/2^n$. This error probability can be further amplified to at most $1/2^n$ (the stated error probability in Item (6) of Lemma 9.2.1) as follows: run the IC $t = O(1)$ times to obtain outputs $\alpha_1, \ldots, \alpha_t$; output $\perp$ if any of the α_i equals $\perp$ or they are not all identical (*i.e.*, there are $i < j$ such that $\alpha_i \neq \alpha_j$), and output α_1 otherwise. The amplification procedure can be implemented in AC^0, so it won't affect the complexity of IC, which is discussed below.

Complexity. Finally, we show that IC can be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ circuit family.

The crucial observation here is that we can first draw $z_i, \ldots, z_{n-1} \in_{\mathsf{R}} \mathbb{F}_n$ beforehand and run each iteration of the for loop in Algorithm 9.1 in parallel (and return $\perp$ if any of the checks on Line 11, Line 14, or Line 18 fails). Note that for each $j \in \{i, \ldots, n\}$ and $\ell \in [n]$, we have

$$(\vec{\alpha}_j)_\ell = \begin{cases} x_\ell & \text{there is no } j' < j \text{ such that } J_{n,j'} = \ell \text{ and } Q_{n,j'} \neq \mathsf{MUL} \\ z_{j_{\mathsf{max}}} & \text{otherwise,} \end{cases} \tag{9.8}$$

where j_{max} is the maximum $j' < j$ such that $J_{n,j'} = \ell$ and $Q_{n,j'} \neq \mathsf{MUL}$.

Using (9.8), for every $j \in \{i, \ldots, n\}$, we can compute $\vec{\alpha}_j$ by a uniform projection given $\vec{x}, z_i, \ldots, z_{n-1}$. It then follows from Algorithm 9.1, Corollary 9.1.2, and Lemma 9.1.1 that IC can be implemented by a randomized uniform non-adaptive AC^0_2 circuit family. ■

9.3 Construction of the PSPACE-complete Language

In this section, we prove Theorem 8.3.4. We will first construct a PSPACE-complete language $L^{\mathsf{WH\text{-}TV}}$, and then prove it satisfies all the desired properties stated in Theorem 8.3.4 except for the paddability. Then we modify $L^{\mathsf{WH\text{-}TV}}$ into another PSPACE-complete language L^{PSPACE} that also satisfies the paddability.

9.3.1 The Language $L^{\mathsf{WH\text{-}TV}}$

To construct our PSPACE-complete language $L^{\mathsf{WH\text{-}TV}}$, we apply Walsh-Hadamard codes to turn the polynomials from Lemma 9.2.1 into Boolean functions. (Indeed, WH-TV stands for "Walsh-Hadamard version of $\mathscr{F}^{\mathsf{TV}}$".)

Let $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n \in \mathbb{N}_{\geq 1},\, i \in [n]}$ be as in Lemma 9.2.1. First, we list all polynomials in $\mathscr{F}^{\mathsf{TV}}$ in the following order

$$f_{1,1}, f_{2,2}, \ldots, f_{2,1}, f_{3,3}, \ldots, f_{3,1}, \ldots, f_{n,n}, \ldots, f_{n,1}, \ldots. \tag{9.9}$$

For every $k \in \mathbb{N}$, we let g_k be the k-th polynomial in (9.9). We also set n_k and i_k so that $g_k = f_{n_k,i_k}$. When their meanings are clear from the context, we may write n and i instead of n_k and i_k. Recall that κ_n is the bijection from $\{0,1\}^{\mathsf{sz}_n}$ to $\mathbb{F}_n$ described in Section 9.1.

Construction of the interpolated polynomial G_k. We now define the following polynomial $G_k \colon \mathbb{F}_n^n \times \mathbb{F}_n^n \to \mathbb{F}_n$:

$$G_k(\vec{x}, \vec{y}) \coloneqq \sum_{j=i_k}^{n_k} f_{n_k,j}(x) \cdot y_j, \tag{9.10}$$

where $\vec{x} \in \mathbb{F}_n^n$ and $\vec{y} \in \mathbb{F}_n^n$.

We define $F_k \colon \mathbb{F}_n^{2n} \times \{0,1\}^{\mathsf{sz}_n} \to \{0,1\}$ as

$$F_k(\vec{z}, \vec{r}) \coloneqq \langle \kappa_n^{-1}(G_k(\vec{z})), \vec{r} \rangle, \tag{9.11}$$

where $\langle \kappa_n^{-1}(G_k(\vec{z})), \vec{r} \rangle$ denotes the inner product between the two vectors over $\mathsf{GF}(2)$.

F_k can be interpreted as a function from $\{0,1\}^{e_k}$ to $\{0,1\}$, where $e_k = (2 \cdot n_k + 1) \cdot \mathsf{sz}_{n_k}$. The following claim follows immediately from the definition of e_k.

Claim 9.3.1. *For every $k \in \mathbb{N}_{\geq 1}$, it holds that $e_k < e_{k+1}$.*

The language $L^{\mathsf{WH\text{-}TV}}$. Now we are ready to define $L^{\mathsf{WH\text{-}TV}}$ via the following algorithm.

From Claim 9.3.1 and Algorithm 9.2, the following claim is immediate.

Claim 9.3.2. *For every $k \in \mathbb{N}_{\geq 1}$, $L^{\mathsf{WH\text{-}TV}}_{e_k}$ equals F_k.*

Algorithm 9.2: Algorithm $A^{\mathsf{WH\text{-}TV}}$ for $L^{\mathsf{WH\text{-}TV}}$

Given an input $x \in \{0,1\}^m$ for some $m \in \mathbb{N}$;
if $m < e_1$ **then**
  return 0
Let k be the largest integer such that $e_k \leq m$;
return $F_k(x_{\leq e_k})$;

9.3.2 Verifying Properties of $L^{\mathsf{WH\text{-}TV}}$

Next, we verify that $L^{\mathsf{WH\text{-}TV}}$ has all the desired properties stated in Theorem 8.3.4. First, we show $L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ same-length checkable.

Lemma 9.3.3. *$L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ same-length checkable.*

Proof. Let $m \in \mathbb{N}$ be an input length and we can assume $m \geq e_1$ since otherwise $A^{\mathsf{WH\text{-}TV}}_m$ computes the constant-zero function. Let k be the largest integer such that $e_k \leq m$ and it suffices to establish the instance checkability of F_k. In the following we use n to denote n_k and i to denote i_k.

Instance checker for G_k. We first show how to establish an instance checker $\mathsf{G\text{-}IC}$ for G_k.

Recall that

$$G_k(\vec{x}, \vec{y}) := \sum_{j=i}^{n} f_{n,j}(x) \cdot y_j, \tag{9.12}$$

for every $\vec{x}, \vec{y} \in \mathbb{F}_n^n$.

Note that for every $j \in \{i, i+1, \ldots, n\}$, letting $\vec{r}_j$ be the vector from $\mathbb{F}_n^n$ with every entry being 0 except for the j-th entry being 1, we have

$$f_{n,j}(\vec{x}) = G_k(\vec{x}, \vec{r}_j) \quad \text{for every } \vec{x} \in \mathbb{F}_n^n, \tag{9.13}$$

meaning that the oracle access to $f_{n,j}$ can be simulated by the oracle access to G_k via fixing part of the input. $\mathsf{G\text{-}IC}$ works as follows:

1. Given $\vec{x} \in \mathbb{F}_n^n$ and $\vec{y} \in \mathbb{F}_n^n$ as input, and access to an oracle $\tilde{G} \colon \mathbb{F}_n^{2n} \to \mathbb{F}_n$ that is supposed to compute G_k.
2. For every $j \in \{i, i+1, \ldots, n\}$, $\mathsf{G\text{-}IC}$ runs $\mathsf{IC}_{n,j}$ (from Item (6) of Lemma 9.2.1) on input $\vec{x}$ with oracle access to $\tilde{f}_{j+1}, \ldots, \tilde{f}_n$ simulated by $\tilde{G}$ via (9.13) to obtain an output $u_j \in \mathbb{F}_n \cup \{\perp\}$.[7]
3. If any of the u_j equals $\perp$, we output $\perp$. Otherwise, we output $\sum_{j=i}^{n} u_j \cdot y_j$.

[7] That is, $\tilde{f}_\ell(\vec{x}) = \tilde{G}(\vec{x}, \vec{r}_\ell)$ for every $\ell \in \{j+1, \ldots, n\}$.

Since $\mathsf{IC}_{n,i}$ can be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ oracle circuit, so does G-IC. (Applying Lemma 9.1.1, we can use a uniform $\mathsf{AC}^0[2]$ circuit to implement the algorithm above.)

Now we show that when $\tilde{G} = G_k$, G-IC outputs $G_k(\vec{x}, \vec{y})$ with probability 1. Note that for every $j \in \{i, i+1, \ldots, n\}$, since $\tilde{G} = G_k$, we have $\tilde{f}_\ell = f_{n,\ell}$ for every $\ell \in \{j+1, \ldots, n\}$ from (9.13). Applying Item (6) of Lemma 9.2.1, it holds that with probability 1, $u_j = f_{n,j}(\vec{x})$ for every $j \in \{i, i+1, \ldots, n\}$. Therefore, with probability 1, G-IC outputs $\sum_{j=i}^{n} u_j \cdot y_j$, which equals $G_k(\vec{x}, \vec{y})$ by definition.

Next we show that for every oracle $\tilde{G}$, with probability at least $2/3$, $\mathsf{G\text{-}IC}^{\tilde{G}}$ outputs either $G_k(\vec{x}, \vec{y})$ or $\perp$. We first note that by Item (6) of Lemma 9.2.1 and a union bound, with probability at least $2/3$, $u_j \in \{f_{n,j}(\vec{x}), \perp\}$ for every $j \in \{i, i+1, \ldots, n\}$, which implies that G-IC outputs either $G_k(\vec{x}, \vec{y})$ (when no u_j equals $\perp$) or $\perp$ (when some u_j equals $\perp$). This completes the construction of the instance checker G-IC for G_k.

Instance checker for F_k. Next we show how to construct the desired instance checker F-IC for F_k:

1. Given $\vec{x} \in \mathbb{F}_n^{2n}$ and $\vec{z} \in \{0,1\}^{\mathsf{sz}_n}$ as input, and access to an oracle $\tilde{F} \colon \mathbb{F}_n^{2n} \times \{0,1\}^{\mathsf{sz}_n} \to \{0,1\}$ that is supposed to compute F_k.
2. F-IC simulates G-IC on input $\vec{x}$ given oracle access to the function[8]

$$\vec{x} \mapsto \tilde{F}(\vec{x}, \vec{e}_1) \circ \tilde{F}(\vec{x}, \vec{e}_2) \circ \cdots \circ \tilde{F}(\vec{x}, \vec{e}_{\mathsf{sz}_n}),$$

   to obtain an output $u \in \mathbb{F}_n \cup \{\perp\}$.
3. F-IC outputs $\perp$ if u equals $\perp$ and outputs $\langle \kappa_n^{-1}(u), \vec{z} \rangle$ (inner product is over $\mathsf{GF}(2)$) otherwise.

Since we encode an element of $\mathbb{F}_n$ via κ_n, when $\tilde{F} = F_k$, G-IC above indeed gets access to G_k, and hence F_k outputs $\langle \kappa_n^{-1}(G_k(\vec{x})), \vec{z} \rangle = F_k(\vec{x}, \vec{z})$. Also, for every oracle $\tilde{F}$, from the promise of G-IC, we know that G-IC outputs an element in $\{G_k(\vec{x}), \perp\}$ with probability at least $2/3$. This implies that F-IC outputs an element in $\{F_k(\vec{x}, \vec{z}), \perp\}$ with probability at least $2/3$ as well. Therefore, F-IC is an instance checker for F_k. Since G-IC can be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ oracle circuit, so does F-IC. ■

To prove that $L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ weakly error correctable, we need the following standard decoder for Reed-Muller codes. While the decoding algorithm is a standard interpolation, we analyze its complexity carefully using Lemma 9.1.1 and show that the decoder can be

[8] Below $\vec{e}_\ell$ denotes the sz_n-bit vector with every entry being 0 except for the ℓ-th entry being 1

implemented by an $\mathsf{AC}^0[2]$ circuit.

Lemma 9.3.4 ($\mathsf{AC}^0[2]$ decoder for Reed-Muller code, [AB09, Section 19.3, 19.4])**.** *Let $n, m \in \mathbb{N}_{\geq 1}$ and $\mathbb{F} = \mathbb{F}_n$. Suppose there is a (hidden) degree-d m-variate polynomial P over $\mathbb{F}$, and $\delta \in \left(0, \frac{1}{3(d+1)}\right)$. For any oracle $O\colon \mathbb{F}^m \to \mathbb{F}$ such that*

$$\Pr_{\vec{x} \in_{\mathsf{R}} \mathbb{F}^m}[O(\vec{x}) = P(\vec{x})] > 1 - \delta,$$

there is a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit C of size $\mathrm{poly}(m, n)$, such that for every $\vec{x} \in \mathbb{F}^m$, $C^O(\vec{x}) = P(\vec{x})$.

Proof. Let $\vec{x} \in \mathbb{F}^m$ be the input. We will show a randomized non-adaptive $\mathsf{AC}^0[2]$ oracle circuit $\mathbf{C}^O$ (with the oracle set to O) of size $\mathrm{poly}(m, n)$ that computes $P(\vec{x})$ with probability at least $2/3$. Then by Adleman's argument, we can obtain a deterministic oracle $\mathsf{AC}^0[2]$ circuit of size $\mathrm{poly}(m, n)$ that correctly computes P, by drawing $\mathrm{poly}(m, n)$ independent samples from $\mathbf{C}^O$ and applying approximate majorities to the output (which can be done in AC^0; see Lemma 3.1.1).

We choose a random vector $\vec{\mathbf{v}} \in_{\mathsf{R}} \mathbb{F}^m$, and for every $\alpha \in \mathbb{F}$ we define $Q(\alpha) = P(\vec{x} + \alpha \cdot \vec{\mathbf{v}})$ (note that $Q\colon \mathbb{F} \to \mathbb{F}$ has degree at most d). Let $\alpha_1, \ldots, \alpha_{d+1}$ be the first $d+1$ non-zero elements from $\mathbb{F}$. We then query O to obtain $\beta_i = O(\vec{x} + \alpha_i \cdot \vec{\mathbf{v}})$ for every $i \in [d+1]$, and output $D^{\mathsf{intp0}}_{n,d+1}(\{\beta_i\}_{i \in [t+1]})$. This algorithm can be implemented in $\mathsf{AC}^0[2]$ by Corollary 9.1.2.

To show the correctness of the algorithm above. Let $\mathbf{z}$ denote the number of i's from $[d+1]$ such that $O(\vec{x} + \alpha_i \cdot \vec{\mathbf{v}}) \neq Q(\alpha_i)$, then $\mathbb{E}[\mathbf{z}] \leq \delta(d+1)$, and by Markov bound $\Pr[\mathbf{z} = 0] \geq 2/3$. If $\mathbf{z} = 0$, we know that $D^{\mathsf{intp0}}_{n,d+1}(\{\beta_i\}_{i \in [t+1]}) = Q(0) = P(\vec{x})$, which completes the proof. ∎

Lemma 9.3.5. *$L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ weakly error correctable.*

Proof. Let $m \in \mathbb{N}$ be an input length and we can assume $m \geq e_1$ since otherwise $L^{\mathsf{WH\text{-}TV}}_m$ computes the constant-zero function. Let k be the largest integer such that $e_k \leq m$ and it suffices to establish the weakly error correctability of F_k. Again, in the following we will also use n to denote n_k and i to denote i_k. Without loss of generality we can assume $m = e_k = (2 \cdot n + 1) \cdot \mathsf{sz}_n$. Let $\mu > 1$ be a sufficiently large universal constant.

Let $\tilde{f}\colon \{0,1\}^m \to \{0,1\}$ be a function that $(1 - 1/m^\mu)$-approximates $L^{\mathsf{WH\text{-}TV}}_m$. By Markov bound and recall that the definition of F_k from (9.11), for at least a $(1 - 1/m^{\mu-1})$ fraction of inputs $\vec{z} \in \mathbb{F}^{2n}$, we have

$$\Pr_{\vec{r} \in_{\mathsf{R}} \{0,1\}^{\mathsf{sz}_n}}[\tilde{f}(\vec{z}, \vec{r}) = \langle \kappa_n^{-1}(G_k(\vec{z})), \vec{r} \rangle] \geq 1 - 1/m. \tag{9.14}$$

We say that an input $\vec{z}$ is *good* if (9.14) holds. If some $\vec{z} \in \mathbb{F}^{2n}$ is good, then for every $i \in [\mathsf{sz}_n]$, we can compute the i-th bit of $\kappa_n^{-1}(G_k(\vec{z}))$ with probability at least $1 - 2^{-4n \cdot \mathsf{sz}_n}$ by the following algorithm:

1. We pick $\vec{r} \in_{\mathsf{R}} \{0,1\}^{\mathsf{sz}_n}$ and output $\tilde{f}(\vec{z}, \vec{r}) \oplus \tilde{f}(\vec{z}, \vec{r} \oplus \vec{e}_i)$, where $\vec{e}_i$ is the sz_n-bit string with 1 on the i-th bit and 0 everywhere else.
2. We repeat this procedure $\mathrm{poly}(n, \mathsf{sz}_n)$ times and take an approximate majority of the results (this can be done in AC^0, by Lemma 3.1.1).

By a union bound, we can fix the randomness used by the above algorithm, and obtain a polynomial-size non-adaptive $\mathsf{AC}^0[2]$ oracle circuit C with $\tilde{f}$ oracle gates that correctly computes $\kappa_n^{-1}(G_k(\vec{z}))$ for every good $\vec{z}$. In other words, $C^{\tilde{f}}$ computes $\kappa_n^{-1}(G_k(\vec{z}))$ on an $(1 - 1/m^{\mu-1})$ fraction of inputs $\vec{z}$. Since $G_k \colon \mathbb{F}^{2n} \to \mathbb{F}$ is a degree-$O(m)$ polynomial and μ is sufficiently large, we can use Lemma 9.3.4 to compute G_k in the worst-case by a polynomial-size non-adaptive $\mathsf{AC}^0[2]$ oracle circuit D with $\tilde{f}$ oracle gates. From the definition of F_k in (9.11), we can convert $D^{\tilde{f}}$ into another polynomial-size non-adaptive $\mathsf{AC}^0[2]$ oracle circuit $E^{\tilde{f}}$ that computes F_k. This completes the proof. ■

Next, we prove the "Moreover part" in Theorem 8.3.4, which also implies that $L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ downward self-reducible.

Lemma 9.3.6. *There are two algorithms* DSR *and* Aux *satisfying the following:*

1. Aux *takes* $m \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^m$ *as input, and output a value from* $\{0,1\}$.
2. Aux *can be implemented by a uniform* $\mathsf{AC}^0[2]$ *circuit.*
3. DSR *takes* $m \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^m$ *as input, and functions* $h_1 \colon \{0,1\}^{m-1} \to \{0,1\}$ *and* $h_2 \colon \{0,1\}^m \to \{0,1\}$ *as oracles.*
4. *For every* $m \in \mathbb{N}$, $\mathsf{DSR}_m^{L_{m-1}^{\mathsf{WH\text{-}TV}}, \mathsf{Aux}_m}$ *computes* $L_m^{\mathsf{WH\text{-}TV}}$.
5. DSR *can be implemented by a uniform non-adaptive* $\mathsf{XOR} \circ \mathsf{AND}_3$ *oracle circuit family. In more details,* DSR *first queries its oracle on some* projections *of the input* $\vec{x}$ *to obtain some intermediate values, and then applies an* $\mathsf{XOR} \circ \mathsf{AND}_3$ *circuit on those intermediate values and the input* $\vec{x}$ *to obtain the output.*

Proof. Let $m \in \mathbb{N}$ be the input length. We note that when $A_m^{\mathsf{WH\text{-}TV}}$ and $A_{m-1}^{\mathsf{WH\text{-}TV}}$ (we use $A_m^{\mathsf{WH\text{-}TV}}$ to denote the restriction of $A^{\mathsf{WH\text{-}TV}}$ on m-bit inputs; $A^{\mathsf{WH\text{-}TV}}$ is described in Algorithm 9.2) computes the same function on their prefixes, DSR_m and Aux_m can be constructed trivially.

Hence, from now on we can assume that for some $k \in \mathbb{N}$, $A_{m-1}^{\mathsf{WH\text{-}TV}}$ computes F_{k-1} and $A_m^{\mathsf{WH\text{-}TV}}$ computes F_k (*i.e.*, $m = e_k$).[9] Let n_k and i_k be such that $g_k = f_{n_k,i_k}$. There are two different cases:

1. $i_k = n_k$. In this case we have $n_{k-1} = n_k - 1$ and $i_{k-1} = 1$.
2. $i_k < n_k$. In this case we have $n_{k-1} = n_k$, and $i_{k-1} = i_k + 1$.

For simplicity, in the following we will use n to denote n_k and i to denote i_k. And our goal is to compute F_k given oracle access to F_{k-1} and Aux_m (we will define Aux_m later).

Case I: $i_k = n_k$**.** In this case, we know that $g_k = f_{n,n}$, and from (9.10), we have $G_k(\vec{x}, \vec{y}) = y_n \cdot f_{n,n}(\vec{x}) = y_n$ for every $\vec{x}, \vec{y} \in \mathbb{F}_n^n$ (note that $f_{n,n}$ is the constant-one polynomial by Item (3) of Lemma 9.2.1). Hence, $F_k(\vec{x}, \vec{z}) = \langle \kappa_n^{-1}(G_k(\vec{x})), \vec{z} \rangle$ can be computed by an $\mathsf{XOR} \circ \mathsf{AND}_2$ circuit without querying Aux_m or F_{k-1}, from which we can construct the desired DSR_m. (Aux_m does not matter here, for concreteness we set it to be the constant-zero function.)

Case II: $i_k < n_k$**.** We first note that by the definition of G_{k-1} and G_k in (9.10)

$$G_k(\vec{x}, \vec{y}) = G_{k-1}(\vec{x}, \vec{y}) + f_{n,i}(\vec{x}) \cdot y_i. \tag{9.15}$$

for every $\vec{x}, \vec{y} \in \mathbb{F}_n^n$.

We first show how to compute G_k with oracle access to G_{k-1}. From (9.15), it suffices to compute $f_{n,i}(\vec{x})$ with oracle access to G_{k-1}. Let $\vec{r}_{i+1}$ be the vector from $\mathbb{F}_n^n$ with all entries being 0 except for the $(i+1)$-th entry being 1, we have

$$f_{n,i+1}(\vec{x}) = G_{k-1}(\vec{x}, \vec{r}_{i+1}).$$

Hence, oracle access to $f_{n,i+1}(\vec{x})$ can be simulated by oracle access to G_{k+1} via the projection $\vec{x} \mapsto (\vec{x}, \vec{r}_{i+1})$. Now we can compute $f_{n,i}(\vec{x})$ by running the algorithm $\mathsf{Red}_{n,i}$ with oracle access to $f_{n,i+1}$ simulated by oracle access to G_{k-1}, and we define Aux_m to be a Boolean version[10] of $\mathsf{Term}_{n,i}$ and simulate oracle access to $\mathsf{Term}_{n,i}$ by oracle access to Aux_m. This gives us a non-adaptive $\mathsf{XOR} \circ \mathsf{AND}_2$ circuit computing $f_{n,i}(\vec{x})$ given oracle access to G_{k-1} and Aux_m.

Now, we note that $f_{n,i}(\vec{x}) \cdot y_i$ can now be computed by a non-adaptive $\mathsf{XOR} \circ \mathsf{AND}_3$ circuit given oracle access to G_k and Aux_m, and so does G_k (via (9.15)). Finally, note that a single query to G_{k-1} can be simulated by $\log |\mathbb{F}_n|$ queries to F_{k-1} and recall the definition of F_k in (9.11), we obtain

[9] For convenience, we will simply say that $A_m^{\mathsf{WH\text{-}TV}}$ computes F_k when it computes F_k on its prefix of length e_k.

[10] For example, we can apply the Hadamard-Walsh encoding to turn $\mathsf{Term}_{n,i}$ into a Boolean function, similar to (9.11).

the desired non-adaptive $\mathsf{XOR} \circ \mathsf{AND}_3$ oracle computing F_k given oracle access to F_{k-1} and Aux_m, which completes the proof. ■

Finally, we show the PSPACE-completeness of $L^{\mathsf{WH\text{-}TV}}$.

Lemma 9.3.7. *$L^{\mathsf{WH\text{-}TV}}$ is PSPACE-complete.*

Proof. We first note that $L^{\mathsf{WH\text{-}TV}} \in \mathsf{PSPACE}$ since every downward self-reducible language is in PSPACE (see, *e.g.*, [AB09, Exercise 8.9]).

Let $L \in \mathsf{PSPACE}$, and let $(A_L^{\mathsf{len}}, A_L^{\mathsf{red}})$ be the pair of algorithms in Lemma 9.2.1. The following is a polynomial-time reduction R_L from L to $L^{\mathsf{WH\text{-}TV}}$:

1. Given an input $x \in \{0,1\}^n$ for $n \in \mathbb{N}$, let $m = A_L^{\mathsf{len}}(n)$.
2. Compute $\vec{z} = A_L^{\mathsf{red}}(x)$ and let $k \in \mathbb{N}$ be such that $g_k = f_{m,1}$.
3. Let $\vec{y}$ be the vector from $\mathbb{F}_m^m$ with all entries being 0 except for the first entry being 1, and $\vec{u} \in \{0,1\}^{\mathsf{sz}_n}$ be the vector that $u_1 = 1$ and $u_j = 0$ for $j > 1$.
4. Output $L_{e_k}^{\mathsf{WH\text{-}TV}}(\vec{z}, \vec{y}, \vec{u})$.

By Lemma 9.2.1, we have $f_{m,1}(\vec{z}) = (L(x))_{\mathbb{F}_m}$. Since $L(x) \in \{0,1\}$ and we encode $\mathbb{F}_m$ as a Boolean string in $\{0,1\}^{\mathsf{sz}_m}$ via κ_m. One can see that

$$\left(\kappa_m^{-1}(f_{m,1}(\vec{z}))\right)_1 = L(x). \tag{9.16}$$

Now, by the definition of G_k in (9.10), we have that $G_k(\vec{z}, \vec{y}) = g_k(\vec{z}) = f_{m,1}(\vec{z})$. Then by the definition of F_k, Claim 9.3.2 and (9.16), we have

$$L_{e_k}^{\mathsf{WH\text{-}TV}}(\vec{z}, \vec{y}, \vec{u}) = F_k(\vec{z}, \vec{y}, \vec{u}) = \left(\kappa_m^{-1}(f_{m,1}(\vec{z}))\right)_1 = L(x).$$

Therefore, $L^{\mathsf{WH\text{-}TV}}$ is PSPACE-complete. ■

9.3.3 The Final Language L^{PSPACE}

Finally, we modify $L^{\mathsf{WH\text{-}TV}}$ into a new language L^{PSPACE} that is also paddable, via Algorithm 9.3.

In other words, A^{PSPACE} partitions the input $x \in \{0,1\}^m$ into consecutive blocks $x^{[1]}, x^{[2]}, \ldots, x^{[t]}$ of length $1, 2, 3, \ldots$ until running out of the input bits, and output $\bigoplus_{i \in [t]} L_i^{\mathsf{WH\text{-}TV}}(x^{[i]})$.

Now we are ready to prove Theorem 8.3.4.

Algorithm 9.3: Algorithm A^{PSPACE} for L^{PSPACE}

Given an input $x \in \{0,1\}^m$ for some $m \in \mathbb{N}$;
Let $i_{\mathsf{cur}} = 1$ and $\mathsf{at} = 1$;
Let $\mathsf{res} = 0$;
while $\mathsf{at} + i_{\mathsf{cur}} - 1 \leq m$ **do**
    $\mathsf{res} = \mathsf{res} \oplus L^{\mathsf{WH\text{-}TV}}_{i_{\mathsf{cur}}}(x_{[\mathsf{at},\mathsf{at}+i_{\mathsf{cur}})})$; // $\oplus$ denotes XOR, and $x_{[\mathsf{at},\mathsf{at}+i_{\mathsf{cur}})}$ denotes $(x_{\mathsf{at}}, x_{\mathsf{at}+1}, \ldots, x_{\mathsf{at}+i_{\mathsf{cur}}-1})$
    $\mathsf{at} = \mathsf{at} + i_{\mathsf{cur}}$;
    $i_{\mathsf{cur}} = i_{\mathsf{cur}} + 1$;
return res;

Proof of Theorem 8.3.4. The PSPACE-completeness of L^{PSPACE} follows from the PSPACE-completeness of $L^{\mathsf{WH\text{-}TV}}$. Given an input length $m \in \mathbb{N}_{\geq 1}$. We aim to establish the paddability from $L^{\mathsf{PSPACE}}_{m-1}$ to L^{PSPACE}_m, the downward self-reducibility from L^{PSPACE}_m to $L^{\mathsf{PSPACE}}_{m-1}$, the instance checkability of L^{PSPACE}_m, and the weak error correctability of L^{PSPACE}_m.

Let t be the largest integer such that $\binom{t+1}{2} \leq m$. We first note that if $\binom{t+1}{2} < m$, then indeed A^{PSPACE}_m and $A^{\mathsf{PSPACE}}_{m-1}$ compute the same function on their first $\binom{t+1}{2}$ input bits, and downward self-reducibility and paddability are trivial. So it suffices to consider $m = \binom{t+1}{2}$ for paddability and downward self-reducibility. We can also observe that only considering $m = \binom{t+1}{2}$ suffices for establishing instance checkability and weak error correctability as well. So from now on we assume $m = \binom{t+1}{2}$ without loss of generality. For an input $x \in \{0,1\}^*$, we use $x^{[i]}$ to denote $x_{\left[\binom{i}{2}+1, \binom{i+1}{2}+1\right)}$.

From the definition of $\mathbb{F}_k$ in (9.11) and Algorithm 9.2, we can see that $L^{\mathsf{WH\text{-}TV}}_\ell(0^\ell) = 0$ for every $\ell \in \mathbb{N}$. Hence, we have

$$L^{\mathsf{PSPACE}}_{m-1}(x) = \bigoplus_{i=1}^{t-1} L^{\mathsf{WH\text{-}TV}}_i(x^{[i]}) = L^{\mathsf{PSPACE}}_m\left(x_{\leq \binom{t}{2}} \circ 0^t\right),$$

which establishes the paddability.

To see the downward self-reducibility, we note that

$$L^{\mathsf{PSPACE}}_m(x) = \bigoplus_{i=1}^{t} L^{\mathsf{WH\text{-}TV}}_i(x^{[i]}) = L^{\mathsf{PSPACE}}_{m-1}(x_{\leq \binom{t}{2}} \circ 0^{t-1}) \oplus L^{\mathsf{WH\text{-}TV}}_t(x^{[t]}).$$

It is easy to see that oracle access to $L^{\mathsf{WH\text{-}TV}}_{t-1}$ can be simulated via oracle access to $L^{\mathsf{PSPACE}}_{m-1}$ by a projection, so the required algorithms DSR and Aux can be established using the corresponding

algorithms from Lemma 9.3.6.

Now, to see the instance-checkability, we note that for every $i \in [t]$, $L_i^{\mathsf{WH\text{-}TV}}$ can be simulated via oracle access to L_m^{PSPACE} by a projection, hence we can first run the instance checker for $L^{\mathsf{WH\text{-}TV}}$ on each of $x^{[i]}$ with the simulated oracle, returns $\perp$ if any of them returns $\perp$, and output the XOR of all the outputs otherwise. And it is straightforward to see that the instance-checker above for L^{PSPACE} can also be implemented by uniform $\mathsf{AC}^0[2]$ circuits.

Finally, we will prove that the non-adaptive $\mathsf{AC}^0[2]$ weakly error correctability follows from that of $L^{\mathsf{WH\text{-}TV}}$. Let μ be the constant from the weakly error correctability of $L^{\mathsf{WH\text{-}TV}}$ (Lemma 9.3.5) and $\tilde{f}\colon \{0,1\}^m \to \{0,1\}$ be a function that $(1-m^{-\mu})$-approximates L_m^{PSPACE}. Now we fix an $i \in [t]$, by an averaging principle, there exists $\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}$ such that

$$\Pr_{x^{[i]} \in_R \{0,1\}^i} \left[\tilde{f}(\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, x^{[i]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}) = L_m^{\mathsf{PSPACE}}(\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, x^{[i]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}) \right] \geq 1 - m^{-\mu}.$$

For notational convenience, we use $\tilde{x}^{[-i]}$ to denote $\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}$, and $\tilde{x}^{[-i]} \circ x^{[i]}$ to denote $\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, x^{[i]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}$. From the definition of L_m^{PSPACE}, the above simplifies to

$$\Pr_{x^{[i]} \in_R \{0,1\}^i} \left[\tilde{f}(\tilde{x}^{[-i]} \circ x^{[i]}) = L_i^{\mathsf{WH\text{-}TV}}(x^{[i]}) \oplus \bigoplus_{\ell \in [t] \setminus \{i\}} L_\ell^{\mathsf{WH\text{-}TV}}(\tilde{x}^{[\ell]}) \right] \geq 1 - m^{-\mu} \geq 1 - i^{-\mu}.$$

Now we define $\tilde{g}(x^{[i]}) = \tilde{f}(\tilde{x}^{[-i]} \circ x^{[i]}) \oplus \bigoplus_{\ell \in [t] \setminus \{i\}} L_\ell^{\mathsf{WH\text{-}TV}}(\tilde{x}^{[\ell]})$. By Lemma 9.3.5, there is an i^μ-size non-adaptive $\mathsf{AC}^0[2]$ circuit $C_{[i]}$ such that $C_{[i]}^{\tilde{g}}$ computes $L_i^{\mathsf{WH\text{-}TV}}$. Since $\tilde{x}^{[-i]}$ is fixed, oracle access to $\tilde{g}$ can be simulated via oracle access to $\tilde{f}$ by a projection, and there is an $O(i^\mu \cdot m)$-size non-adaptive $\mathsf{AC}^0[2]$ circuit $D_{[i]}$ such that $D_{[i]}^{\tilde{f}}$ computes $L_i^{\mathsf{WH\text{-}TV}}$.

Finally, we define a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit $E^{\tilde{f}}(x^{[1]}, \ldots, x^{[t]}) = \bigoplus_{i \in [t]} D_{[i]}^{\tilde{f}}(x^{[i]})$. From the discussions above we know that $E^{\tilde{f}}$ computes L_m^{PSPACE}, and it has size at most $m^{\mu+2}$. Setting $\tau_{\mathsf{wc2ac}} = \mu + 2$ completes the proof. ■

Chapter 10

Inverse-exponential Correlation Bounds and Extremely Rigid Matrices from a New Derandomized XOR Lemma

In this chapter we focus on proving Theorem 1.7.3 and Theorem 1.7.4 , which are restated below.

Reminder of Theorem 1.7.3. *For every constant $\varepsilon \in (0,1)$, there is a P^{NP} algorithm that on input 1^n outputs a matrix $H_n \in \mathbb{F}_2^{n \times n}$ satisfying $\mathcal{R}_{H_n}(2^{\log^{1-\varepsilon} n}) \geq (1/2 - \exp(-\log^{2/3 \cdot \varepsilon} n)) \cdot n^2$, for every sufficiently large n.*

Reminder of Theorem 1.7.4. *There is a function $f \in \mathsf{E}^{\mathsf{NP}}$ such that for every sufficiently large $n \in \mathbb{N}_{\geq 1}$ and for any $d \leq o(n/\log n)^{1/2}$, it holds that $\operatorname{corr}(f_n, d) \leq 2^{-d}$.*

Due to some technical obstacles, the proof of Theorem 1.7.4 does not give any non-trivial correlation bounds for higher degrees (*i.e.*, $d \geq \sqrt{n}$). Still, using a different proof, we manage to show a trade-off between error and degree for E^{NP} against degree-d $\mathbb{F}_2$-polynomials.

Theorem 10.0.1. *For every $\beta \in (0,1)$, there is an E^{NP} function f such that, for every sufficiently large n, it holds that $\operatorname{corr}(f, n^\beta/\log n) \leq \exp(-\Omega(n^{\frac{2}{3}(1-\beta)}))$.*

Contents

Notation. Throughout this chapter, we will always use Boolean functions to denote a function from $\{0,1\}^*$ to $\{-1,1\}$, where -1 and 1 are interpreted as True and False, respectively. This choice will be particularly convenient for studying and stating correlation bounds or average-case lower bounds. For two functions $f, g\colon \{0,1\}^n \to \{-1,1\}$, we will use $\langle f, g\rangle$ to denote their inner product $\mathbb{E}_{x\in\{0,1\}^n}[f(x)\cdot g(x)]$.[1]

For a collection of functions $\mathcal{F}$, we always use $\mathcal{F}_n$ to denote the subset of $\mathcal{F}$ consisting of n-bit functions from $\mathcal{F}$. We will also need to define $\mathsf{Sum}\circ\mathcal{F}$-functions: a $\mathsf{Sum}\circ\mathcal{F}$-function $C\colon \{0,1\}^n \to \mathbb{R}$ can be written as $C(x) = \sum_{i=1}^{\ell} \alpha_i \cdot C_i(x)$, where each α_i is a real, and each $C_i(x)$ is an $\mathcal{F}_n$-function. Here ℓ is called the *sparsity* of C. We also use $\mathsf{complexity}(C)$ to denote $\max(\ell, \sum_{i=1}^{\ell} |\alpha_i|)$.

For every real $p \geq 1$, recall that the ℓ_p-norm of f is defined as $\|f\|_p := (\mathbb{E}_{x\leftarrow U_n}|f(x)|^p)^{1/p}$, and the ℓ_∞-norm of f is defined as $\|f\|_\infty := \max_{x\in\{0,1\}^n}|f(x)|$. For $p, q \in \mathbb{R}_{\geq 1}\cup\{\infty\}$, we say that p and q are *Hölder conjugates* of each other, if it holds that $1/p + 1/q = 1$.[2]

10.1 Techniques: A New Derandomized XOR Lemma

Our new results of this chapter are all proved by a new derandomized XOR lemma based on approximate linear sums. Before formally stating and discussing our new derandomized XOR lemma, it is instructive to review the XOR Lemma in [CLW20] (Lemma 1.5.10), and why it cannot be used to prove the strong correlation bounds as in Theorem 1.7.4.

10.1.1 The XOR lemma in [CLW20] and Its Disadvantage

Formally, following Levin's proof of Yao's XOR Lemma [Lev87, GNW11], [CLW20] proved the following lemma.[3]

Lemma 10.1.1 ([Lev87] and [CLW20, Lemma 3.8]). *Let $\mathcal{F}$ be a collection of functions closed under negation and restriction. For $n \in \mathbb{N}_{\geq 1}$, $\delta, \varepsilon \in (0,1)$ and every function $f\colon \{0,1\}^n \to \{-1,1\}$, if*

$$\langle f, C\rangle < (1-\delta)$$

for every $\mathsf{Sum}\circ\mathcal{F}_n$-function C such that $\mathsf{complexity}(C) \leq 10\cdot n/\varepsilon^2$ and $\|C\|_\infty \leq 1$, then $\langle f^{\oplus k}, C\rangle \leq$

[1] See Section 10.3 for more details on notation used in this chapter.
[2] We use the convention that $1/\infty = 0$, so it can be the case that $p = 1$ and $q = \infty$ and vice versa.
[3] The below is a reformulated and slightly more general version of Lemma 1.5.10.

$(1-\delta)^k + \varepsilon/\delta$ *for any* $f \in \mathcal{F}$.[4]

That is, given a function f which cannot be weakly approximated (say, 0.99-approximated) by $\mathsf{Sum} \circ \mathcal{F}_n$-functions, one can show that $f^{\oplus k}$ is strongly average-case hard for $\mathcal{F}$. The advantage of Lemma 10.1.1 above over other versions of XOR lemmas [Yao82, Imp95, GNW11, IW97] is that it adds *minimal* computational overhead from the target class $\mathcal{F}$ to the starting class $\mathsf{Sum} \circ \mathcal{F}$, which enables [CR20, CLW20] to apply Williams' algorithmic method to obtain the required hardness against $\mathsf{Sum} \circ \mathcal{F}$ *using algorithms only for* $\mathcal{F}$.

Still, to obtain a $2^{-\Omega(\sqrt{n})}$ correlation bound using Lemma 10.1.1 with a constant δ (say, $\delta = 0.01$), one has to set $\varepsilon \approx 2^{-\sqrt{n}}$ and $k \approx \sqrt{n}$. Applying the algorithmic method, [CLW20] indeed managed to prove that there is an E^{NP}-computable function $f\colon \{0,1\}^n \to \{-1,1\}$ which cannot be $(1-\delta)$-approximated by linear sums of $2^{\sqrt{n}}$ many $\mathbb{F}_2$-polynomials of degree at most $\sqrt{n}$. Applying Lemma 10.1.1, one can obtain a function $\mathsf{Amp}^f := f^{\oplus k}$ such that $\mathrm{corr}(\mathsf{Amp}^f, \sqrt{n}) \leq 2^{-\Omega(\sqrt{n})}$. However, this is not enough, since Amp^f in fact takes $m = \Theta(n^{1.5})$ bits of input, the correlation bounds *deteriorate* to $\mathrm{corr}(\mathsf{Amp}^f, m^{1/3}) \leq 2^{-\Omega(m^{1/3})}$.

10.1.2 A New Derandomized XOR Lemma

To further improve the correlation bounds in [CLW20], we manage to prove a derandomized XOR lemma based on approximate linear sums. Roughly speaking, we construct a *pseudorandom instances generator* $\mathcal{G}\colon \{0,1\}^m \to \{0,1\}^{nk}$ that takes a seed of length $m = \widetilde{O}(n)$, and produces k instances to the function f. We can show that our generator $\mathcal{G}$ is pseudorandom enough to *fool the proof of XOR lemma*, and establish the following new derandomized XOR lemma based on approximate linear sums.

Lemma 10.1.2 ((Informal)). *Let* $n \in \mathbb{N}_{\geq 1}$, $\varepsilon \in (0,1)$, $k = \Theta(\log \varepsilon^{-1})$ *and* $\mathcal{F} = \bigcup_{n \in \mathbb{N}_{\geq 1}} \mathcal{F}_n$ *be a function collection satisfying some technical conditions*[5]. *There is a polynomial-time generator* $\mathcal{G}\colon \{0,1\}^m \to \{0,1\}^{nk}$ *with* $m = \widetilde{O}(n)$ *such that, for every function* $f\colon \{0,1\}^n \to \{-1,1\}$ *that cannot be weakly approximated by* $\mathsf{Sum} \circ \mathcal{F}$*-functions of complexity at most* $O(n/\varepsilon^2)$, *then* $\langle f^{\oplus k} \circ \mathcal{G}, C \rangle \leq \varepsilon^{\Omega(1)}$ *holds for every* $C \in \mathcal{F}_m$.

The proof of our new derandomized XOR lemma is based on a "non-Boolean" generalization of the concept of hardcore sets, which is thoroughly discussed in Section 10.2.1. Such a generalization can also be used to give a completely different and duality-based proof of Lemma 10.1.1.

[4] The function $f^{\oplus k}\colon (\{0,1\}^n)^k \to \{-1,1\}$ is defined as $f^{\oplus k}(x_1, \ldots, x_k) = \prod_{i=1}^k f(x_i)$.
[5] See Lemma 10.4.1 for the details.

Combing Lemma 10.1.2 with the algorithmic method developed in [Wil10, Wil11, CW19, CR20, CLW20] for proving hardness against linear sums of collection of functions which admit efficient circuit-analysis algorithms, we can then obtain our improved correlation bounds against $\mathbb{F}_2$-polynomials and construction of extremely rigid matrices.

10.2 Overview of Proofs

In this section we give an overview of the proof ideas behind our new results. In Section 10.2.1 we define and discuss the key concept, $(\varepsilon, \delta)_{\ell_p}$-witnesses, behind our proofs. In Section 10.2.2 we give a duality-based new proof for Lemma 10.1.1, which can be seen as a warm-up for later proofs. In Section 10.2.3 we discuss the intuitions behind our proof of the new derandomization XOR lemma. In Section 10.2.4 we give some intuitions behind our specific constructions of restrictable generators.

10.2.1 The Dual Witness to Inapproximability by Linear Sums

The first ingredient of our proof is a dual witness for inapproximability of a function f by $\mathsf{Sum} \circ \mathcal{F}_n$-functions.

Definition 10.2.1. *Let $f\colon \{0,1\}^n \to \{-1,1\}$ be a function, let $p \in \mathbb{R}_{\geq 1} \cup \{\infty\}$, and let $\delta > 0, \varepsilon > 0$ be two reals. We say that a function $h\colon \{0,1\}^n \to \mathbb{R}$ is a $(\delta,\varepsilon)_{\ell_p}$-witness for f against $\mathcal{F}_n$-functions, if $\|h\|_p \leq 1$, $\|h\|_1 \leq 1-\delta$ and $|\langle C, f-h\rangle| \leq \varepsilon$ for every $C \in \mathcal{F}_n$.*

We will often consider the setting where ε is very small and δ is a small constant (*e.g.*, $\varepsilon \leq n^{-\omega(1)}$ and $\delta = 0.01$). That is, a $(\delta,\varepsilon)_{\ell_p}$-witness h for f against $\mathcal{F}_n$-functions can be used to *perturb* f so that the resulting function $f-h$ is *extremely hard* for $\mathcal{F}_n$-functions.

If additionally we can also make $1-\delta$ very small (instead of being a constant), then we would immediately obtain strong average-case lower bounds for f against $\mathcal{F}_n$. Formally, we have the following remark.

Remark 10.2.2. *If there is a $(\delta,\varepsilon)_{\ell_p}$-witness for f against $\mathcal{F}_n$-functions, then $|\langle C, f\rangle| \leq \varepsilon + (1-\delta)$ for every $C \in \mathcal{F}_n$.*

By Remark 10.2.2, to show that a function f is strongly average-case hard against $\mathcal{F}_n$-functions, it suffices to construct a witness h with very small ε and ℓ_1-norm. This will be the approach adopted in the proofs of this section.

$(\delta, \varepsilon)_{\ell_p}$-witnesses and Hardcore Sets

One may notice that the witness defined in Definition 10.2.1 appears to be very similar to the concept of *hardcore sets*[6], which are studied extensively in the complexity theory [Imp95, Hol05, RTTV08, TTV09, BHK09]. The following discussions in this subsection are aimed to provide more intuitions on Definition 10.2.1 and its connections to hardcore sets, and may be skipped without affecting the understanding of the proofs in this chapter.

From $(\delta, \varepsilon)_{\ell_\infty}$-witnesses to (Boolean) hardcore sets. We remark quickly that when $p = \infty$, a $(\delta, \varepsilon)_{\ell_\infty}$-witness h is *essentially equivalent* to an $\Omega(\delta)$-dense hardcore set of f. Let $P_{\mathsf{core}} := f - h$, since $\|h\|_\infty \leq 1$ and f is Boolean, it immediately implies that $P_{\mathsf{core}}(x)$ either has the same sign with $f(x)$ or is zero. We can then construct a function f_{core} by setting each $f_{\mathsf{core}}(x) = f(x)$ independently with probability $|P_{\mathsf{core}}(x)|$, and 0 otherwise.[7] Note that for every x, $\mathbb{E}[f_{\mathsf{core}}(x)] = P_{\mathsf{core}}(x)$.

We can then obtain a hardcore set of f from f_{core} since with high probability: (1) Since $\|P_{\mathsf{core}}\|_1 \geq \|f\|_1 - \|h\|_1 \geq \delta$, at least an $\Omega(\delta)$-fraction of $f_{\mathsf{core}}(x)$ are non-zero (either -1 or 1), those are the inputs in our hardcore set. (2) $\langle f_{\mathsf{core}}, C\rangle$ will be very close to $\langle P_{\mathsf{core}}, C\rangle$ (by a Chernoff bound), which is at most ε. This means f is extremely hard on this hardcore set.

$(\delta, \varepsilon)_{\ell_p}$-witnesses as "non-Boolean" hardcore sets. When $p \neq \infty$, a $(\delta, \varepsilon)_{\ell_p}$-witness h (or more accurately, the function $P^{\ell_p}_{\mathsf{core}} := f - h$ obtained by perturbing f with h) can be thought of as a non-Boolean hardcore set, in the following sense: (1) $P^{\ell_p}_{\mathsf{core}}$ has ℓ_1-norm at least $\|f\|_1 - \|h\|_1 \geq \delta$, so $P^{\ell_p}_{\mathsf{core}}$ is still of "δ-density" and (2) $P^{\ell_p}_{\mathsf{core}}$ is still extremely hard against for $\mathcal{F}$-functions.

We cannot construct from $P^{\ell_p}_{\mathsf{core}}$ a Boolean hardcore $f^{\ell_p}_{\mathsf{core}}$ anymore, since many points in $P^{\ell_p}_{\mathsf{core}}$ can have very large absolute values. Indeed, the norm p controls the *Booleanness* of $f^{\ell_p}_{\mathsf{core}}$: the larger the p is, the closer the $f^{\ell_p}_{\mathsf{core}}$ is to Boolean functions.

We also remark that another way to interpret $P^{\ell_p}_{\mathsf{core}}$ is that it corresponds to a certain hardcore pseudodistribution instead of a hardcore distribution.[8]

From Inapproximability by Linear sums to $(\delta, \varepsilon)_{\ell_p}$-Witnesses

The following "inapproximability-to-witness" lemma shows that we can construct a non-trivial witness for f against $\mathcal{F}_n$-function from the weak inapproximability of f by $\mathsf{Sum} \circ \mathcal{F}_n$-function.

[6] Roughly speaking, a hardcore set H for a function f is a subset of $\{0,1\}^n$ with at least $\delta \cdot 2^n$ elements such that f is strongly average-case hard to compute by a certain class of functions with respect to the uniform distribution over H.

[7] If $\|P_{\mathsf{core}}\|_\infty > 1$, we can scale P_{core} by $1/2$, this only reduces its density by a factor of 2.

[8] see [BCG20, CL20] for more discussions on recent works in derandomization using pseudodistributions.

This lemma serves as the starting point for our derandomized XOR lemma. Its proof can be found in Section 10.4.1.

Lemma 10.2.3. *Let $n \in \mathbb{N}_{\geq 1}$, and let $\mathcal{F}_n$ be a collection of n-input functions that is closed under negation. Let $p, q \in \mathbb{R}_{\geq 1} \cup \{\infty\}$ be such that p and q are Hölder conjugates of each other. For every function $f\colon \{0,1\}^n \to \{-1,1\}$ and $\delta, \varepsilon > 0$, if we have*

$$\langle f, C \rangle < (1 - \delta)$$

for every $\mathsf{Sum} \circ \mathcal{F}_n$-function C such that $\mathsf{complexity}(C) \leq 10 \cdot n/\varepsilon^2$ and $\|C\|_q \leq 1$, then there is a $(\delta, \varepsilon)_{\ell_p}$-witness h for f against $\mathcal{F}_n$-functions.

Moreover, for the case $p = \infty$ and $q = 1$, the condition can be replaced by that for every $\mathsf{MAJ} \circ \mathcal{F}$-function C with top-sparsity bounded by $10n/\varepsilon^2$, it holds that $\langle f, C \rangle < 1 - 2\delta$.

Remark 10.2.4. *By the discussions in Section 10.2.1, when $(p, q) = (\infty, 1)$, a $(\delta, \varepsilon)_{\ell_p}$-witness immediately implies the existence of an $\Omega(\delta)$-dense hardcore set of f against $\mathcal{F}_n$-functions. Thus, the moreover part of Lemma 10.2.3 is equivalent to Impagliazzo's Hardcore Lemma.*

10.2.2 A New Proof of the Original XOR Lemma

As a warm-up, in this section we will first give a new proof of Levin's XOR Lemma [Lev87], reformulated by [CLW20, Lemma 3.8].

Reminder of Lemma 10.1.1. *Let $\mathcal{F}$ be a collection of functions closed under negation and restriction. For $n \in \mathbb{N}_{\geq 1}$, $\delta, \varepsilon \in (0,1)$ and every function $f\colon \{0,1\}^n \to \{-1,1\}$, if*

$$\langle f, C \rangle < (1 - \delta)$$

for every $\mathsf{Sum} \circ \mathcal{F}_n$-function C such that $\mathsf{complexity}(C) \leq 10 \cdot n/\varepsilon^2$ and $\|C\|_\infty \leq 1$, then $\langle f^{\oplus k}, C \rangle \leq (1 - \delta)^k + \varepsilon/\delta$ for any $f \in \mathcal{F}$.

ε-Indistinguishability. For two functions $f, g\colon \{0,1\}^n \to \mathbb{R}$ and a parameter $\varepsilon > 0$, we say that f and g are *ε-indistinguishable* by $\mathcal{F}_n$-functions if $|\langle f - g, C \rangle| \leq \varepsilon$ for every $C \in \mathcal{F}_n$.

Proof of Lemma 10.1.1. Applying Lemma 10.2.3 with $(p, q) = (1, \infty)$, the condition in the lemma implies that there is a $(\delta, \varepsilon)_{\ell_1}$-witness h for f against $\mathcal{F}_n$-functions. That is:

1. f and h are ε-indistinguishable by $\mathcal{F}$-functions.
2. h has ℓ_1-norm at most $(1-\delta)$, which is slightly less than 1.

Proof plan. Our proof will be duality-based. That is, to show $f^{\oplus k}(x)$ is strongly average-case hard, we will show that the function $h^{\oplus k}$ is a sufficient witness to apply Remark 10.2.2. That is, we want to show the following:

1. (**Indistinguishability.**) $f^{\oplus k}$ is (ε/δ)-indistinguishable from $h^{\oplus k}$ by $\mathcal{F}$-functions.
2. (**Bounded ℓ_1-norm.**) $h^{\oplus k}$ has ℓ_1-norm bounded by $(1-\delta)^k$.

The second item above is easy to establish, since $\|h^{\oplus k}\|_1 = \|h\|_1^k \leq (1-\delta)^k$. Hence it only remains to show the first item.

A hybrid argument. We will show the indistinguishability between $f^{\oplus k}$ and $h^{\oplus k}$ by a hybrid argument. For every $i \in \{0,\ldots,k\}$, we define a hybrid function $H_i^{h,f} := h^{\oplus i} \otimes f^{\oplus k-i}$. That is, for every $r = (r_1,\ldots,r_k) \in (\{0,1\}^n)^k$, we have

$$H_i^{h,f}(r) = \prod_{j=1}^{i} h(r_j) \cdot \prod_{j=i+1}^{k} f(r_j).$$

Note that $H_0^{h,f}$ and $H_k^{h,f}$ are just $f^{\oplus k}$ and $h^{\oplus k}$, respectively. We will show that $H_0^{h,f}$ and $H_k^{h,f}$ are indistinguishable by showing that for every $i \in \{0,\ldots,k-1\}$, the two consecutive functions $H_i^{h,f}$ and $H_{i+1}^{h,f}$ are indistinguishable. Formally, we have the following claim.

Claim 10.2.5. *For every $i \in [k]$ and every $C \in \mathcal{F}_{nk}$, $|\langle H_{i-1}^{h,f} - H_i^{h,f}, C\rangle| \leq \varepsilon \cdot (1-\delta)^{i-1}$.*

We will prove Claim 10.2.5 later, but assuming it for now, for every $C \in \mathcal{F}_{nk}$, we have

$$\begin{aligned}
|\langle f^{\oplus k}, C\rangle| &\leq |\langle h^{\oplus k}, C\rangle| + |\langle f^{\oplus k} - h^{\oplus k}, C\rangle| \\
&\leq \|h\|_1^k + \sum_{i=1}^{k} \left|\langle H_{i-1}^{h,f} - H_i^{h,f}, C\rangle\right| && (\|C\|_\infty = 1) \\
&\leq (1-\delta)^k + \varepsilon \cdot \sum_{i=0}^{k-1} (1-\delta)^i && (\text{Claim 10.2.5 and } \|h\|_1 \leq 1-\delta) \\
&\leq (1-\delta)^k + \varepsilon/\delta. \quad \blacksquare
\end{aligned}$$

Finally, we prove Claim 10.2.5.

Proof of Claim 10.2.5. From the definition of $H_{i-1}^{h,f}$ and $H_i^{h,f}$, we have

$$\langle H_{i-1}^{h,f} - H_i^{h,f}, C\rangle = \mathop{\mathbb{E}}_{r\leftarrow U_{nk}}\left[C(r_1,\ldots,r_k)\cdot\prod_{j=1}^{i-1}h(r_j)\cdot\prod_{j=i}^{k}f(r_j)\right] - \mathop{\mathbb{E}}_{r\leftarrow U_{nk}}\left[C(r_1,\ldots,r_k)\cdot\prod_{j=1}^{i}h(r_j)\cdot\prod_{j=i+1}^{k}f(r_j)\right]. \tag{10.1}$$

We use r_{-i} to denote $(r_1,\ldots,r_{i-1},r_{i+1},\ldots,r_k)\in(\{0,1\}^n)^{k-1}$, so that $r\in\{0,1\}^{nk}$ can be decomposed into r_i and r_{-i}. Organizing the right side of (10.1), we have

$$\langle H_{i-1}^{h,f} - H_i^{h,f}, C\rangle = \mathop{\mathbb{E}}_{r_{-i}\leftarrow U_{n(k-1)}}\prod_{j=1}^{i-1}h(r_j)\cdot\prod_{j=i+1}^{k}f(r_j)\cdot\mathop{\mathbb{E}}_{r_i\leftarrow U_n}[C(r_1,\ldots,r_k)\cdot(f_i(r_i)-h_i(r_i))]. \tag{10.2}$$

To further bound (10.2), for each $r_{-i}\in(\{0,1\}^n)^{k-1}$, we define a function $D^{r_{-i}}\colon\{0,1\}^n\to\{-1,1\}$ as

$$D^{r_{-i}}(x) := C(r_1,\ldots,r_{i-1},x,r_{i+1},\ldots,r_k).$$

It follows that $D^{r_{-i}}\in\mathcal{F}_n$ since $\mathcal{F}$ is closed under restriction. Therefore, since h is a $(\delta,\varepsilon)_{\ell_1}$-witness for f against $\mathcal{F}_n$-functions, we have

$$|\langle f-h, D^{r_{-i}}\rangle|\le\varepsilon. \tag{10.3}$$

Plugging in (10.2), we have

$$\begin{aligned}
|\langle H_{i-1}^{h,f} - H_i^{h,f}, C\rangle| &= \left|\mathop{\mathbb{E}}_{r_{-i}\leftarrow U_{n(k-1)}}\prod_{j=1}^{i-1}h(r_j)\cdot\prod_{j=i+1}^{k}f(r_j)\cdot\langle f-h, D^{r_{-i}}\rangle\right| && (10.4)\\
&\le \varepsilon\cdot\mathop{\mathbb{E}}_{r_{-i}\leftarrow U_{n(k-1)}}\prod_{j=1}^{i-1}|h(r_j)| && \text{(by (10.3) and } \|f\|_\infty=1)\\
&\le \varepsilon\cdot\prod_{j=1}^{i-1}\mathop{\mathbb{E}}_{r_j\leftarrow U_n}|h(r_j)| && (10.5)\\
&\le \varepsilon\cdot(1-\delta)^{i-1}. && (\|h\|_1\le 1-\delta)
\end{aligned}$$

■

10.2.3 New Derandomized XOR Lemma

Now we turn to the proof intuitions behind the proof of our new derandomized XOR lemma. We begin by introducing the concept of pseudorandom instance generator and some useful notation.

Pseudorandom instance generators and notation. For convenience, let $\mathcal{F} = \bigcup_{n\in\mathbb{N}_{\geq 1}} \mathcal{F}_n$ be a collection of functions closed under negation and restriction. We will always use $f\colon \{0,1\}^n \to \{-1,1\}$ to denote a weakly average-case hard function on which we will apply the hardness amplification, and we will always use n to denote the input length of f.

The idea is to use a *pseudorandom instance generator* $\mathcal{G}\colon \{0,1\}^m \to (\{0,1\}^n)^k$ (m is much less than nk) to generate inputs to the function $f^{\oplus k}$, similar to the original derandomized XOR lemma in [IW97]. That is, by directly composing the generator $\mathcal{G}$ and $f^{\oplus k}$, one obtains a function $\mathsf{Amp}^f := f^{\oplus k} \circ \mathcal{G}$, which has input length m instead of nk.

High-level idea. Our goal would be to construct the desired pseudorandom instance generator $\mathcal{G}$ such that a similar argument as in the proof of Lemma 10.1.1 still goes through. That is, we wish to show that $\mathsf{Amp}^h := h^{\oplus k} \circ \mathcal{G}$ is a dual-witness showing that $\mathsf{Amp}^f = f^{\oplus k} \circ \mathcal{G}$ is strongly average-case hard against $\mathcal{F}$-functions (see Remark 10.2.2). Therefore, we need to establish the following two statements:

1. (**Indistinguishability.**) Amp^f and Amp^h are $(\varepsilon^{\Omega(1)})$-indistinguishable by $\mathcal{F}_m$-functions.
2. (**Bounded ℓ_1-norm.**) Amp^h has ℓ_1-norm at most $(1-\delta)^k$.

In the following, we show how to construct a generator meeting the two requirements above. We will omit some technical details and focus on the key insights in our approach.

Establishing the Indistinguishability

First, our constructed $\mathcal{G}$ needs to ensure that $\mathsf{Amp}^f = f^{\oplus k} \circ \mathcal{G}$ and $\mathsf{Amp}^h = h^{\oplus k} \circ \mathcal{G}$ are indistinguishable. In the following we will try to adapt the proof of Lemma 10.1.1, and figure out along the way that which properties $\mathcal{G}$ has to satisfy for the adaption to go through.

A new hybrid argument and the difficulty. Again we will try to apply a hybrid argument, recall that we have defined the hybrid functions $H_i^{h,f} = h^{\oplus i} \otimes f^{\oplus k-i}$ in the proof of Lemma 10.1.1. To simplify notation, we let $\mathcal{G}_i^{h,f} = H_i^{h,f} \circ \mathcal{G}$ to denote our new hybrid functions. Note that $\mathcal{G}_0^{h,f} = \mathsf{Amp}^f$ and $\mathcal{G}_k^{h,f} = \mathsf{Amp}^h$.

Fix $i \in [k]$, our goal is to show that $|\langle \mathcal{G}_{i-1}^{h,f} - \mathcal{G}_i^{h,f}, C\rangle|$ is small for every $C \in \mathcal{F}_m$. Recall in the proof of Lemma 10.1.1, an analogous bound (Claim 10.2.5) is proved by considering the following

equalities:

$$|\langle H_{i-1}^{h,f} - H_i^{h,f}, C\rangle| = \left| \mathop{\mathbb{E}}_{r\leftarrow U_{nk}} \prod_{j=1}^{i-1} h(r_j) \cdot \prod_{j=i+1}^{k} f(r_j) \cdot [C(r_1,\ldots,r_k) \cdot (f_i(r_i) - h_i(r_i))] \right|$$
$$= \left| \mathop{\mathbb{E}}_{r_{-i}\leftarrow U_{n(k-1)}} \prod_{j=1}^{i-1} h(r_j) \cdot \prod_{j=i+1}^{k} f(r_j) \cdot \langle f - h, D^{r_{-i}}\rangle \right|, \tag{10.6}$$

where $D^{r_{-i}} : \{0,1\}^n \to \{-1,1\}$ is obtained by restricting the inputs $r_1,\ldots,r_{i-1},\ldots,r_{i+1},\ldots,r_k$ to C by r_{-i}. Since $\mathcal{F}$ is closed under restriction, it follows that $D^{r_i} \in \mathcal{F}$, and we can then apply the bound on $|\langle f - h, D^{r_{-i}}\rangle|$, since f and h are indistinguishable by $\mathcal{F}_n$-functions.

The key intuition in the proof above is that, since the i-th input r_i in $r = (r_1,\ldots,r_k)$ is *completely independent* to the other $k-1$ inputs, one can *fix the other inputs* first and then apply the indistinguishability between f and h to replace f by h on r_i.

Switching to our new setting. For a seed $r \in \{0,1\}^m$, we use $\tilde{r}_i$ to denote $\mathcal{G}(r)_i$ for simplicity. Then we can still write

$$|\langle \mathcal{G}_{i-1}^{h,f} - \mathcal{G}_i^{h,f}, C\rangle| = \mathop{\mathbb{E}}_{r\leftarrow U_m} \prod_{j=1}^{i-1} h(\tilde{r}_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j) \cdot [C(r) \cdot (f_i(\tilde{r}_i) - h_i(\tilde{r}_i))].$$

But since now all the $\tilde{r}_i$ are no longer independent (since they are generated from a seed r with length m much less than nk). We cannot proceed as (10.6) anymore. That is, if we try to fix $\tilde{r}_{-i}$ first, then it *may even completely fix* the value of $\tilde{r}_i$, and we can no longer obtain a similar function $D^{\tilde{r}_{-i}}$ on $\tilde{r}_i$.

Partial independence and $\mathcal{F}$-restrictable generators. Inspired by the famous Nisan-wigderson generator [NW94], and similar to the proof of the original derandomized XOR lemma in [IW97]. Our idea to resolve the issue above is to design the generator $\mathcal{G}$ in a way that, for each i, some part of the seed r directly corresponds to $\tilde{r}_i$, yet for all other bits, they are *almost independent* to $\tilde{r}_i$.

More formally, we want a mapping $T_i\colon \{0,1\}^n \times \{0,1\}^{m-n} \to \{0,1\}^m$, such that: (1) T_i is a bijection and (2) $\mathcal{G}(T_i(x,\alpha))_i = x$ for all $(x,\alpha) \in \{0,1\}^n \times \{0,1\}^{m-n}$. That is, the first condition says that T_i is just a "reorganization" of the input space $\{0,1\}^m$ while the second condition says that x corresponds directly to $\tilde{r}_i$.

Using the mapping T_i, we can write

$$|\langle \mathcal{G}_{i-1}^{h,f} - \mathcal{G}_i^{h,f}, C\rangle| = \left| \mathop{\mathbb{E}}_{\substack{(x,\alpha)\leftarrow U_m \\ r=T_i(x,\alpha)}} \prod_{j=1}^{i-1} h(\tilde{r}_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j) \cdot [C(r) \cdot (f_i(x) - h_i(x))] \right|. \tag{10.7}$$

For $\alpha \in \{0,1\}^{m-n}$ and $x \in \{0,1\}^n$, we let

$$D^\alpha(x) := \prod_{j=1}^{i-1} h(\tilde{r}_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j) \cdot C(T_i(x,\alpha)),$$

where the $\tilde{r}_j$ above corresponds to $\mathcal{G}(T_i(x,\alpha))_j$ ($\mathcal{G}(r)_j$ if $r = T_i(x,\alpha)$).

Plugging in the above into (10.7), it follows that

$$|\langle \mathcal{G}_{i-1}^{h,f} - \mathcal{G}_i^{h,f}, C\rangle| = \mathop{\mathbb{E}}_{\alpha\leftarrow U_{m-n}} \langle f - h, D^\alpha\rangle. \tag{10.8}$$

Therefore, if the function D^α above still belongs to $\mathcal{F}_n$, we can then apply the indistinguishability between f and h by $\mathcal{F}_n$-functions, and proceed just as in the proof of Lemma 10.1.1. This motivates our definition of $\mathcal{F}$-restrictable generator as follows.

Definition 10.2.6 ($\mathcal{F}$-restrictable generators)**.** *Given a function collection $\mathcal{F}$ and $n \in \mathbb{N}_{\geq 1}$, a generator $\mathcal{G}\colon \{0,1\}^m \to \{0,1\}^{nk}$ is called $\mathcal{F}$-restrictable, if there are k embedding functions $T_1, \ldots, T_k\colon \{0,1\}^n \times \{0,1\}^{m-n} \to \{0,1\}^m$ such that the following hold:*

1. *All the T_i are bijections.*
2. *For every $i \in [k]$ and $(x,\alpha) \in \{0,1\}^n \times \{0,1\}^{m-n}$, $\mathcal{G}(T_i(x,\alpha))_i = x$. That is, $T_i(x,\alpha) \in \{0,1\}^m$ is a seed to $\mathcal{G}$ which fixes the i-th instance of $\mathcal{G}(T_i(x,\alpha))$ to be x.*
3. *For every $\mathcal{F}_m$-function $C\colon \{0,1\}^m \to \{-1,1\}$, $i \in [k]$, $\alpha \in \{0,1\}^{m-n}$ and functions $u_1, \ldots, u_k\colon \{0,1\}^n \to \{1,-1\}$, the function $D(x) := C(T_i(x,\alpha)) \cdot \prod_{j\in[k]\setminus\{i\}} u_j(\mathcal{G}(T_i(x,\alpha))_j)$ belongs to $\mathcal{F}_n$.*

In the proof of the original derandomized XOR Lemma by [IW97], a restrictable generator for small circuits was constructed by directly adapting the Nisan-Wigderson generator [NW94], which is unfortunately not enough for our applications. So instead, we design two restrictable generators which are tailored to $\mathbb{F}_2$-polynomials and low-rank matrices.

To prove Theorem 1.7.4, we carefully construct a "star-like" $\mathcal{F}$-restrictable generator for a function collection $\mathcal{F}$ which contains low-degree polynomials as a subset. And similarly, we design a "bi-coloring" restrictable generator for low-rank matrices to prove Theorem 1.7.3. We will

overview the high-level ideas behind these constructions in Section 10.2.4.

If the function h is Boolean as well, then Item (3) of Definition 10.2.6 tells us $D^\alpha \in \mathcal{F}_n$ and we can then bound (10.3). However, the function h may be non-Boolean, and in fact, it may be even unbounded. This causes D^α to also be a non-Boolean function, and we can not directly apply the third condition in Definition 10.2.6.

Smooth witnesses come to help. We first observe that the aforementioned issue can be resolved if h is smooth in a certain sense. Suppose h is $[-1,1]$-valued (that is, $\|h\|_\infty \le 1$), we can view $D^\alpha(x)$ as a probabilistic $\mathcal{F}$-function and apply a similar argument.

In more details, we sample $i-1$ independent functions $u_1 \dots u_{i-1} \colon \{0,1\}^n \to \{-1,1\}$ from certain distributions, in a way that for every $x \in \{0,1\}^n$, letting $r = T_i(x,\alpha)$, we have

$$\mathop{\mathbb{E}}_{u_j}[u_j(\tilde{r}_j)] = h(\tilde{r}_j) \quad \text{for every } j \in [i-1]. \tag{10.9}$$

We then set

$$D^{\alpha;u_1,\dots,u_{i-1}}(x) := \prod_{j=1}^{i-1} u_j(\tilde{r}_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j) \cdot C(T_i(x,\alpha)).$$

Recall that we have set $r = T_i(x,\alpha)$, and hence $\tilde{r}_j$ above corresponds to $\mathcal{G}(T_i(x,\alpha))_j$.

By Item (3) of Definition 10.2.6, $D^{\alpha;u_1,\dots,u_{i-1}}$ is an $\mathcal{F}_n$-function for every possible $(i-1)$-tuples $(u_1,\dots,u_{i-1})$. Hence, we have

$$\begin{aligned}
|\langle \mathcal{G}^{h,f}_{i-1} - \mathcal{G}^{h,f}_i, C\rangle| &= \mathop{\mathbb{E}}_{\alpha \leftarrow U_{m-n}} \langle f-h, D^\alpha\rangle. \\
&= \mathop{\mathbb{E}}_{\alpha \leftarrow U_{m-n}} \mathop{\mathbb{E}}_{u_1 \dots u_{i-1}} \langle f-h, D^{\alpha;u_1\dots u_{i-1}}\rangle && \text{(by (10.9))} \\
&\le \varepsilon.
\end{aligned}$$

When $\|h\|_\infty \le M$, a simple scaling argument (replace h by h/M) can be used to show that $|\langle \mathcal{G}^{h,f}_{i-1} - \mathcal{G}^{h,f}_i, C\rangle| \le M^{i-1} \cdot \varepsilon$. We refer to Lemma 10.4.7 for a formal (and more general) proof of the argument above.

Norms and Smoothness of the witnesses. Setting $(p,q) = (\infty,1)$, Lemma 10.2.3 shows that if we can show that f is weakly inapproximable by $\mathsf{Sum} \circ \mathcal{F}_n$-functions of unit ℓ_1-norm, then we would obtain a $(\delta,\varepsilon)_{\ell_\infty}$-witness h. And one can then proceed to prove our derandomized XOR lemma.

Unfortunately, due to some inherent limitation of the polynomial method, using the algorithmic method, it seems very hard to prove there is a function $f \in \mathsf{E}^{\mathsf{NP}}$ which is weakly inapprox-

imable by $\mathsf{Sum} \circ \mathcal{F}_n$-functions of unit ℓ_1-norm.[9]

By carefully analyzing the approaches in [CW19, CR20, CLW20], we adapt the algorithmic method to show that f is weakly inapproximable by $\mathsf{Sum} \circ \mathcal{F}_n$-functions (think of $\mathcal{F}_n$ as low-degree $\mathbb{F}_2$-polynomials) of unit ℓ_4-norm. By Lemma 10.2.3, this gives us a $(\delta, \varepsilon)_{\ell_{4/3}}$-witness h.

The bound on the $\ell_{4/3}$-norm of h imposes a smoothness condition on h. Formally, since $\mathbb{E}_{x \leftarrow U_n}[|h(x)|^{4/3}] \leq 1$, for every $t \geq 1$, it holds that

$$\mathop{\mathbb{E}}_{x \leftarrow U_n}\left[|h(x)| \cdot \mathbb{1}_{h(x)|\geq t}\right] \leq t^{-1/3} \mathop{\mathbb{E}}_{x \leftarrow U_n}\left[|h(x)|^{4/3}\right] \leq t^{-1/3}.$$

That is, the total mass of "heavy points" in h is very small. This inspires us to decompose the h as the sum of two functions h_{light} and h_{heavy}, where $h_{\mathsf{light}}(x) := h(x) \cdot \mathbb{1}_{|h(x)|<t}$ and $h_{\mathsf{heavy}}(x) := h(x) \cdot \mathbb{1}_{|h(x)|\geq t}$. Now, since $\|h_{\mathsf{light}}\|_\infty$ is a most t, one can deal with it with the approach discussed above. For h_{heavy}, since $\|h_{\mathsf{heavy}}\|_1$ is small, one may try to simply ignore this part.

The real execution of the above plan is, however, much more complicated than the above sounds. For one, after decomposing h into 2 parts, the function

$$D^\alpha(x) := \prod_{j=1}^{i-1} h(\tilde{r}_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j) \cdot C(T_i(x, \alpha))$$

actually breaks into 2^{i-1} parts. We have to carefully make sure that this exponential blow-up does not cancel any advantage we gain in the decomposition.

Indeed, a simple two-way decomposition seems not sufficient, and in the real proof (see the proof of Lemma 10.4.5), we will actually decompose h into many levels, where the k-th level is defined as $h_k(x) := h(x) \cdot \mathbb{1}_{|h(x)|\in(2^{k-1},2^k]}$ (together with $h_0(x) := h(x) \cdot \mathbb{1}_{|h(x)|\leq 1}$), and apply a novel way to partition the exponential parts of $D^\alpha(\tilde{r}_i)$ into only polynomially many groups, and bound each of them separately. Our decomposition is somewhat similar to the analysis of the "bounded independence plus noise" framework for constructing PRGs developed by Haramaty, Lee, and Viola in [HLV18, LV20], which is later used by Forbes and Kelley to construct PRGs for unordered branching programs [FK18].[10]

[9] Indeed, this is *impossible* if we relax the condition on the total sum of absolute values of coefficients in $C \in \mathsf{Sum} \circ \mathcal{F}_n$: Fixing a function $f\colon \{0,1\}^n \to \{-1,1\}$, one can always construct a $\mathsf{Sum} \circ \mathcal{F}_n$-function C such that $C(0^n) = 2^n \cdot f(0^n)$ while $C(x) = 0$ for every $x \neq 0^n$, by summing only two functions. This C is of sparsity 2, and satisfies $\|C\|_1 = 1$ and $\langle C, f\rangle = 2^{-n} \cdot 2^n = 1$. On the other hands, the algorithmic method can still be used to show weak-inapproximability by $\mathsf{Sum} \circ \mathcal{F}_n$ with unit ℓ_4-norm, even allowing the coefficients to be $2^{O(n)}$.

[10] In more details, in the analysis of [FK18], they partition all monomials of a polynomial into roughly n groups depending on when the monomials become "heavy" (see [FK18, Proposition 6.1]). For the exponentially many terms

Bounding the ℓ_1-Norm

Finally, let us turn to the second condition we wish to establish for Amp^h, which requires us to bound its ℓ_1-norm.

Mixing Amp^h by introducing fresh randomness. Since we essentially have no control over Amp^h, if the generator $\mathcal{G}$ always "hits" the parts of h with large magnitude, then Amp^h could have ℓ_1-norm even larger than 1. For example, if for all $r \in \{0,1\}^m$ and $i \in [k]$ it holds $h(\tilde{r}_i) \geq 1$, then clearly $\|\mathsf{Amp}^h\|_1 \geq 1$ as well.

We resolve this issue by introducing some new fresh randomness to "mix" different parts of h. To see the idea, let $\mathcal{G}$ be an arbitrary generator. Suppose that we sample k uniformly random strings from $\{0,1\}^n$, denoted by $w = (w_1, \ldots, w_k) \in (\{0,1\}^n)^k$. We then consider the following generator

$$\mathcal{G}^w(r) := (\mathcal{G}(r)_1 \oplus w_1, \ldots, \mathcal{G}(r)_k \oplus w_k).$$

We can similarly define

$$\mathsf{Amp}^{f;w} := f^{\oplus k} \circ \mathcal{G}^w \quad \text{and} \quad \mathsf{Amp}^{h;w} := h^{\oplus k} \circ \mathcal{G}^w.$$

For any fixed $r \in \{0,1\}^m$, we have

$$\mathop{\mathbb{E}}_{(w_1,\ldots,w_k)\leftarrow U_{nk}} |\mathsf{Amp}^{h;w}(r)| = \mathop{\mathbb{E}}_{(w_1,\ldots,w_k)\leftarrow U_{nk}} \prod_{i=1}^{k} h(\mathcal{G}(r)_i \oplus w_i) = \|h\|_1^k \leq (1-\delta)^k.$$

The second equality above holds since all the w_i are i.i.d., which means the strings $\{\mathcal{G}(r)_i \oplus w_i\}_{i\in[k]}$ are i.i.d. as well.

Hence, taking an average over all $r \in \{0,1\}^m$, we have

$$\mathop{\mathbb{E}}_{(w_1,\ldots,w_k)\leftarrow U_{nk}} \|\mathsf{Amp}^{h;w}\|_1 = \mathop{\mathbb{E}}_{(w_1,\ldots,w_k)\leftarrow U_{nk}} \mathop{\mathbb{E}}_{r\leftarrow U_m} |\mathsf{Amp}^{h;w}(r)| \leq (1-\delta)^k.$$

With some complications, we will still be able to show that $\mathsf{Amp}^{h;w}$ and $\mathsf{Amp}^{f;w}$ are indistinguishable.[11] This is not surprising at all: for every fixed $w = (w_1, \ldots, w_k)$ the overall effect of w to the generator is simply flipping some input bits to the functions f and h.

resulting from decomposing h, we also partition them into roughly n groups depending on when they become "heavy". The definitions of "heavy" in our work and [FK18] differ since we are in very different settings.

[11] To be more precise, we will show that $\mathbb{E}_{w\in\{0,1\}^{nk}} \mathrm{corr}(\mathsf{Amp}^{f;w} - \mathsf{Amp}^{h;w}, \mathcal{F}_m)$ is small. See the proof of Lemma 10.4.5 for details.

Full derandomization by PRGs for space-bounded computation. However, sampling w still requires nk bits, so it may seem we did not gain anything. Our final proof ingredient is to show that we can in fact generate "good enough" w by PRGs for space-bounded computation ([Nis92]), which only require seeds of length $O(n \log k)$. We denote this generator as $\mathcal{G}_{\mathsf{Nisan}} \colon \{0,1\}^{O(n \log k)} \to \{0,1\}^{nk}$, and take our final generator $\mathcal{G}_{\mathsf{final}}$ as $\mathcal{G}_{\mathsf{final}}(r_1, r_2) \coloneqq \mathcal{G}(r_1) \oplus \mathcal{G}_{\mathsf{Nisan}}(r_2)$, where $\oplus$ denotes the bit-wise XOR. We remark that PRGs for space-bounded computation is also used in the proof of hardness amplification for NP [HVV06, Lemma 5.7], although the usage there is quite different from our usage.

10.2.4 Specific Restriction Generators for $\mathbb{F}_2$-Polynomials and Low-Rank Matrices

In this subsection, we give a high-level overview of our restrictable generators for $\mathbb{F}_2$-Polynomials and low-rank matrices. Recall the definition of a $\mathcal{F}$-restrictable generator $\mathcal{G} : \{0,1\}^m \to \{0,1\}^{nk}$: for every $i \in [k]$, advice $\alpha \in \{0,1\}^{m-n}$, functions $u_1, u_2, \ldots, u_k \colon \{0,1\}^n \to \{-1,1\}$ and $C \in \mathcal{F}_m$, the function:

$$D(x) \coloneqq C(T_i(x,\alpha)) \cdot \prod_{j \in [k] \setminus \{i\}} u_j(\mathcal{G}(T_i(x,\alpha))_j). \tag{10.10}$$

is an $\mathcal{F}_n$-function.

The Star-Like Generator for Correlation Bounds

Roughly speaking, the smaller the function class $\mathcal{F}$, the harder it is to get a restrictable generator for $\mathcal{F}$. Since low-degree $\mathbb{F}_2$-polynomials are not very expressive, it seems extremely hard to obtain a restrictable generator for them directly. On the other hand, exactly due to the fact that low-degree $\mathbb{F}_2$-polynomials are simple enough to be analyzed non-trivially by algorithms, we can utilize the algorithmic method to prove lower bounds for them.

We will consider a larger function collection $\mathcal{F}$ containing low-degree $\mathbb{F}_2$-polynomials as a subset, such that the following hold: (1) $\mathcal{F}$ is still simple enough to be analyzed non-trivially by algorithms and (2) it is expressive enough so that one can design a near-optimal $\mathcal{F}$-restrictable generator. Therefore, we can then apply our derandomized XOR Lemma to prove strong average-case lower bounds against $\mathcal{F}$, which immediately implies correlation bounds against $\mathbb{F}_2$-polynomials.

The larger function collection $\mathcal{F}$. Formally, fixing an integer $n \in \mathbb{N}$ and let $d = \sqrt{n}$. We define $\mathcal{F}$ as the function collection such that $\mathcal{F}$ consists of all functions f which has at least n input bits, and can be written as $f(x) = (-1)^{P(x)} \cdot g(x)$ for $x \in \{0,1\}^m$, where P is an $\mathbb{F}_2$-polynomial with degree bounded by d and $g\colon \{0,1\}^m \to \{-1,1\}$ is a function that only depends on the $(n-d)$-length prefix of its input.

$\mathcal{F}$ is algorithmic friendly. We observe that the fast #SAT algorithm for low-degree polynomials can be extended to $\mathcal{F}_m$-functions naturally. In fact, given a degree-d polynomial $P\colon \mathbb{F}_2^m \to \mathbb{F}_2$, we set $\ell = n/d$. For each $(x_1, \ldots, x_{m-\ell}) \in \{0,1\}^{m-\ell}$, we define

$$s(x_1, \ldots, x_{m-\ell}) = \sum_{(y_1,\ldots,y_\ell)\in\{0,1\}^\ell} (-1)^{P(x_1,\ldots,x_{m-\ell},y_1,\ldots,y_\ell)},$$

where the above sum is over $\mathbb{Z}$ instead of over $\mathbb{F}_2$. Then applying the modulus-amplifying polynomials (see Lemma 10.6.2 for details), there is an algorithm which can compute the list

$$(s(x_1, \ldots, x_{m-\ell}))_{(x_1,\ldots,x_{m-\ell})\in\{0,1\}^{m-\ell}}$$

in $O(2^{m-\Omega(\ell)})$ time. Finally, taking a sum over the list, one can then compute $\sum_{x\in\{0,1\}^m}(-1)^{P(x)}$ in $2^{m-\Omega(\ell)}$ time.

Now, observing that we have set $\ell = d = \sqrt{n}$ and noting that $g(x)$ only depends on the first $n - d = n - \ell \le m - \ell$ bits of x (recall that $m \ge n$), we have

$$\sum_{x\in\{0,1\}^m} f(x) = \sum_{(x_1,\ldots,x_{m-\ell})\in\{0,1\}^{m-\ell}} s(x_1, \ldots, x_{m-\ell}) \cdot g(1, \ldots, x_{n-\ell}, 0, \ldots, 0),$$

which allows us to compute $\sum_{x\in\{0,1\}^m} f(x)$ in $2^{m-\Omega(\ell)}$ time as well.

The star-like generator for $\mathcal{F}$. Since we aim to prove Theorem 1.7.4, in the following we fix the number of instances generated by the generator to be $k = \sqrt{n}$. The $\mathcal{F}$-restrictble generator $\mathcal{G}$ is then designed as follows: It has seed length $m = (n-d) + kd \le 2n$ (recall that $d = \sqrt{n}$). For a seed $r \in \{0,1\}^m$, we write $r = \alpha \circ x_1 \circ \cdots \circ x_k$ where $\alpha \in \{0,1\}^{n-d}$ and $x_1, \ldots, x_k \in \{0,1\}^d$. Then $\mathcal{G}$ is defined as follows:

$$\mathcal{G}(r) := (\alpha \circ x_1, \ldots, \alpha \circ x_k).$$

Now we can justify why we call it the star-like generator: the k instances generated by $\mathcal{G}$ form a star with their common intersection α as the center. For a string $\alpha \in \{0,1\}^{m-n}$, we write $\alpha = \alpha_1 \circ \cdots \circ \alpha_{k-1}$ where $\alpha_i \in \{0,1\}^d$ for every $i \in [k-1]$. For each $i \in [k]$, we define the embedding function T_i as

$$T_i(x,\alpha) = (x_{\leq n-d}, \alpha_1, \ldots, \alpha_{i-1}, x_{>n-d}, \alpha_i, \ldots, \alpha_{k-1}).$$

That is, T_i uses α to fill in the length-d suffix for all instances except for the i-th one, and use x to fill in the i-th instance. It is straightforward to verify that $\mathcal{G}$ satisfies the first two requirements of Definition 10.2.6.

To show that $\mathcal{G}$ satisfies the third requirement of Definition 10.2.6, we have to argue that for every $C \in \mathcal{F}_m$, (10.10) is still in $\mathcal{F}_n$. Observe that for every $j \in [k] \setminus \{i\}$, $u_j(\mathcal{G}(T_i(x,\alpha))_j)$ only depends on $x_{\leq n-d}$, hence it belongs to $\mathcal{F}_n$. It is also easy to verify that $C(T_i(x,\alpha)) \in \mathcal{F}_n$. Since $\mathcal{F}_n$ is closed under multiplication, we can conclude that (10.10) belongs to $\mathcal{F}_n$ as well, which completes the proof. (See the proof of Lemma 10.6.3 for more details.)

The Bi-coloring Generator for Constructing Rigid Matrices

Now we turn to the generator for low-rank matrices, which will be used to construct the extremely rigid matrices in Theorem 1.7.3. First we define the class of "low-rank matrices". For every even $n \geq 1$, we can view a function $f\colon \{0,1\}^n$ as a $2^{n/2} \times 2^{n/2}$ matrix, denoted by M_f, where $M_f(x,y) = f(x,y)$ for every $x,y \in \{0,1\}^{n/2}$. We call the first and last $n/2$-bits of inputs as the *row* index and the *column* index, respectively. Letting $r(n) \geq n^{\omega(1)}$ be the rank parameter, we let $\mathcal{M}_n$ denote the class of functions f whose matrix representation M_f satisfies $\mathrm{rank}(M_f) \leq r(n)$.

We will construct an $\mathcal{M}$-restrictable generator with seed length $n\sqrt{k}$, which improves upon the trivial seed length of nk. More precisely, assuming $\sqrt{k}$ is an integer for simplicity, and letting $t = \sqrt{k}$ and $m = nt$, we choose an arbitrary but fixed injective mapping $\rho\colon [k] \to [t] \times [t]$, denoted by $\rho(i) = (\rho(i)_u, \rho(i)_v)$. For every $z \in \{0,1\}^m$, we write $z = x_1 \circ \cdots \circ x_t \circ y_1 \circ \cdots \circ y_t$ where $|x_i| = |y_j| = n/2$ for every $i,j \in [t]$. Our generator $\mathcal{G}$ is then defined as

$$\mathcal{G}(z) := (x_{\rho(1)_u} \circ y_{\rho(1)_v}, \ldots, x_{\rho(k)_u} \circ y_{\rho(k)_v}).$$

It is then straightforward to construct the required mappings T_i: given $x \in \{0,1\}^n$ and $\alpha \in$

$\{0,1\}^{n(t-1)}$, we simply set $x_{\rho(i)_u}$ to $x_{\leq n/2}$ and $y_{\rho(i)_v}$ to $x_{>n/2}$, and use α to fill the rest of z. It is easy to verify that $\mathcal{G}$ satisfies the first two requirements of Definition 10.2.6.

Intuitively, one can interpret the above construction by thinking about a bipartite graph, where the left and right side have t vertices each, and the seed z as a labeling of all vertices by strings in $\{0,1\}^{n/2}$(*i.e.*, the strings $x_1, \ldots, x_t$ and $y_1, \ldots, y_t$). For every $(i,j) \in [t] \times [t]$, we add an edge between the i-th vertex on the left side and the j-th vertex on the right side, and label this edge with the concatenation of the two $n/2$-bit strings on its endpoints (*i.e.* $x_i \circ y_j$).

Each edge can then be viewed as an instance generated by the seed z (so there are $t^2 \geq k$ instances in total. Now we argue why $\mathcal{G}$ satisfies the third requirement of Definition 10.2.6. The crucial property is that every two distinct edges can only share at most one common points. Hence, fixing $i \in [k]$ and $\alpha \in \{0,1\}^{n(t-1)}$, then for $j \neq i$, either $\rho(j)_u \neq \rho(i)_u$ or $\rho(j)_v \neq \rho(i)_v$.

We can now observe that, for fixed $\alpha \in \{0,1\}^{n(t-1)}$ and every function $u_j \colon \{0,1\}^n \to \{-1,1\}$, $u_j(\mathcal{G}(T_i(x,\alpha))_j)$ either only depends on $x_{\leq n/2}$, or only depends on $x_{>n/2}$, which means it is a matrix of rank at most 1. Taking an XOR (multiplication over the $\{-1,1\}$ basis is equivalent to XOR over the Boolean basis) of $k-1$ such matrices resulting in a matrix of rank at most $k-1$. Moreover, one can also observe that for a low-rank matrix C, $C(T_i(x,\alpha))$ has the same rank as of C. Therefore, (10.10) has low rank too, which completes the argument. (See the proof of Lemma 10.7.5.)

10.3 Preliminaries

Collections of functions. For $n \in \mathbb{N}_{\geq 1}$, an *$n$-input function collection* $\mathcal{F}_n$ is a subset of all n-input Boolean functions. A *function collection* $\mathcal{F} = \bigcup_{n \in \mathbb{N}_{\geq 1}} \mathcal{F}_n$ is a subset of all Boolean functions, where $\mathcal{F}_n$ is an n-input function collection. We say f is an $\mathcal{F}_n$-function (resp. $\mathcal{F}$-function) if $f \in \mathcal{F}_n$ (resp. $f \in \mathcal{F}$).

By *probabilistic $\mathcal{F}$-function* we mean a *distribution* $\mathcal{D}$ over $\mathcal{F}$-functions of same input length. For every n-bit input probabilistic $\mathcal{F}$-function $\mathcal{D}$, we define the *expectation function* of $\mathcal{D}$ as $P_{\mathcal{D}} \colon \{0,1\}^n \to \mathbb{R}$, where $P_{\mathcal{D}}(x) = \mathbb{E}_{D \leftarrow \mathcal{D}}[D(x)]$.

We say a function collection $\mathcal{F}$ is a typical function collection, if it is closed under negation, and flipping a subset of input bits. (That is, for $f \in \mathcal{F}_n$ and every $w \in \{0,1\}^n$, the function $g(x) \coloneqq f(x \oplus w)$ and $-f$ both belong to $\mathcal{F}$ as well.)

Correlation and approximation. For a Boolean function $f\colon \{0,1\}^m \to \{-1,1\}$ and a function collection $\mathcal{F}$, we define the (maximum) correlation between f and $\mathcal{F}$-functions as

$$\mathrm{corr}(f,\mathcal{F}) := \max_{C\in\mathcal{F}_m} \langle f, C\rangle.$$

Slightly abusing the notation, we also use $\mathrm{corr}(f,d)$ to denote $\mathrm{corr}(f,\mathcal{P}_d)$, where $\mathcal{P}_d$ is the collection of all degree-2 $\mathcal{F}_2$-polynomials.

For two functions $f\colon \{0,1\}^n \to \{-1,1\}$ and a function collection $\mathcal{F}_n$, we say that f cannot be γ-approximated by $\mathcal{F}_n$ if $\Pr_{x\leftarrow U_n}[f(x) = g(x)] < \gamma$ for every $g \in \mathcal{F}_n$. By a standard connection, $\mathrm{corr}(f,\mathcal{F}_n) < \varepsilon$ if and only if f cannot be $(1/2+\varepsilon/2)$-approximated by $\mathcal{F}_n$.

Arithmetization. We will crucially exploit multi-linear extension of Boolean functions of the following form. For every function $f\colon \{-1,1\}^n \to \{0,1\}$[12], we use $\tilde{f}\colon \mathbb{R} \to \mathbb{R}$ to denote the multi-linear extension of f. That is, we define

$$\tilde{f}(x) := \sum_{y\in\{-1,1\}^n} f(y)\cdot \prod_{j=1}^{n}\left(y_i \cdot \frac{x_i+y_i}{2}\right).$$

One can verify that $\tilde{f}(x)$ is multi-linear and is indeed an extension of f. Moreover, one can observe that the absolute value of the coefficient of each monomial in $\tilde{f}$ is at most 2^n.

10.4 Derandomized XOR Lemma

In this section we prove our derandomized XOR lemma, which is stated formally as below.

Throughout this section, for every $p \in \mathbb{R}_{>1}$, we set $c_p^{\mathsf{k}} \in (0,1)$ to be a small universal constant such that

$$H_{\mathrm{b}}(c_p^{\mathsf{k}}/(1+c_p^{\mathsf{k}}))(1+c_p^{\mathsf{k}}) < \frac{1}{p}(1-c_p^{\mathsf{k}}),$$

where $H_{\mathrm{b}}(q) := -q\log_2 q - (1-q)\log_2(1-q)$ is the binary entropy function. Recall that we say a function collection $\mathcal{F}$ is typical, if it is closed under negation and flipping a subset of input bits.

Lemma 10.4.1 (Derandomized XOR lemma)**.** *Let $\delta \in (0,0.1)$ and $p \in \mathbb{R}_{>1}$ be three constants. For*

[12]Note that the input of f is from $\{-1,1\}^n$, while the output is in $\{0,1\}$. We will use this form of arithmetization in both Section 10.5 and Section 10.7. Check Section 10.5.1 and Section 10.7.1 for corresponding discussions.

every sufficiently large $n \in \mathbb{N}$, every $\varepsilon \in [2^{-n}, 1)$ and every function $f\colon \{0,1\}^n \to \{-1,1\}$.[13] *Let $\mathcal{F}$ be a typical function collection, and $k = \left\lceil c_p^{\mathsf{k}} \cdot \log \varepsilon^{-1}/5 \right\rceil$. Suppose the following two conditions hold:*

1. *(**Weak inapproximability by** $\mathsf{Sum} \circ \mathcal{F}_n$.) $\langle f, C\rangle < (1-\delta)$ for every $\mathsf{Sum} \circ \mathcal{F}_n$-function C such that $\mathsf{complexity}(C) \le 10 \cdot n/\varepsilon^2$ and $\|C\|_p \le 1$.*
2. *(**Existence of an $\mathcal{F}$-restrictable generator**.) There is an $\mathcal{F}$-restrictable generator $\mathcal{G}_{\mathsf{res}}\colon \{0,1\}^m \to \{0,1\}^{nk}$ with seed length $m \ge n$, which is computable in $\mathrm{poly}(m)$ time.*

Then there is a polynomial-time computable generator $\mathcal{G}\colon \{0,1\}^{m+\ell} \to \{0,1\}^{nk}$ such that

$$\mathrm{corr}(f^{\oplus k} \circ \mathcal{G}, \mathcal{F}) \le \varepsilon^{\Omega(\delta)},$$

where $\ell \le O(m \log m)$.

Moreover, if $\varepsilon \ge 2^{-n^{1-c}}$ for some constant $c \in (0, 0.1)$, then this bound on ℓ can be further improved to $\ell \le O_c(m)$.

Our proof of Lemma 10.4.1 will follow the proof outline in Section 10.2.3: In Section 10.4.1, we show how to construct $(\delta,\varepsilon)_{\ell_p}$-witnesses from weak inapproximability against $\mathsf{Sum} \circ \mathcal{F}_n$-functions, and formally prove Lemma 10.2.3. In Section 10.4.2, we first establish a partially derandomized XOR lemma, which is captured by Lemma 10.4.2. In Section 10.4.3, we apply PRGs for space-bounded computation to finish the proof of Lemma 10.4.1.

The "moreover" part says that if ε is slightly sub-exponential (*i.e.* $\varepsilon \ge 2^{-n^{1-\Omega(1)}}$), then we can obtain an optimal linear-seed generator.

10.4.1 The Existence of (δ,ε)-Witnesses from Hardness Against Linear Sum of Functions

In this section, we prove Lemma 10.2.3 (restated below).

Reminder of Lemma 10.2.3. *Let $n \in \mathbb{N}_{\ge 1}$, and let $\mathcal{F}_n$ be a collection of n-input functions that is closed under negation. Let $p, q \in \mathbb{R}_{\ge 1} \cup \{\infty\}$ be such that p and q are Hölder conjugates of each other. For every function $f\colon \{0,1\}^n \to \{-1,1\}$ and $\delta, \varepsilon > 0$, if we have*

$$\langle f, C\rangle < (1-\delta)$$

[13] By Lemma 10.1.1, the original XOR Lemma takes $O(n \log 1/\varepsilon)$ bits of inputs. Therefore, we mainly focus on the case that ε is sufficiently small.

for every $\mathsf{Sum} \circ \mathcal{F}_n$*-function* C *such that* $\mathsf{complexity}(C) \leq 10 \cdot n/\varepsilon^2$ *and* $\|C\|_q \leq 1$*, then there is a* $(\delta, \varepsilon)_{\ell_p}$*-witness* h *for* f *against* $\mathcal{F}_n$*-functions.*

Moreover, for the case $p = \infty$ *and* $q = 1$*, the condition can be replaced by that for every* $\mathsf{MAJ} \circ \mathcal{F}$*-function* C *with top-sparsity bounded by* $10n/\varepsilon^2$*, it holds that* $\langle f, C \rangle < 1 - 2\delta$*.*

Proof. For the general case, we argue as follows.

The Challenger-Distinguisher game. We consider the following two-player zero-sum game:

1. The Max-Player (Distinguisher) chooses an $\mathcal{F}_n$-probabilistic function $\mathcal{D}$ on n-bit inputs. (That is, $\mathcal{D}$ is a distribution over $\mathcal{F}_n$-functions.) Recall that we use $P_{\mathcal{D}} \colon \{0,1\}^n \to \mathbb{R}$ to denote the expectation function of $\mathcal{D}$. It is defined as $P_{\mathcal{D}}(x) = \mathbb{E}_{D \leftarrow \mathcal{D}}[D(x)]$ for every $x \in \{0,1\}^n$.
2. The Min-Player (Challenger) chooses a function $h \colon \{0,1\}^n \to \mathbb{R}$ such that $\|h\|_p \leq 1$ and $\|h\|_1 \leq 1 - \delta$. (*i.e.*, h satisfies the norm conditions for being a $(\delta, \varepsilon)_{\ell_p}$-witness.)
3. The payoff of game is $\langle P_{\mathcal{D}}, f - h \rangle$. (Note that $f \colon \{0,1\}^n \to \{-1, 1\}$ is a fixed function.) The Min-Player (resp. Max-Player) wants to minimize (resp. maximize) the payoff.

Note that the strategy spaces of both players are compact convex sets and the payoff has a bilinear form. Therefore, by the minimax theorem, the game has a unique equilibrium payoff V_{game} when both players play optimally. We claim that $V_{\mathsf{game}} \leq \varepsilon$.

Indeed, suppose on the contrary that $V_{\mathsf{game}} > \varepsilon$. This implies that the Max-Player has a strategy $\mathcal{D}$, which is a probabilistic $\mathcal{F}_n$-function, such that for every Min-Player strategy h, it holds that

$$\langle P_{\mathcal{D}}, f - h \rangle > \varepsilon. \tag{10.11}$$

Next, we sample $t = 10 \cdot n/\varepsilon^2$ independent functions from $\mathcal{D}$, denoted by $D_1, \ldots, D_t$. By a Chernoff bound, for every $x \in \{0,1\}^n$, it holds that

$$\Pr_{D_1, \ldots, D_t} \left[\left| P_{\mathcal{D}}(x) - \mathop{\mathbb{E}}_{i \leftarrow [t]} D_i(x) \right| \geq \frac{\varepsilon}{2} \right] \leq 2^{-n-1}.$$

Then we can fix a set of functions $\{D_i\}_{i \in [t]}$ such that

$$\left| P_{\mathcal{D}}(x) - \mathop{\mathbb{E}}_{i \leftarrow [t]} D_i(x) \right| \leq \varepsilon/2 \quad \text{holds for every } x \in \{0,1\}^n. \tag{10.12}$$

We now construct a $\mathsf{Sum} \circ \mathcal{F}_n$-function D' as

$$D'(x) := \sum_{i=1}^{t} \frac{1}{t} \cdot D_i(x).$$

Note that D' has sparsity bounded by $t = 10 \cdot n/\varepsilon^2$. For every strategy h of Min-Player, it follows that

$$\begin{aligned}
\langle D', f - h\rangle &= \langle P_{\mathcal{D}}, f - h\rangle - \langle P_{\mathcal{D}} - D', f - h\rangle \\
&> \varepsilon - \|D' - P_{\mathcal{D}}\|_\infty \|f - h\|_1 && \text{((10.11) and Lemma 3.2.2)} \\
&\geq \varepsilon - \frac{\varepsilon}{2}(1 + \|h\|_1) && \text{((10.12) and } \|f\|_1 \leq \|f\|_\infty = 1) \\
&\geq 0, && (10.13)
\end{aligned}$$

where the last inequality follows from $\|h\|_1 \leq 1 - \delta \leq 1$. By Lemma 3.2.4 and the assumption that p and q are Hölder conjugates of each other, the Min-Player can choose a strategy h such that $\|h\|_p \leq 1 - \delta$ and $\langle D', h\rangle = (1 - \delta) \cdot \|D'\|_q$. Note that $\|h\|_1 \leq \|h\|_p \leq 1 - \delta$ as well, so h is a valid strategy. Therefore, (10.13) in particular implies that

$$\begin{aligned}
\langle D', f\rangle &= \langle D', h\rangle + \langle D', f - h\rangle \\
&> (1 - \delta)\|D'\|_q.
\end{aligned}$$

Next, we give a lower bound on $\|D'\|_q$. Choosing $h \equiv 0$, (10.11) implies that $\|P_{\mathcal{D}}\|_1 \geq \varepsilon$. Therefore, $\|D'\|_q \geq \|D'\|_1 \geq \|P_{\mathcal{D}}\|_1 - \frac{\varepsilon}{2} \geq \varepsilon/2$.

Letting $C = D'/\|D'\|_q$, we have $\langle C, f\rangle > 1 - \delta$, $\|C\|_q = 1$ and $\mathsf{complexity}(C) \leq \max(t, 1/\|D\|_q) \leq 10 \cdot n/\varepsilon^2$. This contradicts the assumption of the lemma.

Finally, given that $V_{\mathsf{game}} \leq \varepsilon$, it is straightforward to verify that an optimal strategy h of the Min-Player satisfies the requirement of being a $(\delta, \varepsilon)_{\ell_p}$-witness: First, we have that $\|h\|_1 \leq 1 - \delta$ and $\|h\|_p \leq 1$. Second, for every $C \in \mathcal{F}$ it holds that $\langle C, f - h\rangle \leq \varepsilon$. Since $\mathcal{F}$ is closed under negation, it in turn implies that $|\langle C, f - h\rangle| \leq \varepsilon$.

Impagliazzo's hardcore lemma. In the following we prove the "moreover" part in the statement of Lemma 10.2.3. We consider the same Challenger-Distinguisher game as before with $p = \infty$. (*i.e.*, the Min-Player chooses a function h with $\|h\|_1 \leq 1 - \delta$ and $\|h\|_\infty \leq 1$.)

Again by the minimax theorem, this game has a unique payoff V_{game} when both players play optimally. We claim $V_{\mathsf{game}} \leq \varepsilon$.

Suppose that $V_{\mathsf{game}} > \varepsilon$, then there is a probabilistic $\mathcal{F}_n$-function $\mathcal{D}$ such that for every strategy h by the Min-Player, it holds that

$$\langle P_{\mathcal{D}}, f - h \rangle > \varepsilon. \tag{10.14}$$

Call an input $x \in \{0,1\}^n$ bad if $|f(x) - P_{\mathcal{D}}(x)| > (1-\varepsilon)$. Let B be the set of bad inputs.

We claim that (10.14) implies $|B| \leq \delta 2^n$. Otherwise, suppose that $|B| > \delta 2^n$. We define a function h_B as $h_B(x) = \mathbb{1}_{x \notin B} \cdot f(x)$, and observe that $\|h_B\|_1 \leq 1 - \delta$. Then we have

$$\langle P_{\mathcal{D}}, f - h_B \rangle = \mathop{\mathbb{E}}_{x \leftarrow U_n}[\mathbb{1}_{x \in B} \cdot P_{\mathcal{D}}(x) \cdot f(x)] \leq \frac{|B|}{2^n}\varepsilon \leq \varepsilon,$$

which contradicts to (10.14). The penultimate inequality above follows from the fact that when $x \in B$ and $|f(x) - P_{\mathcal{D}}(x)| > (1-\varepsilon)$, we have $f(x) \cdot P_{\mathcal{D}}(x) \leq \varepsilon$ since $P_{\mathcal{D}}(x) \in [-1,1]$.

Now, given $|B| \leq \delta 2^n$, we can use a $\mathsf{Sum} \circ \mathcal{F}_n$ function C with $\mathsf{complexity}(C) \leq 10n/\varepsilon^2$ to pointwise approximate $P_{\mathcal{D}}$ within an additive error of $\varepsilon/2$. It follows that for every $x \notin B$, we have $f(x) = \mathrm{sign}(C(x))$. Therefore, we can convert C to a $\mathsf{MAJ} \circ \mathcal{F}_n$-function C' such that $C'(x) = f(x)$ for every $x \notin B$. Since $|B| \leq \delta 2^n$, we have

$$\langle C', f \rangle \geq 1 - \frac{2|B|}{2^n} \geq 1 - 2\delta.$$

this is a contradiction. So it must be the case that $V_{\mathsf{game}} \leq \varepsilon$.

Finally, similar to the general case above, given that $V_{\mathsf{game}} \leq \varepsilon$, it is straightforward to verify that an optimal strategy h of the Min-Player satisfies the requirement of being a $(\delta, \varepsilon)_{\ell_p}$-witness, which completes the proof of the moreover part.

The above is essentially identical to Nisan's proof of the hardcore lemma. ■

10.4.2 Partial Derandomization Using $\mathcal{F}$-Restrictable Generators

Following our proof overview, in this subsection, we first recall the concept of $\mathcal{F}$-restrictable generators and then prove a partially derandomized XOR lemma (Lemma 10.4.2).

Reminder of Definition 10.2.6. *Given a function collection $\mathcal{F}$ and $n \in \mathbb{N}_{\geq 1}$, a generator $\mathcal{G} \colon \{0,1\}^m \to$*

$\{0,1\}^{nk}$ *is called* $\mathcal{F}$*-restrictable, if there are* k *embedding functions* $T_1, \ldots, T_k \colon \{0,1\}^n \times \{0,1\}^{m-n} \to \{0,1\}^m$ *such that the following hold:*

1. *All the* T_i *are bijections.*
2. *For every* $i \in [k]$ *and* $(x, \alpha) \in \{0,1\}^n \times \{0,1\}^{m-n}$, $\mathcal{G}(T_i(x,\alpha))_i = x$. *That is,* $T_i(x,\alpha) \in \{0,1\}^m$ *is a seed to* $\mathcal{G}$ *which fixes the* i*-th instance of* $\mathcal{G}(T_i(x,\alpha))$ *to be* x.
3. *For every* $\mathcal{F}_m$*-function* $C \colon \{0,1\}^m \to \{-1,1\}$, $i \in [k]$, $\alpha \in \{0,1\}^{m-n}$ *and functions* $u_1, \ldots, u_k \colon \{0,1\}^n \to \{-1,1\}$, *the function*

$$D(x) := C(T_i(x,\alpha)) \cdot \prod_{j \in [k] \setminus \{i\}} u_j(\mathcal{G}(T_i(x,\alpha))_j)$$

belongs to $\mathcal{F}_n$.

The following lemma gives a partially derandomized XOR lemma, modulo the need of nk fresh random bits w.

Lemma 10.4.2. *Let* $\delta \in (0, 0.1)$ *and* $p \in \mathbb{R}_{>1}$ *be two constants. For every sufficiently large* $n \in \mathbb{N}$, *every* $\varepsilon \in [2^{-n}, 1)$ *and every function* $f \colon \{0,1\}^n \to \{-1,1\}$. *Let* $\mathcal{F}$ *be a typical function collection and* $k = \left\lceil c_p^{\mathsf{k}} \cdot \log \varepsilon^{-1} / 5 \right\rceil$. *Suppose the following two conditions hold:*

1. (***Weak inapproximability by*** $\mathsf{Sum} \circ \mathcal{F}_n$.) $\langle f, C \rangle < (1-\delta)$ *for every* $\mathsf{Sum} \circ \mathcal{F}_n$*-function* C *such that* $\mathsf{complexity}(C) \leq 10 \cdot n/\varepsilon^2$ *and* $\|C\|_p \leq 1$.
2. (***Existence of an*** $\mathcal{F}$***-restrictable generator***.) *There is an* $\mathcal{F}$*-restrictable generator* $\mathcal{G}_{\mathsf{res}} \colon \{0,1\}^m \to \{0,1\}^{nk}$ *with seed length* $m \geq n$, *which is computable in* $\mathrm{poly}(m)$ *time.*

For every sequence $w = (w_1, \ldots, w_k) \in (\{0,1\}^n)^k$, *we define a function* $g^w : \{0,1\}^m \to \{-1,1\}$ *as*

$$g^w(r) := \prod_{i=1}^{k} f(\mathcal{G}_{\mathsf{res}}(r)_i \oplus w_i).$$

Then we have

$$\mathop{\mathbb{E}}_{w \leftarrow U_{nk}} [\mathrm{corr}(g^w, \mathcal{F})] \leq \varepsilon^{\Omega(\delta)}.$$

The rest of this subsection is devoted to the proof of Lemma 10.4.2.

Notation and Construction of the Hybrids

We begin by introducing some notation, which will be used throughout Section 10.4.2 and Section 10.4.3. We set a parameter τ as $\lceil \frac{1}{5} \log \varepsilon^{-1} \rceil$ so that

$$k = \left\lceil c_p^{\mathsf{k}} \cdot \log \varepsilon^{-1}/5 \right\rceil \leq c_p^{\mathsf{k}} \cdot \left\lceil \frac{1}{5} \log \varepsilon^{-1} \right\rceil \leq c_p^{\mathsf{k}} \cdot \tau.$$

For a seed $r \in \{0,1\}^m$ to the generator $\mathcal{G}_{\mathsf{res}}$, we use the same notation as in Section 10.2.3 and write

$$\tilde{r}_j = \mathcal{G}_{\mathsf{res}}(r)_j \quad \text{for every } j \in [k].$$

The witness h. First, we observe that by Lemma 10.2.3 and the first condition of Lemma 10.4.2, there is a $(\delta, \varepsilon)_{\ell_{p/(p-1)}}$-witness h' for f against $\mathcal{F}_n$ functions. That is:

1. (**Indistinguishability.**) $\mathrm{corr}(f - h', \mathcal{F}_n) \leq \varepsilon$.
2. (**Bounded ℓ_1-norm**) h' has ℓ_1-norm at most $1 - \delta$ (*i.e.*, $\mathbb{E}_{x \leftarrow U_n}[|h'(x)|] \leq 1 - \delta$).
3. (**Bounded $\ell_{p/(p-1)}$-norm.**) h' has $\ell_{p/(p-1)}$-norm at most 1.

It would be convenient to have an ℓ_∞-norm bound on the witness. To achieve this, we define from h' another function h as

$$h(x) := \mathrm{sign}(h'(x)) \cdot \min(|h'(x)|, \varepsilon^{1-p}).$$

We observe that (see also (10.15))

$$\left\| h'(x) \cdot \mathbb{1}_{|h'(x)| > \varepsilon^{1-p}} \right\|_1 \leq \varepsilon.$$

Therefore, $\|h' - h\|_1 \leq \varepsilon$. Also, since $\|h\|_{p/(p-1)} \leq \|h'\|_{p/(p-1)}$ and $\|h\|_1 \leq \|h'\|_1 \leq 1$, it turns out that h is a $(\delta, 2\varepsilon)_{\ell_{p/(p-1)}}$-witness for f against $\mathcal{F}_n$-functions, which has ℓ_∞-norm bounded above by ε^{1-p}. This witness h will play the pivotal role in the derandomization of the XOR lemma.

The decomposition of the witness h. Letting $\sigma = \min(n, \lceil \log \varepsilon^{1-p} \rceil)$, as discussed in Section 10.2.3, we decompose the witness h into $\sigma + 1$ components as follows. For $\ell \in [\sigma]$, we define the function

$$h_{=\ell}(x) := h(x) \cdot \mathbb{1}_{|h(x)| \in (2^{\ell-1}, 2^\ell]}.$$

And we also define

$$h_{=0}(x) := h(x) \cdot \mathbb{1}_{|h(x)| \leq 1}.$$

Since $\|h\|_\infty \le 2^\sigma$, it follows that $h = \sum_{\ell=0}^{\sigma} h_{=\ell}$. We collect the following two important properties of the functions $h_{=\ell}$.

Claim 10.4.3. *For all $\ell \in \{0, 1, \ldots, \sigma\}$, the following hold:*

1. $\|h_{=\ell}\|_1 \le 2^{-(\ell-1)/(p-1)}$,
2. $\|h_{=\ell}\|_\infty \le 2^\ell$.

In other words, in the decomposition, the higher the absolute values in $h_{=\ell}$ are, the smaller the ℓ_1-norm of h_ℓ is. This observation is the key for our new derandomized XOR lemma.

Proof. The second claim just follows from the definition of the $h_{=\ell}$.

For every $t \ge 1$, we have

$$\mathop{\mathbb{E}}_{x \leftarrow U_n}\left[|h(x)| \cdot \mathbb{1}_{|h(x)| \ge t}\right] \le t^{-1/(p-1)} \mathop{\mathbb{E}}_{x \leftarrow U_n}\left[|h(x)|^{p/(p-1)}\right] \le t^{-1/(p-1)}. \tag{10.15}$$

If $\ell = 0$, then clearly $\|h_{=0}\|_1 \le \|h\|_1 \le 1 \le 2^{-(\ell-1)/(p-1)}$. For $\ell \in \mathbb{N}_{\ge 1}$, applying (10.15), it follows that

$$\|h_{=\ell}(x)\|_1 \le \mathop{\mathbb{E}}_{x \leftarrow U_n}\left[|h(x)| \cdot \mathbb{1}_{|h(x)| \ge 2^{\ell-1}}\right] \le 2^{-(\ell-1)/(p-1)}. \quad \blacksquare$$

Hybrid functions and their decompositions. Following the outline in Section 10.2.3, we will apply a hybrid argument. Recall that we have defined the hybrid functions $H_i^{h,f} = h^{\oplus i} \otimes f^{\oplus k-i}$ in Section 10.2.2, which was later upgraded to $\mathcal{G}_i^{h,f} = H_i^{h,f} \circ \mathcal{G}_{\mathsf{res}}$ in Section 10.2.3.

Here we will also need to define the hybrid functions with respect to the sequence $w = (w_1, \ldots, w_k)$. We use $\mathcal{G}_{\mathsf{res}}^w$ to denote the generator

$$\mathcal{G}_{\mathsf{res}}^w(r) \coloneqq (\tilde{r}_1 \oplus w_1, \ldots, \tilde{r}_k \oplus w_k) \quad \text{for every } r \in \{0,1\}^m.$$

For each $i \in \{0, \ldots, k\}$ and $w \in (\{0,1\}^n)^k$, we define a hybrid function $\mathcal{G}_i^{h,f;w} \coloneqq H_i^{h,f} \circ \mathcal{G}_{\mathsf{res}}^w$. That is, for every $r \in \{0,1\}^m$, we have

$$\mathcal{G}_i^{h,f;w}(r) = \prod_{j=1}^{i} h(\tilde{r}_j \oplus w_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j \oplus w_j).$$

Now we decompose the h functions in $\mathcal{G}_i^{h,f;w}$ using the decomposition of the function h. For

an i-tuple $(\ell_1, \ldots, \ell_i) \in \{0, 1, \ldots, \sigma\}^i$, we define

$$\mathcal{G}^{h,f;w}_{i;(\ell_1,\ldots,\ell_i)}(r) = \prod_{j=1}^{i} h_{=\ell_j}(\tilde{r}_j \oplus w_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j \oplus w_j).$$

That is, for each $j \in [i]$, for the j-th h-function in the product defining $\mathcal{G}^{h,f;w}_i$, we take the ℓ_j-th component $h_{=\ell_j}$ of h.

As discussed in Section 10.2.3, such a decomposition gives us a very *fine-grained* trade-off between ℓ_∞-norm and the ℓ_∞-norm of the h-functions in $\mathcal{G}^{h,f;w}_{i;(\ell_1,\ldots,\ell_i)}$. This will be very helpful for implementing our hybrid argument later.

Final hybrid functions. Now we are ready to define our hybrid functions. For $i \in [k]$, we let $\mathcal{L}_i$ be a set of i-tuples defined as

$$\mathcal{L}_i := \left\{ (\ell_1, \ldots, \ell_i) : (\ell_1, \ldots, \ell_i) \in \{0, 1, \ldots, \sigma\}^i \text{ and } \sum_{j=1}^{i} \ell_j \leq \tau \right\}.$$

We will use ε to denote the empty tuple, and then we define $\mathcal{L}_0 := \{\varepsilon\}$. We now define

$$\mathcal{W}^{h,f;w}_i := \sum_{(\ell_1,\ldots,\ell_i)\in\mathcal{L}_i} \mathcal{G}^{h,f;w}_{i;(\ell_1,\ldots,\ell_i)}.$$

Note that when $i = 0$, since $\mathcal{G}^{h,f;w}_i$ has no h-component to be decomposed, we will let $\mathcal{G}^{h,f;w}_{0;\varepsilon} := \mathcal{G}^{h,f;w}_0$ and hence we have $\mathcal{W}^{h,f;w}_0 = \mathcal{G}^{h,f;w}_0 = f^{\oplus k} \circ \mathcal{G}^w_{\mathsf{res}}$, which is simply g^w.

For every $i \in [k]$, we also let $\mathcal{R}_i$ be a set of i-tuples defined as

$$\mathcal{R}_i := \left\{ (\ell_1, \ldots, \ell_i) : (\ell_1, \ldots, \ell_i) \in \{0, 1, \ldots, \sigma\}^i \text{ and } \sum_{j=1}^{i} \ell_j > \tau \text{ and } \sum_{j=1}^{i-1} \ell_j \leq \tau \right\}.$$

The following upper bounds on the size of sets $\mathcal{L}_i$ and $\mathcal{R}_i$ will be useful for us.

Claim 10.4.4 (Upper bounds on $|\mathcal{L}_i|$ and $|\mathcal{R}_i|$)**.** *For every $i \in \{0, 1, \ldots, k\}$, it holds that $|\mathcal{L}_i| = \binom{\tau+i}{i}$. For every $i \in \{1, \ldots, k\}$, it holds that $|\mathcal{R}_i| \leq |\mathcal{L}_{i-1}|(\sigma + 1)$.*

Implementing the Hybrid Argument

Note that Lemma 10.4.2 is then asking us to bound

$$\mathop{\mathbb{E}}_{w}\left[\max_{C\in\mathcal{F}}\langle C, \mathcal{W}_0^{h,f;w}\rangle\right].$$

We will now apply a standard hybrid argument with respect to the following list of hybrids:

$$g^w = \mathcal{W}_0^{h,f;w}, \quad \mathcal{W}_1^{h,f;w}, \quad \ldots, \quad \mathcal{W}_k^{h,f;w}.$$

The following lemma implement the hybrid argument.

Lemma 10.4.5 ($\mathcal{W}_i^{h,f;w}$ and $\mathcal{W}_{i+1}^{h,f;w}$ are indistinguishable by $\mathcal{F}_m$-functions)**.** *For every $i \in \{0,1,\ldots,k-1\}$, it holds that*

$$\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left[\max_{C\in\mathcal{F}_m}\left|\langle C, \mathcal{W}_i^{h,f;w}\rangle - \langle C, \mathcal{W}_{i+1}^{h,f;w}\rangle\right|\right] \le \varepsilon^{\Omega(1)}.$$

Lemma 10.4.6 ($\mathcal{W}_k^{h,f;w}$ has small ℓ_1-norm)**.** *It holds that*

$$\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left\|\mathcal{W}_k^{h,f;w}\right\|_1 \le \varepsilon^{\Omega(\delta)}.$$

Assuming the two lemmas above, we can derive Lemma 10.4.2 as shown below.

Proof of Lemma 10.4.2. We have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}[\mathrm{corr}(g^w, \mathcal{F})] \\
&= \mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left[\max_{C\in\mathcal{F}}\{\langle C, \mathcal{W}_0^{h,f;w}\rangle\}\right] && \text{(by definition)} \\
&= \mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left[\max_{C\in\mathcal{F}}\left\{\langle C, \mathcal{W}_k^{h,f;w}\rangle + \sum_{i=0}^{k-1}\left(\langle C, \mathcal{W}_i^{h,f;w}\rangle - \langle C, \mathcal{W}_{i+1}^{h,f;w}\rangle\right)\right\}\right] \\
&\le \mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left[\max_{C\in\mathcal{F}}\{\langle C, \mathcal{W}_k^{h,f;w}\rangle\}\right] + k\cdot\varepsilon^{\Omega(\delta)} && \text{(by Lemma 10.4.5)} \\
&\le \mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left\|\mathcal{W}_k^{h,f;w}\right\|_1 + \varepsilon^{\Omega(\delta)} && (\|C\|_\infty = 1 \text{ and } k \le O(\log \varepsilon^{-1})) \\
&\le \varepsilon^{\Omega(\delta)}. && \text{(by Lemma 10.4.6)}
\end{aligned} \tag{10.16}$$

■

Proofs of Lemma 10.4.5 and Lemma 10.4.6

We begin with the proof of Lemma 10.4.6, which is the simple one.

Proof of Lemma 10.4.6. Recall that $k \geq \Omega(\log \varepsilon^{-1})$, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{w \leftarrow U_{nk}} \|\mathcal{W}_k^{h,f;w}\|_1 \leq \mathop{\mathbb{E}}_{w \leftarrow U_{nk}} \|\mathcal{G}_k^{h,f;w}\|_1 &\leq \mathop{\mathbb{E}}_{w \leftarrow U_{nk}} \mathop{\mathbb{E}}_{r \leftarrow U_m} \left[\left| \prod_{j=1}^{k} h(\tilde{r}_j \oplus w_j) \right| \right] \\
&\leq \mathop{\mathbb{E}}_{r \leftarrow U_m} \|h\|_1^k \\
&\leq (1-\delta)^k \\
&\leq \varepsilon^{\Omega(\delta)}. \quad \blacksquare
\end{aligned}$$

The following lemma will be useful for the proof of Lemma 10.4.5. Its proof is based on some simple but lengthy manipulations, we defer its proof to the end of this subsection.

Lemma 10.4.7. *For every $j \in [k]$, $C \in \mathcal{F}_m$ and functions $q_1, \ldots, q_{j-1} \colon \{0,1\}^n \to \mathbb{R}$, it holds that*

$$\begin{aligned}
&\mathop{\mathbb{E}}_{r \leftarrow U_m} \left[C(r) \cdot \prod_{i=1}^{j-1} q_i(\tilde{r}_i) \cdot \prod_{i=j}^{k} f(\tilde{r}_i \oplus w_i) \right] - \mathop{\mathbb{E}}_{r \leftarrow U_m} \left[C(r) \cdot h(\tilde{r}_j \oplus w_j) \cdot \prod_{i=1}^{j-1} q_i(\tilde{r}_i) \cdot \prod_{i=j+1}^{k} f(\tilde{r}_i \oplus w_i) \right] \\
&\leq 2\varepsilon \cdot \prod_{i=1}^{j-1} \|q_i\|_\infty.
\end{aligned}$$

Next we prove Lemma 10.4.5.

Proof of Lemma 10.4.5. For every $i \in \{0, \ldots, k\}$, recall that $\mathcal{L}_i$ is the set of i-tuples $(\ell_1, \ldots, \ell_i) \in \{0, 1, \ldots, \sigma\}^i$ such that $\sum_{j=1}^{i} \ell_j \leq \tau$, and $\mathcal{R}_i$ is the set of i-tuples $(\ell_1, \ldots, \ell_i) \in \{0, 1, \ldots, \sigma\}^i$ such that $\sum_{j=1}^{i} \ell_j > \tau$ and $\sum_{j=1}^{i-1} \ell_j \leq \tau$.

By Claim 10.4.4, $|\mathcal{L}_i| = \binom{\tau+i}{i}$. For each tuple $(\ell_1, \ldots, \ell_i)$ in $\mathcal{L}_i$, we define

$$\begin{aligned}
Q_i(\ell_1, \ldots, \ell_i) &:= \langle C, \mathcal{G}^{h,f;w}_{i;(\ell_1,\ldots,\ell_i)} \rangle \\
&= \mathop{\mathbb{E}}_{r \leftarrow U_m} \left[C(r) \cdot \prod_{j=1}^{i} h_{=\ell_j}(\tilde{r}_j \oplus w_j) \cdot \prod_{j=i+1}^{k} f(\tilde{r}_j \oplus w_j) \right], \\
R_i(\ell_1, \ldots, \ell_i) &:= \mathop{\mathbb{E}}_{r \leftarrow U_m} \left[C(r) \cdot h(\tilde{r}_{i+1} \oplus w_{i+1}) \cdot \prod_{j=1}^{i} h_{=\ell_j}(\tilde{r}_j \oplus w_j) \cdot \prod_{j=i+2}^{k} f(\tilde{r}_j \oplus w_j) \right].
\end{aligned}$$

Applying Lemma 10.4.7, for each $(\ell_1, \ldots, \ell_i) \in \mathcal{L}_i$, we have

$$\begin{aligned} |Q_i(\ell_1, \ldots, \ell_i) - R_i(\ell_1, \ldots, \ell_i)| &\leq \varepsilon \cdot \prod_{j=1}^{i} \|h_{=\ell_j}\|_\infty. \\ &\leq \varepsilon \cdot 2^{\sum_{j=1}^{i} \ell_j} && \text{(Item (2) of Claim 10.4.3)} \\ &\leq \varepsilon \cdot 2^{\tau}. && (\sum_{j=1}^{i} \ell_j \leq \tau) \end{aligned} \tag{10.17}$$

From the definition of $\mathcal{W}_i^{h,f;w}$, it follows that

$$\begin{aligned} \langle C, \mathcal{W}_i^{h,f;w} \rangle &= \left\langle C, \sum_{(\ell_1,\ldots,\ell_i) \in \mathcal{L}_i} \mathcal{G}_{i;(\ell_1,\ldots,\ell_i)}^{h,f;w} \right\rangle \\ &= \sum_{(\ell_1,\ldots,\ell_i) \in \mathcal{L}_i} \langle C, \mathcal{G}_{i;(\ell_1,\ldots,\ell_i)}^{h,f;w} \rangle \qquad (10.18) \\ &= \sum_{(\ell_1,\ldots,\ell_i) \in \mathcal{L}_i} Q_i(\ell_1, \ldots, \ell_i). \qquad (10.19) \end{aligned}$$

Similarly, from the definition of $\mathcal{W}_{i+1}^{h,f;w}$, we have

$$\begin{aligned} \langle C, \mathcal{W}_{i+1}^{h,f;w} \rangle &= \left\langle C, \sum_{(\ell_1,\ldots,\ell_{i+1}) \in \mathcal{L}_{i+1}} \mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w} \right\rangle \\ &= \sum_{(\ell_1,\ldots,\ell_i) \in \mathcal{L}_i} R_i(\ell_1, \ldots, \ell_i) - \left\langle C, \sum_{(\ell_1,\ldots,\ell_{i+1}) \in \mathcal{R}_{i+1}} \mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w} \right\rangle \end{aligned} \tag{10.20}$$

To proceed, we need the following bound on $\binom{\tau+k}{k}$.

Claim 10.4.8. *For every $i \in \{0, 1, \ldots, k\}$*

$$\binom{\tau + i}{i} \leq \binom{\tau + k}{k} \leq 2^{(\tau-k)/p}$$

Now we first prove Claim 10.4.8. It suffices to bound $\binom{\tau+k}{k}$ by monotonicity of the $\binom{\tau+i}{i}$.

We have

$$\begin{aligned}
\binom{\tau+k}{k} &\le 2^{H_{\mathsf{b}}(k/(k+\tau))\cdot(k+\tau)} && (\tbinom{n}{m} \le 2^{H_{\mathsf{b}}(m/n)\cdot n}) \\
&\le 2^{H_{\mathsf{b}}(c_p^{\mathsf{k}}/(1+c_p^{\mathsf{k}}))\cdot(1+c_p^{\mathsf{k}})\tau} && (k \le c_p^{\mathsf{k}}\cdot\tau) \\
&\le 2^{1/p\cdot(1-c_p^{\mathsf{k}})\cdot\tau} && (H_{\mathsf{b}}(c_p^{\mathsf{k}}/(1+c_p^{\mathsf{k}}))(1+c_p^{\mathsf{k}}) < \tfrac{1}{p}(1-c_p^{\mathsf{k}}) \text{ from our choice of } c_p^{\mathsf{k}}) \\
&\le 2^{(\tau-k)/p}, && (k \le c_p^{\mathsf{k}}\cdot\tau)
\end{aligned}$$

which completes the proof of Claim 10.4.8.

Finally, combining (10.17), (10.19) and (10.20), it follows that

$$\begin{aligned}
&\left|\langle C, \mathcal{W}_i^{h,f;w}\rangle - \langle C, \mathcal{W}_{i+1}^{h,f;w}\rangle\right| \\
&\le \sum_{(\ell_1,\ldots,\ell_i)\in\mathcal{L}_i} |R_i(\ell_1,\ldots,\ell_i) - Q_i(\ell_1,\ldots,\ell_i)| + \left|\left\langle C, \sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}} \mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w}\right\rangle\right| \\
&\le \varepsilon\cdot 2^{\tau}\cdot\binom{\tau+i}{i} + \left\|\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}} \mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w}\right\|_1 && (\|C\|_\infty = 1) \\
&\le \varepsilon^{\Omega(1)} + \left\|\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}} \mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w}\right\|_1, && (10.21)
\end{aligned}$$

where the last inequality follows from Claim 10.4.8 and the fact that $1/\varepsilon > 8^{\tau}$ (recall that $\tau = \lceil \frac{1}{5}\log\varepsilon^{-1}\rceil$).

Now it remains to bound

$$\mathop{\mathbb{E}}_{w\leftarrow U_{nk}} \left\|\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}} \mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w}\right\|_1,$$

which is itself bounded by

$$\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}} \mathop{\mathbb{E}}_{w\leftarrow U_{nk}} \left\|\mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w}\right\|_1.$$

We now fix an $(i+1)$-tuple $(\ell_1,\ldots,\ell_{i+1}) \in \mathcal{R}_{i+1}$. From the definition of $\mathcal{R}_{i+1}$, it holds that $\sum_{j=1}^{i+1}\ell_j > \tau$.

Therefore,

$$\begin{aligned}
\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left\|\mathcal{G}^{h,f;w}_{i+1;(\ell_1,\ldots,\ell_{i+1})}\right\|_1 &\le \mathop{\mathbb{E}}_{r\leftarrow U_m}\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left|\prod_{j=1}^{i+1}h_{=\ell_j}(\tilde{r}_j\oplus w_j)\cdot\prod_{j=i+2}^{k}f(\tilde{r}_j\oplus w_j)\right| && (\|f\|_\infty=1)\\
&= \mathop{\mathbb{E}}_{w\leftarrow U_{(i+1)k}}\mathop{\mathbb{E}}_{r\leftarrow U_m}\left|\prod_{j=1}^{i+1}h_{=\ell_j}(\tilde{r}_j\oplus w_j)\right|\\
&= \prod_{j=1}^{i+1}\|h_{=\ell_j}\|_1\\
&\le \prod_{j=1}^{i+1}2^{-(\ell_j-1)/(p-1)} && \text{(Item (1) of Claim 10.4.3)}\\
&\le 2^{-(\tau-k)/(p-1)}. && (\textstyle\sum_{j=1}^{i+1}\ell_j>\tau \text{ and } i\in\{0,1,\ldots,k-1\})
\end{aligned}$$

Finally, putting everything together, we have

$$\begin{aligned}
\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}}\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left\|\mathcal{G}^{h,f;w}_{i+1;(\ell_1,\ldots,\ell_{i+1})}\right\|_1 &\le 2^{-(\tau-k)/(p-1)}|\mathcal{R}_{i+1}|\\
&\le \binom{\tau+k}{k}(\sigma+1)2^{-(\tau-k)/(p-1)} && \text{(Claim 10.4.4)}\\
&\le (\sigma+1)\cdot 2^{(\tau-k)/p}\cdot 2^{-(\tau-k)/(p-1)}. && \text{(Claim 10.4.8)}\\
&\le 2^{-\Omega(k)}\cdot(\sigma+1) && (p>1)\\
&\le \varepsilon^{\Omega(1)}. && (k=\Omega(\tau)=\Omega(\log\varepsilon^{-1}) \text{ and } \sigma=O(\log\varepsilon^{1-p}))
\end{aligned}$$

This completes the proof of Lemma 10.4.5.

■

In the following remark, we record two very useful facts from the proof of Lemma 10.4.5, which will be very helpful for the next section.

Remark 10.4.9. *For every $i\in\{0,1,\ldots,k-1\}$, the following hold:*

1. *For every $w\in\{0,1\}^{nk}$,*

$$\max_{C\in\mathcal{F}_m}\left|\langle C,\mathcal{W}^{h,f;w}_i\rangle-\langle C,\mathcal{W}^{h,f;w}_{i+1}\rangle\right|\le\varepsilon^{\Omega(1)}+\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}}\left\|\mathcal{G}^{h,f;w}_{i+1;(\ell_1,\ldots,\ell_{i+1})}\right\|_1.$$

2.

$$\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}}\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\left\|\mathcal{G}^{h,f;w}_{i+1;(\ell_1,\ldots,\ell_{i+1})}\right\|_1\le\varepsilon^{\Omega(1)}.$$

Proof of Lemma 10.4.7

Finally, we finish this subsection by proving Lemma 10.4.7 (restated below).

Reminder of Lemma 10.4.7. *For every $j \in [k]$, $C \in \mathcal{F}_m$ and functions $q_1, \ldots, q_{j-1} \colon \{0,1\}^n \to \mathbb{R}$, it holds that*

$$\mathop{\mathbb{E}}_{r \leftarrow U_m}\left[C(r) \cdot \prod_{i=1}^{j-1} q_i(\tilde{r}_i) \cdot \prod_{i=j}^{k} f(\tilde{r}_i \oplus w_i)\right] - \mathop{\mathbb{E}}_{r \leftarrow U_m}\left[C(r) \cdot h(\tilde{r}_j \oplus w_j) \cdot \prod_{i=1}^{j-1} q_i(\tilde{r}_i) \cdot \prod_{i=j+1}^{k} f(\tilde{r}_i \oplus w_i)\right]$$
$$\leq 2\varepsilon \cdot \prod_{i=1}^{j-1} \|q_i\|_\infty.$$

Proof of Lemma 10.4.7. For every $i \in [j-1]$, we let $M_i = \|q_i\|_\infty = \max_{x \in \{0,1\}^n}\{|q_i(x)|\}$. Recall that for $r \in \{0,1\}^m$, we use $\tilde{r}_i$ to denote $\mathcal{G}_{\mathsf{res}}(r)_i$ for every $i \in [k]$.

We claim that, for every $j \in [k]$ and $\alpha \in \{0,1\}^{m-n}$, the following holds

$$\mathop{\mathbb{E}}_{\substack{x \leftarrow U_n \\ r = T_j(x,\alpha)}}\left[C(r) \cdot \prod_{i=1}^{j-1} q_i(\tilde{r}_i) \cdot \prod_{i=j}^{k} f(\tilde{r}_i \oplus w_i)\right] - \mathop{\mathbb{E}}_{\substack{x \leftarrow U_n \\ r = T_j(x,\alpha)}}\left[C(r) \cdot \prod_{i=1}^{j-1} q_i(\tilde{r}_i) \cdot h(\tilde{r}_j \oplus w_j) \cdot \prod_{i=j+1}^{k} f(\tilde{r}_i \oplus w_i)\right]$$
$$\leq 2\varepsilon \cdot \prod_{i=1}^{j-1} M_i \tag{10.22}$$

We will prove (10.22) shortly, but assuming it holds now. The lemma follows directly by taking an expectation over all $\alpha \in \{0,1\}^{m-n}$ (Note that here we used the condition that $T_j(\star)$ is a bijection).

In the rest of the proof we prove (10.22). Now we fix $j \in [k]$ and $\alpha \in \{0,1\}^{m-n}$. We consider a probabilistic algorithm $\mathcal{A}$ specified as follows: $\mathcal{A}$ first samples functions $u_1 \ldots u_{j-1}$ by the following rule. For every $i \in [j-1]$ and $y \in \{0,1\}^n$, we independently set $u_i(y)$ as

$$u_i(y) = \begin{cases} 1 & \text{with probability } \dfrac{1+q_i(y)}{2 \cdot M_i}, \\ -1 & \text{with probability } \dfrac{1-q_i(y)}{2 \cdot M_i}, \\ \text{a uniform random bit in } \{-1,1\} & \text{otherwise.} \end{cases}$$

We can verify that

$$\mathop{\mathbb{E}}_{u_i}[u_i(y)] = \frac{q_i(y)}{M_i} \quad \text{for each } i \in [j-1],$$

where the expectation is taken over the sample distribution of u_i. Then, for an input $x \in \{0,1\}^n$, letting $r = T_j(x, \alpha)$, $\mathcal{A}$ outputs

$$C(T_j(x,\alpha)) \cdot \prod_{i=1}^{j-1} u_i(\tilde{r}_i) \cdot \prod_{i=j+1}^{k} f(\tilde{r}_i \oplus w_i). \tag{10.23}$$

By Definition 10.2.6, the formula (10.23) is an $\mathcal{F}_n$-function with input x for every sampled $(u_1, \ldots, u_{i-1})$ (recall that α is fixed). Therefore, $\mathcal{A}$ can be implemented by a probabilistic $\mathcal{F}_n$-function $\mathcal{D}$. Let $P_{\mathcal{D}}$ be the expectation function of $\mathcal{D}$. For $x \in \{0,1\}^n$, again letting $r = T_j(x, \alpha)$, we have

$$P_{\mathcal{D}}(x) = \frac{1}{\prod_{i=1}^{j-1} M_i} \cdot C(T_j(x,\alpha)) \cdot \prod_{i=1}^{j-1} q_i(\tilde{r}_i) \cdot \prod_{i=j+1}^{k} f(\tilde{r}_i \oplus w_i).$$

We construct another probabilistic $\mathcal{F}$-function $\mathcal{D}'$ such that $P_{\mathcal{D}'}(x) = P_{\mathcal{D}}(x \oplus w_j)$ (this step uses the assumption that $\mathcal{F}$ is typical, and hence it is closed under flipping a subset of inputs). Since h is a $(\delta, 2\varepsilon)_{\ell_{p/(p-1)}}$-witness, we have

$$\langle f - h, P_{\mathcal{D}'} \rangle \le 2\varepsilon. \tag{10.24}$$

Now, it follows from (10.24) that

$$\mathop{\mathbb{E}}_{x \leftarrow U_n}[f(x \oplus w_j) \cdot P_{\mathcal{D}}(x)] - \mathop{\mathbb{E}}_{x \leftarrow U_n}[h(x \oplus w_j) \cdot P_{\mathcal{D}}(x)] \le 2\varepsilon.$$

This is equivalent to (10.22) after scaling both sides by $\prod_{i=1}^{j-1} M_i$: since $\mathcal{G}_{\mathsf{res}}(T_j(x,\alpha))_j = x$, it follows that $f(\tilde{r}_j \oplus w_j) = f(x \oplus w_j)$ and $h(\tilde{r}_j \oplus w_j) = h(x \oplus w_j)$. This finishes the proof of (10.22).

■

10.4.3 Full Derandomization by PRGs for Space-Bounded Computation

Lemma 10.4.2 tells us that for a randomly chosen $(w_1, \ldots, w_k)$, the function $g^w(r)$ is hard with high probability. However, it requires nk bits to describe a list of good w_i. In this section, we will further derandomize Lemma 10.4.2 using PRGs for space-bounded computation.

Branching programs and PRGs for them. We first define read-once branching programs, which captures space-bounded computation.

Definition 10.4.10. *A (probabilistic, read-once, and oblivious) branching program of size s with block size n is a finite state machine with s states, over the alphabet $\{0,1\}^n$ (with a fixed start state, and an arbitrary number of accepting states). Each edge is labeled with a symbol in $\{0,1\}^n$. For every state s and a symbol $\alpha \in \{0,1\}^n$, the edges leaving a and labelled with α is assigned a probability distribution. The computation proceeds as follows. The input is read sequentially, one block of n bits at a time. If the machine is in state x and it reads α, then it chooses an edge leaving x and labeled with α according to its probability, and moves along it.*

From now on, for breivity, we will always use branching programs to refer to read-once and oblivious branching programs. Next we define pseudorandom generators for branching programs.

Definition 10.4.11. *A generator $G\colon \{0,1\}^\ell \to \{0,1\}^{nk}$ is ε-pseudorandom for branching programs of size s and block size n if for every branching program B of size s and block size n, it holds that*

$$|\Pr[B(G(U_\ell)) = 1] - \Pr[B(U_{nk}) = 1]| \leq \varepsilon.$$

Nisan's PRG. We need the well-known construction of Nisan's PRGs fooling branching programs [Nis92].

Theorem 10.4.12 ([Nis92])**.** *For every n and $k \leq 2^n$, there exists a generator*

$$\mathcal{G}^{\mathsf{Nisan}}_{n,k}\colon \{0,1\}^\ell \to \{0,1\}^{nk}$$

such that the following hold:

- *$\mathcal{G}^{\mathsf{Nisan}}_{n,k}$ is 2^{-3n}-pseudorandom for branching programs of size 2^{3n} and block size n.*
- *$\mathcal{G}^{\mathsf{Nisan}}_{n,k}$ has seed length $\ell = O(n \log k)$.*
- *$\mathcal{G}^{\mathsf{Nisan}}_{n,k}$ can be computed in $\mathrm{poly}(n,k)$ time.*

Fully derandomized XOR lemma. Now we are prove our fully derandomized XOR lemma except the moreover part.

Reminder of the main part of Lemma 10.4.1. *Let $\delta \in (0, 0.1)$ and $p \in \mathbb{R}_{>1}$ be two constants. For every sufficiently large $n \in \mathbb{N}$, every $\varepsilon \in [2^{-n}, 1)$ and every function $f\colon \{0,1\}^n \to \{-1,1\}$.*[14] *Let $\mathcal{F}$ be a*

[14] By Lemma 10.1.1, the original XOR Lemma takes $O(n \log 1/\varepsilon)$ bits of inputs. Therefore, we mainly focus on the case that ε is sufficiently small.

typical function collection, and $k = \left\lceil c_p^{\mathsf{k}} \cdot \log \varepsilon^{-1}/5 \right\rceil$. *Suppose the following two conditions hold:*

1. *(**Weak inapproximability by** $\mathsf{Sum} \circ \mathcal{F}_n$.)* $\langle f, C \rangle < (1-\delta)$ *for every* $\mathsf{Sum} \circ \mathcal{F}_n$*-function* C *such that* $\mathsf{complexity}(C) \leq 10 \cdot n/\varepsilon^2$ *and* $\|C\|_p \leq 1$.
2. *(**Existence of an** $\mathcal{F}$**-restrictable generator**.) There is an* $\mathcal{F}$*-restrictable generator* $\mathcal{G}_{\mathsf{res}} \colon \{0,1\}^m \to \{0,1\}^{nk}$ *with seed length* $m \geq n$*, which is computable in* $\mathrm{poly}(m)$ *time.*

Then there is a polynomial-time computable generator $\mathcal{G} \colon \{0,1\}^{m+\ell} \to \{0,1\}^{nk}$ *such that*

$$\mathrm{corr}(f^{\oplus k} \circ \mathcal{G}, \mathcal{F}) \leq \varepsilon^{\Omega(\delta)},$$

where $\ell \leq O(m \log m)$.

Proof of Lemma 10.4.1. We let $\mathcal{G}_{m,k}^{\mathsf{Nisan}} \colon \{0,1\}^{\ell} \to \{0,1\}^{mk}$ be the PRG from Theorem 10.4.12, which fools every branching program of size at most 2^{3m} within error 2^{-3m}. By Theorem 10.4.12 we know that $\ell = O(m \log k)$. We construct from $\mathcal{G}_{m,k}^{\mathsf{Nisan}}$ a generator $\mathcal{G}_2 \colon \{0,1\}^{\ell} \to \{0,1\}^{nk}$ by only keeping the first n-bits of each block of $\mathcal{G}_{m,k}^{\mathsf{Nisan}}(r)$. We construct the final generator as $G(r_1, r_2) \coloneqq \mathcal{G}_{\mathsf{res}}(r_1) \oplus \mathcal{G}_2(r_2)$.

Now we show that the above generator $\mathcal{G}$ satisfies the requirement of Lemma 10.4.1. For this purpose we need to prove result analogous to Lemma 10.4.2. Recall that g^w is defined as $g^w(r) = \prod_{i=1}^{k} f(\mathcal{G}_{\mathsf{res}}(r)_i \oplus w_i)$, and Lemma 10.4.2 says that for a randomly chosen $w \in \{0,1\}^{nk}$, it holds that

$$\mathop{\mathbb{E}}_{w \leftarrow U_{nk}} [\mathrm{corr}(g^w, \mathcal{F})] \leq \varepsilon^{\Omega(\delta)}.$$

We will prove a derandomized version of Lemma 10.4.2.

Lemma 10.4.13. *It holds that*

$$\mathop{\mathbb{E}}_{w \leftarrow \mathcal{G}_2(U_\ell)} [\mathrm{corr}(g^w, \mathcal{F})] \leq \varepsilon^{\Omega(\delta)}. \tag{10.25}$$

We will prove Lemma 10.4.13 shortly. Assuming Lemma 10.4.13 and noting that $\mathcal{F}$ is closed

under restriction, for every $C \in \mathcal{F}$ we have

$$\begin{aligned}\mathop{\mathbb{E}}_{r \leftarrow U_{m+\ell}}[(f^{\oplus k} \circ \mathcal{G})(r) \cdot C(r)] &= \mathop{\mathbb{E}}_{r_2 \leftarrow U_\ell} \mathop{\mathbb{E}}_{r_1 \leftarrow U_m}[g^{\mathcal{G}_2(r_2)}(r_1) \cdot C(r_1, r_2)] \\ &= \mathop{\mathbb{E}}_{w \leftarrow \mathcal{G}_2(U_\ell)} \langle g^w(r_1), C(\cdot, r_2) \rangle \\ &\leq \mathop{\mathbb{E}}_{w \leftarrow \mathcal{G}_2(U_\ell)}[\mathrm{corr}(g^w, \mathcal{F})] \leq \varepsilon^{\Omega(\delta)}.\end{aligned}$$

This establishes the hardness of $f^{\oplus k} \circ \mathcal{G}$ as desired.

■

Now we prove Lemma 10.4.13.

Proof of Lemma 10.4.13. Throughout the proof we will use the same notation as in the proof of Lemma 10.4.2. In particular, the witness function h will play a key role in the proof. We will also follow the structure of its proof structure.

We first recall the following crucial bounds.

Reminder of Lemma 10.4.5. *For every $i \in \{0, 1, \ldots, k-1\}$, it holds that*

$$\mathop{\mathbb{E}}_{w \leftarrow U_{nk}}\left[\max_{C \in \mathcal{F}_m}\left|\langle C, \mathcal{W}_i^{h,f;w}\rangle - \langle C, \mathcal{W}_{i+1}^{h,f;w}\rangle\right|\right] \leq \varepsilon^{\Omega(1)}.$$

Reminder of Lemma 10.4.6. *It holds that*

$$\mathop{\mathbb{E}}_{w \leftarrow U_{nk}}\left\|\mathcal{W}_k^{h,f;w}\right\|_1 \leq \varepsilon^{\Omega(\delta)}.$$

We will derandomize Lemma 10.4.5 and Lemma 10.4.6 by proving the following two lemmas.

Lemma 10.4.14 ($\mathcal{W}_i^{h,f;w}$ and $\mathcal{W}_{i+1}^{h,f;w}$ are indistinguishable by $\mathcal{F}_m$-functions, derandomized)**.** *For every $i \in \{0, 1, \ldots, k-1\}$, it holds that*

$$\mathop{\mathbb{E}}_{w \leftarrow \mathcal{G}_2(U_\ell)}\left[\max_{C \in \mathcal{F}_m}\left|\langle C, \mathcal{W}_i^{h,f;w}\rangle - \langle C, \mathcal{W}_{i+1}^{h,f;w}\rangle\right|\right] \leq \varepsilon^{\Omega(1)}.$$

Lemma 10.4.15 ($\mathcal{W}_k^{h,f;w}$ has small ℓ_1-norm, derandomized)**.** *It holds that*

$$\mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}\left\|\mathcal{W}_k^{h,f;w}\right\|_1 \leq \varepsilon^{\Omega(\delta)}.$$

We can then proceed almost identically as in the proof of Lemma 10.4.2:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}[\mathrm{corr}(g^w,\mathcal{F})]\\
&= \mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}\left[\max_{C\in\mathcal{F}}\{\langle C,\mathcal{W}_0^{h,f;w}\rangle\}\right] && \text{(by definition)}\\
&= \mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}\left[\max_{C\in\mathcal{F}}\left\{\langle C,\mathcal{W}_k^{h,f;w}\rangle + \sum_{i=0}^{k-1}\left(\langle C,\mathcal{W}_i^{h,f;w}\rangle - \langle C,\mathcal{W}_{i+1}^{h,f;w}\rangle\right)\right\}\right]\\
&\leq \mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}\left[\max_{C\in\mathcal{F}}\{\langle C,\mathcal{W}_k^{h,f;w}\rangle\}\right] + k\cdot\varepsilon^{\Omega(1)} && \text{(by Lemma 10.4.14)}\\
&\leq \mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}\left\|\mathcal{W}_k^{h,f;w}\right\|_1 + \varepsilon^{\Omega(1)} && (\|C\|_\infty = 1 \text{ and } k = O(\log 1/\varepsilon))\\
&\leq \varepsilon^{\Omega(\delta)}. && \text{(by Lemma 10.4.15)}
\end{aligned}$$

■

To prove Lemma 10.4.14 and Lemma 10.4.15, we need the following lemma, showing that $\mathcal{G}_2$ can be used to derandomize certain computation.

Lemma 10.4.16. *Let $q_1,\ldots,q_k\colon\{0,1\}^n\to[0,2^n]$ be k functions such that:*

- $\|q_i\|_1\leq 1$ *for every* $i\in[k]$*, and*
- $\prod_{i=1}^k\|q_i\|_\infty\leq 2^{2n}$.

For every $w\in(\{0,1\}^n)^k$, let $\mu^w\colon\{0,1\}^m\to\mathbb{R}$ be such that

$$\mu^w(r)=\prod_{i=1}^k q_i(\tilde{r}_i\oplus w_i).$$

Then, it holds that

$$\mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}\|\mu^w(r)\|_1\leq 2^{-m}+\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}\|\mu^w(r)\|_1 = 2^{-m}+\prod_{i=1}^k\|q_i\|_1.$$

We will prove Lemma 10.4.16 later, but assuming it for now, we finish the proofs of Lemma 10.4.14 and Lemma 10.4.15.

We begin by the proof of Lemma 10.4.15.

Proof of Lemma 10.4.15. We recall the decomposition of $\mathcal{W}_k^{h,f;w}$:

$$\mathcal{W}_k^{h,f;w} := \sum_{(\ell_1,\ldots,\ell_k)\in\mathcal{L}_k} \mathcal{G}_{k;(\ell_1,\ldots,\ell_k)}^{h,f;w}.$$

For each $(\ell_1,\ldots,\ell_k) \in \mathcal{L}_k$, we have

$$\mathcal{G}_{k;(\ell_1,\ldots,\ell_k)}^{h,f;w}(r) = \prod_{i=1}^{k} h_{=\ell_i}(\tilde{r}_i \oplus w_i).$$

From the definition of $\mathcal{L}_k$ and Item (2) of Claim 10.4.3, we have $\prod_{i=1}^{k} \|h_{\ell=i}\|_\infty \le 2^{2n}$. Also, note that $\|h_{=\ell_i}\|_1 \le 1$ for every $i \in [k]$. Setting q_i as $h_{=\ell_i}$ for each $i \in [k]$ and applying Lemma 10.4.16, we have

$$\mathop{\mathbb{E}}_{w\leftarrow G_2(U_\ell)} \|\mathcal{G}_{k;(\ell_1,\ldots,\ell_k)}^{h,f;w}\| \le \mathop{\mathbb{E}}_{w\leftarrow U_{nk}} \|\mathcal{G}_{k;(\ell_1,\ldots,\ell_k)}^{h,f;w}\| + 2^{-m}.$$

Recall that $\varepsilon \ge 2^{-n}$ and $|\mathcal{L}_k| \le \binom{\tau+k}{k} \le \varepsilon^{-1/2}$ since $\tau = \lceil \frac{1}{5} \cdot \log \varepsilon^{-1} \rceil$. Taking a summation over all tuples in $|\mathcal{L}_k|$ completes the proof, since

$$2^{-m} \cdot |\mathcal{L}_k| \le \varepsilon^{\Omega(1)}$$

and

$$\sum_{(\ell_1,\ldots,\ell_k)\in\mathcal{L}_k i} \mathop{\mathbb{E}}_{w\leftarrow U_{nk}} \|\mathcal{G}_{k;(\ell_1,\ldots,\ell_k)}^{h,f;w}\| \le \mathop{\mathbb{E}}_{w\leftarrow U_{nk}} \|\mathcal{G}_k^{h,f;w}\| \le \varepsilon^{\Omega(\delta)},$$

where the last inequality follows from Lemma 10.4.6. ■

Next we prove Lemma 10.4.14.

Proof of Lemma 10.4.14. From Item (1) of Remark 10.4.9, it suffices to bound

$$\sum_{(\ell_1,\ldots,\ell_{i+1})\in\mathcal{R}_{i+1}} \mathop{\mathbb{E}}_{w\leftarrow G_2(U_\ell)} \left\|\mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w}\right\|_1.$$

Fix an $(i+1)$-tuple $(\ell_1,\ldots,\ell_{i+1}) \in \mathcal{R}_{i+1}$, for every $r \in \{0,1\}^m$, we have

$$\mathcal{G}_{i+1;(\ell_1,\ldots,\ell_{i+1})}^{h,f;w}(r) = \prod_{j=1}^{i+1} h_{=\ell_j}(\tilde{r}_j \oplus w_j) \cdot \prod_{j=i+2}^{k} f(\tilde{r}_j \oplus w_j).$$

Now, for each $j \in [i+1]$, we set $q_j = h_{=\ell_j}$ and for each $j \in \{i+2,\ldots,k\}$, we set $q_j = f$. It is straightforward to verify that the functions $q_1,\ldots,q_j$ satisfy the requirement of Lemma 10.4.16.

Applying Lemma 10.4.16, we have

$$\mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)} \|\mathcal{G}^{h,f;w}_{i+1;(\ell_1,\ldots,\ell_{i+1})}(r)\|_1 \le 2^{-m} + \mathop{\mathbb{E}}_{w\leftarrow U_{nk}} \|\mathcal{G}^{h,f;w}_{i+1;(\ell_1,\ldots,\ell_{i+1})}(r)\|_1.$$

Recall that $|\mathcal{R}_{i+1}| \le \binom{\tau+i}{i} \cdot (\sigma+1)$ by Claim 10.4.4, summing up for all $(\ell_1,\ldots,\ell_{i+1}) \in \mathcal{R}_{i+1}$ and applying Item (2) of Remark 10.4.9 completes the proof. ■

Proof of Lemma 10.4.16

Proof of Lemma 10.4.16. For every $j \in [k]$, let $M_j = \|q_j\|_\infty$, it follows that $\prod_{j=1}^k M_j \le 2^{2n}$. Let us consider the following (probabilisitc) branching program, denoted by B.

1. First, sample $r \leftarrow U_m$.
2. Read k blocks $w_1,\ldots,w_k$ in sequence. For every $i \in [k]$, after reading w_i, reject immediately with probability $1 - \frac{q_j(w_j \oplus \tilde{r}_j)}{M_j}$.
3. After reading k blocks without rejection, accept.

Note that B can be implemented by a probabilisitc branching program of size at most 2^{3m}. Now, associate with B an expectation function Q_B, where $Q_B(w_1,\ldots,w_k)$ denotes the probability of B outputting "accept" on input $(w_1,\ldots,w_k)$. By definition, we have

$$\mathop{\mathbb{E}}_{w\leftarrow U_{nk}}[Q_B(w)] = \frac{1}{\prod_{j\in[k]} M_j} \mathop{\mathbb{E}}_{w\leftarrow U_{nk}} \|\mu^w\|_1. \tag{10.26}$$

By Theorem 10.4.12, it follows that

$$\left| \mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}[Q_B(w)] - \mathop{\mathbb{E}}_{w\leftarrow U_{nk}}[Q_B(w)] \right| \le 2^{-3m}. \tag{10.27}$$

Also, observe that

$$\mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)}[Q_B(w)] = \frac{1}{\prod_{j\in[k]} M_j} \mathop{\mathbb{E}}_{w\leftarrow\mathcal{G}_2(U_\ell)} \|\mu^w\|_1. \tag{10.28}$$

We combine (10.26), (10.27) and (10.28) together and note that $\prod_{j\in[k]} M_j \le 2^{2n} \le 2^{2m}$. This completes the proof.

■

Achieving Linear Seed-Length in Slightly Sub-Exponential Error Regime

In the case that $\varepsilon \geq 2^{-n^{1-c}}$ for a Aconstant $c > 0$, it is possible to use a linear-length PRG for space-bounded computation, so that we can reduce the seed length of the pseudorandom instance generator to be linear in n.

The Nisan-Zuckerman PRG. For this purpose, we will need the following PRG for space-bounded computation by Nisan and Zuckerman.

Theorem 10.4.17 ([NZ96]). *For every $c > 0$, the following is true. For every $n \in \mathbb{N}$ and $k \leq O(n)$, there is a generator*

$$\mathcal{G}^{\mathsf{NZ}}_{n,k} \colon \{0,1\}^{\ell} \to \{0,1\}^{nk}$$

such that the following hold:

- *$\mathcal{G}^{\mathsf{NZ}}_{n,k}$ is $2^{-3n^{1-c}}$-pseudorandom for branching programs of size 2^{3n} and block size n.*
- *$\mathcal{G}^{\mathsf{NZ}}_{n,k}$ has seed length $\ell = O_c(n)$.*
- *$\mathcal{G}^{\mathsf{NZ}}_{n,k}$ can be computed in $\mathrm{poly}(n,k)$ time.*

Fully derandomized XOR lemma with linear seed length. We now prove the linear-length seed generators. (*i.e.* the moreover part in Lemma 10.4.1.)

Reminder of the moreover part of Lemma 10.4.1. *Let $\delta, c \in (0, 0.1)$ and $p \in \mathbb{R}_{>1}$ be three constants. For every sufficiently large $n \in \mathbb{N}$, every $\varepsilon \in [2^{-n^{1-c}}, 1)$ and every function $f \colon \{0,1\}^n \to \{-1,1\}$. Let $\mathcal{F}$ be a typical function collection and let $k = \left\lceil c_p^{\mathsf{k}} \cdot \log \varepsilon^{-1} / 5 \right\rceil$. Suppose the following two conditions hold:*

1. *(**Weak inapproximability by** $\mathsf{Sum} \circ \mathcal{F}_n$.) $\langle f, C \rangle < (1-\delta)$ for every $\mathsf{Sum} \circ \mathcal{F}_n$-function C such that $\mathsf{complexity}(C) \leq 10 \cdot n/\varepsilon^2$ and $\|C\|_p \leq 1$.*
2. *(**Existence of an $\mathcal{F}$-restrictable generator**.) There is an $\mathcal{F}$-restrictable generator $\mathcal{G}_{\mathsf{res}} \colon \{0,1\}^m \to \{0,1\}^{nk}$ with seed length $m \geq n$, which is computable in $\mathrm{poly}(m)$ time.*

Then there is a polynomial-time computable generator $\mathcal{G} \colon \{0,1\}^{m+\ell} \to \{0,1\}^{nk}$ such that

$$\mathrm{corr}(f^{\oplus k} \circ \mathcal{G}, \mathcal{F}) \leq \varepsilon^{\Omega(\delta)},$$

where $\ell \leq O_c(m)$.

Proof Sketch. We let $\mathcal{G}_2 \colon \{0,1\}^{\ell} \to \{0,1\}^{nk}$ be the PRG from Theorem 10.4.17, which can fool every branching program of size at most 2^{3m} within error $2^{-3m^{1-c}}$. It follows that $\ell \leq O_c(m)$. We

construct the final generator as $\mathcal{G}(r_1, r_2) := \mathcal{G}_{\mathsf{res}}(r_1) \oplus \mathcal{G}_2(r_2)$. To show that $\mathcal{G}$ works, we use a similar argument as in the proof of Lemma 10.4.1 except one difference: when applying the full derandomization based on the partial derandomization, we use the following lemma to replace Lemma 10.4.16.

Lemma 10.4.18 (A variant of Lemma 10.4.16)**.** *Let $q_1, \dots, q_k \colon \{0,1\}^n \to [0, 2^n]$ be k functions such that*

- $\|q_i\|_1 \leq 1$ *for every* $i \in [k]$*, and*
- $\prod_{i=1}^{k} \|q_i\|_\infty \leq \varepsilon^{-p}$.

For every $w \in (\{0,1\}^n)^k$, let $\mu^w \colon \{0,1\}^m \to \mathbb{R}$ be such that

$$\mu^w(r) = \prod_{i=1}^{k} q_i(\tilde{r}_i \oplus w_i).$$

Then, it holds that

$$\mathop{\mathbb{E}}_{w \leftarrow \mathcal{G}_2(U_\ell)} \|\mu^w(r)\|_1 \leq 2^{-m^{1-c}} + \mathop{\mathbb{E}}_{w \leftarrow U_{nk}} \|\mu^w(r)\|_1 = 2^{-m^{1-c}} + \prod_{i=1}^{k} \|q_i\|_1.$$

The proof of Lemma 10.4.18 is also analogous to Lemma 10.4.16.

■

10.5 Weak-inapproximability by Linear Sums from Non-trivial Circuit-analysis Algorithms

In this section we show that for a function collection $\mathcal{F}$ which admits a sufficient circuit-analysis algorithm, one can use Williams' algorithmic method [Wil13a, Wil18, CW19, CLW20] to construct a hard function f which cannot be weak-approximated by $\mathsf{Sum} \circ \mathcal{F}$-functions. In later sections, this will be combined with our new derandomized XOR lemma to construct strong average-case hard functions against $\mathcal{F}$.

Our connection works for every typical function collections. We first summarize the necessary requirements for the target function collection below.

Definition 10.5.1 (Applicable function collections)**.** *Let $S(n) \geq n^{\omega(1)}$ be a non-decreasing time-constructible function, and $\mathcal{F} = \bigcup_{n \in \mathbb{N}_{\geq 1}} \mathcal{F}_n$ be a function collection. We say that $\mathcal{F}$ is $S(n)$-applicable, if the following hold:*

1. *There is a* #SAT *algorithm for* $\mathsf{AND}_4 \circ \mathcal{F}_n$*-functions that runs in* $O(2^n/S(n))$ *time, where* $\mathsf{AND}_4 \circ \mathcal{F}_n$ *denotes the subset of functions which can be computed by taking an* AND *of four* $\mathcal{F}_n$*-functions.*
2. *For every* $n \in \mathbb{N}_{\geq 1}$ *and* $S \subseteq [n]$*, the function* $\chi_S(x) := \prod_{i \in S}(-1)^{x_i}$ *is in* $\mathcal{F}_n$*. In other words,* $\mathcal{F}$ *contains all the* parities*.*
3. $\mathcal{F}$ *is closed under negation.*

Now we state the main theorem of this section, which claims that for every $S(n)$-applicable function collection $\mathcal{F}$, one can construct an E^{NP} function f, that is weakly-inapproxiamble by $\mathcal{F}$-functions.

Theorem 10.5.2. *There are absolute constants* $\alpha, \delta \in (0,1)$ *and* $K \geq 1$ *such that the following hold. Let* $\mathcal{F}$ *be an* $S(n)$*-applicable collection. There is an* E^{NP} *machine which, for every sufficiently large* n*, on input* 1^n*, outputs in* $2^{O(n)}$ *time a Boolean function* $f\colon \{0,1\}^\ell \to \{-1,1\}$ *where* $\ell \in [n, Kn]$ *such that one of the following holds.*

1. f *cannot be computed by* $S(\ell)^\alpha$*-size general circuits.*
2. f *is hard in the following sense: for every* $\mathsf{Sum} \circ \mathcal{F}_\ell$*-functions* H *such that* $\mathsf{complexity}(H) \leq S(\ell)^{3\alpha}$ *and* $\|H\|_4 \leq 1$*, it holds that*
$$\langle f, H \rangle < (1 - \delta).$$

The rest of this section is devoted to the proof of Theorem 10.5.2 and is organized as follows: In Section 10.5.1 we introduced some previous results which will be crucial to our proof of Theorem 10.5.2. In Section 10.5.2 we design a "cheating algorithm" $\mathcal{A}_{\mathsf{cheat}}$ which attempts to break a certain NTIME hierarchy theorem, this part is very similar to the proof of [CLW20, Theorem 1.2], and is also crucial to our proof of Theorem 10.5.2. In Section 10.5.3, we analyze $\mathcal{A}_{\mathsf{cheat}}$ and conclude the proof of Theorem 10.5.2.

10.5.1 Preliminaries

We will need some technical ingredients from the literature.

Robustly-often NTIME hiearchy theorem and P^{NP} refuter for it. We start with the following robustly-often hard $\mathsf{NTIME}[T(n)]$ language with a corresponding refuter algorithm for it.

Theorem 10.5.3 (Robustly-often NTIME hierarchy [FS17])**.** *For every polynomial* $T(n) = n^K$ *and for some constant* $k \geq 1$*, there exists a language* $L \in \mathsf{NTIME}[T(n)]$ *such that, for any* $L' \in \mathsf{NTIME}[o(T(n))]$*,*

for all but finitely many n, there exists $m \in [n, n + T(n)]$ such that L and L' cannot agree on all inputs of length m.

Theorem 10.5.4 (Refuter for the robustly-often NTIME hierarchy [CLW20, Theorem 4.8])**.** *For every polynomial $T(n) = n^K$ for some constant $k \geq 1$, there is an $\mathsf{NTIME}[T(n)]$ machine $\mathcal{A}^T_{\mathsf{FS}}$ and a P^{NP} algorithm $\mathcal{R}^T$ such that:*

1. ***Input.*** *The input for $\mathcal{R}^T$ is a pair $(M, 1^n)$ with the promise that M is nondeterministic Turing machine runs in $o(T(n))$ time.*
2. ***Output.*** *For every fixed M and all large enough n, $\mathcal{R}^T(M, 1^n)$ outputs a string x such that $|x| \in [n, n + T(n)]$ and $\mathcal{R}^T(x) \neq M(x)$.*

We remark that the original theorems in [FS17] and [CLW20] apply to a large class of functions $T(n)$, but here we will just state them for the special case that $T(n) = n^K$, since this is all we need in the proofs.

Efficient Construction of PCPs. We recall the following Probabilistically Checkable Proofs (PCP) systems and PCP of Proximity systems, which are also used in [CR20] and [CLW20].

Theorem 10.5.5 ([BV14])**.** *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where $|x| = n$. Given $x \in \{0,1\}^n$, one can output in $\mathrm{poly}(n, \log T)$ time circuits $Q\colon \{0,1\}^r \to \{0,1\}^{rt}$ for $t = \mathrm{poly}(r)$ and $R\colon \{-1,1\}^t \to \{0,1\}$ such that:*

- ***Proof length.*** *$2^r \leq T \cdot \mathrm{polylog} T$.*
- ***Completeness.*** *If there is a $y \in \{0,1\}^{T(n)}$ such that $M(x, y)$ accepts then there is a map $\pi\colon \{0,1\}^r \to \{-1,1\}$ such that for all $z \in \{0,1\}^r$, $R(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = Q(z)$.*
- ***Soundness.*** *If no $y \in \{0,1\}^{T(n)}$ causes $M(x, y)$ to accept, then for every map $\pi\colon \{0,1\}^r \to \{-1,1\}$, at most $\frac{2^r}{n^{10}}$ distinct $z \in \{0,1\}^r$ have $R(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = Q(z)$.*
- ***Complexity.*** *Q is a projection, i.e., each output bit of Q is a bit of input, the negation of a bit, or a constant. R is a* 3CNF*.*

Note that this is an extremely efficient PCP, where the 3CNF R and the projection Q collectively form the verifier for the PCP. The following lemma from [CW19, VW20] is a slight modification of the probabilistically checkable proof of proximity (PCPP) system in [BSGH+06].

Theorem 10.5.6 ([CW19, VW20])**.** *There are constants $0 < s_{\mathsf{pcpp}} < c_{\mathsf{pcpp}} < 1$ and a polynomial-time transformation that, given a circuit D on n inputs of size $m \geq n$, outputs a 2-SAT instance F on the variable set $\mathcal{Y} \cup \mathcal{Z}$ where $|\mathcal{Y}| \leq \mathrm{poly}(n), |\mathcal{Z}| \leq \mathrm{poly}(m)$, and the following hold for all $x \in \{0,1\}^n$:*

- *If* $D(x) = 1$, *then* $F\big|_{\mathcal{Y}=\mathsf{Enc}(x)}$ *on variable set* $\mathcal{Z}$ *has a satisfying assignment* $\mathcal{Z}_x$ *such that at least* c_{pcpp}*-fraction of the clauses are satisfied. Furthermore, there is a* $\mathrm{poly}(m)$ *time algorithm that given* x *outputs* $\mathcal{Z}_x$.
- *If* $D(x) = 0$, *then there is no assignment to the* $\mathcal{Z}$ *variables in* $F\big|_{\mathcal{Y}=\mathsf{Enc}(x)}$ *satisfies more than* s_{pcpp}*-fraction of the clauses.*

Moreover, the number of clauses in the 2-SAT instance F *is a power of* 2, *and for each* $i \in [|\mathcal{Y}|]$, $\mathsf{Enc}_i(x)$ *is a parity function depending on at most* $n/2$ *bits of* x.

Arithmetization. In the following, we will frequently apply arithmetization to these PCP verifiers. For input Boolean values to PCP verifier, we always interpret Boolean True and False as real -1 and 1 respectively. This is consistent with the proof fed into it (recall that PCP verifiers get oracle Boolean functions as proof.). For output of PCP verifiers, we always interpret Boolean True and False (Accept and Reject) as real 1 and 0 respectively. By doing so, the expectation of output of PCP verifier is naturally its probability of acceptance. See also Section 10.3 for more details of the arithmetization.

10.5.2 Description of the Cheating Algorithm $\mathcal{A}_{\mathsf{cheat}}$

Now, we first describe a nondeterministic algorithm $\mathcal{A}_{\mathsf{cheat}}$ to speed up the computation $\mathcal{A}^T_{\mathsf{FS}}(x)$, where $\mathcal{A}^T_{\mathsf{FS}}$ is defined in Theorem 10.5.4. We will borrow most notation from [CLW20]. Our algorithm $\mathcal{A}_{\mathsf{cheat}}$ is basically the same as the algorithm $\mathcal{A}_{\mathsf{PCPP}}$ used in the proof of [CLW20, Theorem 1.2].

Set up. The algorithm $\mathcal{A}_{\mathsf{cheat}}$ is parameterized by two sufficiently small constants $\alpha, \delta \in (0,1)$ and a sufficiently large constant $K \geq 1$ specifying the time bound $T(n) = n^K$. We assume that K is large enough so that the PCP construction in Theorem 10.5.5 can be done in $\mathrm{poly}(n) \leq n^{K/2}$ time.

Using PCP first. On an input z of length n, $\mathcal{A}_{\mathsf{cheat}}$ applies the PCP from Theorem 10.5.5 to the computation $\mathcal{A}^T_{\mathsf{FS}}(x)$, and obtains an oracle circuit VPCP_z. Recall that both of VPCP_z and its oracle take inputs of length $\ell = \ell(n) = \log(T(n)) + O(\log\log T(n))$. Theorem 10.5.5 implies the following.

Claim 10.5.7. *The following statements hold.*

1. *If* $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$, *then there an oracle* $\mathcal{O}$ *such that* $\mathsf{VPCP}^{\mathcal{O}}_z(x) = 1$ *for every* $x \in \{0,1\}^\ell$.

2. *It* $\mathcal{A}_{\mathsf{FS}}^{T}(z) = 0$*, then for every oracle* $\mathcal{O}$*, it holds that* $\Pr_{x\in\{0,1\}^{\ell}}[\mathsf{VPCP}_z^{\mathcal{O}}(x) = 1] \le \frac{1}{n^{10}} \le \frac{1}{\ell^{10}}$.

Then $\mathcal{A}_{\mathsf{cheat}}$ guesses a general circuit $C\colon \{0,1\}^{\ell} \to \{-1,1\}$ of size at most $S(\ell)^{\alpha}$ as the oracle for VPCP_z. Feeding C into VPCP_z, it obtains the circuit VPCP_z^C, with circuit size bounded above by $\mathrm{poly}(|C|) = S(\ell)^{O(\alpha)}$. By Claim 10.5.7, the algorithm $\mathcal{A}_{\mathsf{cheat}}$ needs to distinguish between the following two cases:

1. $\mathsf{VPCP}_z^C\colon \{0,1\}^{\ell} \to \{0,1\}$ is a tautology.
2. VPCP_z^C accepts at most $2^{\ell}/\ell^{10}$ many inputs.

The PCPP construction and notation. $\mathcal{A}_{\mathsf{cheat}}$ then applies the PCPP from Theorem 10.5.6 to the circuit VPCP_z^C. It produces a 2SAT instance Φ with $m = \mathrm{poly}(|C|) = S(\ell)^{O(\alpha)}$ many clauses over the variable set $\mathcal{Y} \cup \mathcal{Z}$, as well as an encoding function $\mathsf{Enc}\colon \{0,1\}^{\ell} \to \{0,1\}^{|\mathcal{Y}|}$. For $(s,t) \in [|\mathcal{Y}|] \times [|\mathcal{Z}|]$, we use $\mathcal{Y}_s$ and $\mathcal{Z}_t$ to denote the s-th variable in $\mathcal{Y}$ and the t-th variable in $\mathcal{Z}$, respectively.

Recall by Theorem 10.5.6 that s is a power of two. For brevity, we set $r = \ell + 1 + \log m$.

To elegantly discuss the algorithm and its analysis, we introduce some useful notation. Let the clauses of Φ be $(\mathsf{Cons}_i)_{i=1}^{m}$, where each of Cons_i involves 2 variables from $\mathcal{Y} \cup \mathcal{Z}$. For each clause Cons_i, it extends to a degree-2 polynomial, denoted by $\widetilde{\mathsf{Cons}}_i$.[15] For every $i \in [m]$ and $j \in [2]$, we set an indicator $\mathcal{T}_{i,j} \in \mathcal{Y} \cup \mathcal{Z}$ to indicate the j-th variable in Cons_i.

1. By a "real-valued proof" we mean a pair of two lists of proof functions (Y, Z) for PCPP, where $Y = (Y_s)_{s\in[|\mathcal{Y}|]}$, $Z = (Z_t)_{t\in[|\mathcal{Z}|]}$ and each of Y_s and Z_t is a function from $\{0,1\}^{\ell}$ to $\mathbb{R}$. Based on (Y, Z), we define the following terminologies:
   - Recall that each clause Cons_i involves two variables. We define indicators $T_{i1}^{(Y,Z)}$ and $T_{i2}^{(Y,Z)}$ to indicate the corresponding functions in (Y, Z).
   - Recall that each clause Cons_i extends to a polynomial $\widetilde{\mathsf{Cons}}_i$. We define $F_i^{(Y,Z)} := \widetilde{\mathsf{Cons}}_i(T_{i1}^{(Y,Z)}, T_{i2}^{(Y,Z)})$.

   Note that these objects all depend on the given proof (Y, Z), when the context is clear, we also omit the superscript, and simply write them as T_{ij} and F_i.
2. By a "Boolean-valued proof" we mean a pair of two lists of proof functions $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ where $\widehat{Y}_s(x) = \mathsf{Enc}_s(x)$ for every $x \in \{0,1\}^{\ell}$ and $s \in [|\mathcal{Y}|]$, $\widehat{Z} = (\widehat{Z}_t)_{t\in[|\mathcal{Z}|]}$, and each $\widehat{Z}_t$ is a function from $\{0,1\}^{\ell} \to \{-1,1\}$. Recall that $\mathsf{Enc}\colon \{0,1\}^{\ell} \to \{-1,1\}^{|\mathcal{Y}|}$ is the corresponding function in Theorem 10.5.6. Similar to the case of real-valued proofs, the proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ induces $\widehat{T}_{ij}^{(\widehat{Y},\widehat{Z})}$ and $\widehat{F}_i^{(\widehat{Y},\widehat{Z})}$. When the context is clear, we omit the superscript

[15] For inputs to $\widetilde{\mathsf{Cons}}_i$, we identify Boolean False and True as real 1 and -1 respectively. For outputs of $\widetilde{\mathsf{Cons}}_i$, we identify Boolean False and True as real 0 and 1 respectively.

and write them as $\widehat{T}_{ij}$ and $\widehat{F}_i$.

To clarify, we always use (Y, Z) to denote a real-valued proof, and $(\widehat{Y}, \widehat{Z})$ to denote a Boolean-valued proof. We summarize the properties of the PCPP construction in the following claim. Recall that s_{pcpp} and c_{pcpp} are the soundness and completeness parameters in Theorem 10.5.6.

Claim 10.5.8. *The following statements hold.*

1. *If* $\mathsf{VPCP}_z{}^C$ *is a tautology, then there is a Boolean proof* $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ *such that*

$$\mathop{\mathbb{E}}_{x \leftarrow U_\ell} \mathop{\mathbb{E}}_{i \in [m]} \widehat{F}_i(x) \geq c_{\mathsf{pcpp}}.$$

2. *If* $\mathsf{VPCP}_z{}^C(x) = 1$ *for at most a* $\frac{1}{\ell^{10}} \leq o(1)$ *fraction of* x*, then for every sufficiently large* n*, for every Boolean proof* $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$*, we have*

$$\mathop{\mathbb{E}}_{x \leftarrow U_\ell} \mathop{\mathbb{E}}_{i \in [m]} \widehat{F}_i(x) < c_{\mathsf{pcpp}} - \frac{9}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).$$

Guess proof function for PCPP**.** Next, $\mathcal{A}_{\mathsf{cheat}}$ guesses a $\mathsf{Sum} \circ \mathcal{F}_r$-functions of complexity at most $S(r)^{3\alpha}$, denoted by $H \colon \{0,1\}^{\log m} \times \{0,1\}^1 \times \{0,1\}^\ell \to \mathbb{R}$. We identify the first $\log m$ bits of inputs as an index in $[m]$, so that the first $\log m + 1$ bits can identify a variable $\mathcal{T}_{i,j}$. Based on H, we construct a real-valued proof (Y, Z) as

$$Y_s(x) := \mathop{\mathbb{E}}_{i,j:\mathcal{T}_{ij}=\mathcal{Y}_s} H(i,j,x) \quad \text{for } s \in [|\mathcal{Y}|],$$
$$Z_t(x) := \mathop{\mathbb{E}}_{i,j:\mathcal{T}_{ij}=\mathcal{Z}_t} H(i,j,x) \quad \text{for } t \in [|\mathcal{Z}|].$$

Recall that we defined $T_{i,j} \in Y \cup Z$ as the circuit corresponds to variable $\mathcal{T}_{i,j} \in \mathcal{Y} \cup \mathcal{Z}$. We define

$$P_{ij}(x) = \begin{cases} (1 + T_{ij}(x))^2 (1 - T_{ij}(x))^2, & \text{if } T_{ij} \in Z, \\ (\mathsf{Enc}_s(x) - T_{ij}(x))^2, & \text{if } T_{ij} \in Y. \end{cases} \tag{10.29}$$

Verification. Finally, let $\mathcal{A}_{\mathsf{cheat}}$ verify the following:

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} P_{i,j}(x) \le \delta, \tag{10.30}$$

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} H(i,j,x)^2 \le 1, \tag{10.31}$$

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} F_i(x) \ge c_{\mathsf{pcpp}} - \frac{1}{2}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}). \tag{10.32}$$

$\mathcal{A}_{\mathsf{cheat}}$ accepts if and only if all of the conditions above hold. Note that each of those summations above can reduce to solving $S(\ell)^{O(\alpha)}$ many #SAT tasks for $\mathsf{AND}_4 \circ \mathcal{F}$-functions[16].

Running time of $\mathcal{A}_{\mathsf{cheat}}$. We verify that the algorithm $\mathcal{A}_{\mathsf{cheat}}$ runs in $\mathsf{NTIME}[o(T(n))]$ for small enough α: the construction of VPCP_z requires $\mathrm{poly}(n) < T(n)^{1/2}$ time for sufficiently large constant K; the PCPP construction runs in $S(\ell)^{O(1)}$ time; the guess and verification run in $2^r/S(r) \cdot S(r)^{O(\alpha)} \le 2^\ell/S(\ell)^{\Omega(1)} \le o(T(n))$ time for small enough α. (Recall that $r = \ell + O(\alpha \log S(\ell))$, and $S(\ell) \ge \ell^{\omega(1)}$.) This completes the description of algorithm.

10.5.3 Proof of Theorem 10.5.2

We state two crucial properties of $\mathcal{A}_{\mathsf{cheat}}$ below. First, we observe that the algorithm $\mathcal{A}_{\mathsf{cheat}}$ only makes one-sided error.

Lemma 10.5.9. *For every small enough constants $\alpha, \delta \in (0,1)$ and for every sufficiently large constant $K \ge 1$, the following holds: $\mathcal{A}_{\mathsf{cheat}}(z) \le \mathcal{A}^T_{\mathsf{FS}}(z)$ for all but finitely many inputs z.*

The proof of Lemma 10.5.9 can be found in Section 10.8.

Combining Lemma 10.5.9 with Theorem 10.5.4. We conclude that for every sufficiently large n, one can apply the refuter $\mathcal{R}^T$ to find an input z of length $|z| \in [n, n^K + n]$ such that $\mathcal{A}_{\mathsf{cheat}}(z) = 0$ and $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$. Considering one such z, by $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$ and Item (1) of Claim 10.5.7 we know that there is an oracle $\mathcal{O}$ such that $\mathsf{VPCP}^{\mathcal{O}}_z$ is a tautology. If there is no circuit C of size at most $S(\ell)^\alpha$ such that VPCP^C_z is a tautology, then in particular it implies that this $\mathcal{O}$ cannot be computed by circuits of size at most $S(\ell)^\alpha$. This proves the Case (1) in Theorem 10.5.2 assuming that no circuit C of size at most $S(\ell)^\alpha$ can make VPCP^C_z a tautology.

[16] This is the same as the algorithm used in [CLW20]. We refer interested readers to [CLW20, Theorem 4.8] for the details about how this works.

In the following, we will show Case (2) in Theorem 10.5.2 holds if there is a circuit C of size at most $S(\ell)^\alpha$ such that VPCP_z^C is a tautology. This will finish the proof of Theorem 10.5.2 as at least one of Case (1) or Case (2) holds regardless of such a circuit C exists or not.

Lemma 10.5.10. *For every small enough constants $\alpha, \delta \in (0,1)$ and for every sufficiently large constant $K \geq 1$, the following holds: Suppose for some input z of length n such that $\mathcal{A}_{\mathsf{cheat}}(z) = 0$, there is a circuit of size at most $S(\ell)^\alpha$ such that VPCP_z^C is a tautology with $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ being its correct Boolean-valued proof. Then the function $H^{\widehat{Y},\widehat{Z}}$ defined by*

$$\begin{aligned} H^{\widehat{Y},\widehat{Z}} \colon \{0,1\}^{\log(m)+1+\ell} &\to \{-1,1\} \\ (i,j,x) &\mapsto \widehat{T}_{ij}(x) \end{aligned} \tag{10.33}$$

is hard in the following sense: letting $r = \log(m) + 1 + \ell$, for every $\mathsf{Sum} \circ \mathcal{F}_r$-functions H such that $\mathsf{complexity}(H) \leq S(r)^{3\alpha}$ and $\|H\|_4 \leq 1$, it holds that

$$\langle H^{\widehat{Y},\widehat{Z}}, H \rangle < (1 - \delta/5). \tag{10.34}$$

The proof of Lemma 10.5.10 can be found in Section 10.8.

We are finally ready to prove Theorem 10.5.2.

Proof of Theorem 10.5.2. We design the E^{NP} algorithm $\mathcal{A}_{\mathsf{hard}}$ as follows. Let n be a sufficiently large input length. On an input 1^n, $\mathcal{A}_{\mathsf{hard}}$ sets $m = 2^{n/K}$. By Theorem 10.5.4, $\mathcal{A}_{\mathsf{hard}}$ can find in $\mathrm{poly}(m) \leq 2^{O(n)}$ time (with access to an NP oracle) an input z of length $|z| \in [m, m + m^K]$ such that $\mathcal{A}_{\mathsf{FS}}^T(z) = 1$ and $\mathcal{A}_{\mathsf{cheat}}(z) = 0$.

Consider the PCP system VPCP_z. Since $\mathcal{A}_{\mathsf{FS}}^T(z) = 1$, by Item (1) of Claim 10.5.7, there is an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ for $\ell = K \log |z| + O(\log \log |z|)$ such that $\mathsf{VPCP}_z^{\mathcal{O}}$ is a tautology. Recall that $m = 2^{n/K}$ and $|z| \in [m, m + m^K]$, it follows that $n \leq \ell \leq n(K+1)$. $\mathcal{A}_{\mathsf{hard}}$ then construct the lexicographically first such oracle, still denoted by $\mathcal{O}$ for convenience, which can be found with the help of an NP oracle in $\mathrm{poly}(m) \leq 2^{O(n)}$ time. Depending on whether $\mathcal{O}$ can be computed by small circuits or not, we have the following two cases:

1. ($\mathcal{O}$ **is hard**.) That is, $\mathcal{O}$ cannot be computed by a (general) circuit of size at most $S(\ell)^\alpha$. In this case, $\mathcal{O}$ induces a hard function on ℓ-bit inputs, and Case (1) of Theorem 10.5.2 holds. We let $\mathcal{A}_{\mathsf{hard}}$ output $\mathcal{O}$.
2. ($\mathcal{O}$ **is easy**.) That is, $\mathcal{O}$ can be computed by a (general) circuit of size at most $S(\ell)^\alpha$. In this

case, $\mathcal{A}_{\mathsf{hard}}$ first constructs the lexicographically first such circuit C (with access to an NP oracle in $2^{O(n)}$ time). Feeding C into the oracle circuit VPCP_z, $\mathcal{A}_{\mathsf{hard}}$ then obtains a circuit VPCP_z^C. Consider the PCP of Proximity proof (the 2SAT instance) Φ for VPCP_z^C over variables $(\mathcal{Y}, \mathcal{Z})$. Since VPCP_z^C is a tautology, we have a list of proof functions $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ such that for every $x \in \{0,1\}^\ell$, at least c_{pcpp}-fraction of clauses are satisfied by the assignment $(\mathcal{Y}, \mathcal{Z}) = (\widehat{Y}(x) = \mathsf{Enc}(x), \widehat{Z}(x))$. Now, given that $\mathcal{A}_{\mathsf{cheat}}(z) = 0$, it follows from Lemma 10.5.10 that the function $H^{\widehat{Y},\widehat{Z}}$ defined by

$$H^{\widehat{Y},\widehat{Z}}(i,j,x) := \widehat{T}_{ij}(x)$$

satisfies Case (2) of Theorem 10.5.2 statement[17]. We let $\mathcal{A}_{\mathsf{hard}}$ output $H^{\widehat{Y},\widehat{Z}}$.

■

10.6 Strong Correlation Bounds against $\mathbb{F}_2$-Polynomials

In this section, we apply Theorem 10.5.4 to prove Theorem 1.7.4 (the strong correlation bound against $\mathbb{F}_2$-polynomials). For an $\mathbb{F}_2$-polynomial $P\colon \mathbb{F}_2^n \to \mathbb{F}_2$, we will consider its corresponding Boolean function (recall that we take Boolean functions to be functions from $\{0,1\}^*$ to $\{-1,1\}$) defined by $B_P(x) := (-1)^{P(x)}$ for every $x \in \{0,1\}^n$.

10.6.1 Special Collections of Functions Extending F_2-Polynomials

We will work with two special collections of functions, which contains low-degree $\mathbb{F}_2$-polynomials as a sub-collection. We give their definitions below.

Definition 10.6.1. *For every $n, d, p \in \mathbb{N}_{\geq 1}$ such that $d, p \leq n$, we define the n-bit function collection $\mathcal{H}_n^{d,p}$ as the set of all functions $C\colon \{0,1\}^n \to \{-1,1\}$ that can be written as*

$$C(x) = (-1)^{P(x)} \cdot Q(x_{\leq n-p}),$$

where $P\colon \mathbb{F}_2^n \to F_2$ is an $\mathbb{F}_2$-polynomial of degree at most d and Q is an arbitrary function from $\{0,1\}^{n-p}$ to $\{-1,1\}$ (recall that $x_{\leq n-p}$ is the length-$(n-p)$ prefix of x). For convenience, we say that $P(x)$ and $Q(x)$ are the polynomial part and the free part of C, respectively.

[17] Note that the inapproximability parameter here is $(1-\delta/5)$ instead of $(1-\delta)$. This is OK since Theorem 10.5.2 only claims the existence of one such $\delta > 0$.

We also define a function collection

$$\mathcal{F}^{n,d,p} := \bigcup_{q=n}^{\infty} \mathcal{H}_q^{d,(q-n)+p}.$$

Clearly, $\mathcal{H}_n^{d,p}$ and $\mathcal{F}^{n,d,p}$ are closed under XOR (which is multiplication over $\{-1,1\}$). In the following, we will state and prove the following two crucial properties of the collections:

1. $\mathcal{H}_n^{d,p}$ admits a non-trivial #SAT algorithm, as shown in Lemma 10.6.2.
2. There are $\mathcal{F}^{n,d,p}$-restrictable generators with relatively short seeds, as shown in Lemma 10.6.3.

Lemma 10.6.2. *For every $n, d, p \in \mathbb{N}_{\geq 1}$ such that $d, p \leq n$, there is an algorithm which given any $\mathcal{H}_n^{d,p}$-function C can evaluate*[18] $\sum_{x\in\{0,1\}^n} C(x)$ *in* $2^{n-\Omega(\min(n/d,p))+O(\log(n))}$ *time.*

Lemma 10.6.3. *For every $n, d, p \in \mathbb{N}_{\geq 1}$ such that $d, p \leq n$ and for every $k \in \mathbb{N}_{\geq 1}$, there is an $\mathcal{F}^{n,d,p}$-restrictable generator $\mathcal{G}\colon \{0,1\}^m \to \{0,1\}^{nk}$ with seed length $m = n + (k-1)p$.*

In the rest of this subsection we prove Lemma 10.6.2 and Lemma 10.6.3 separately.

Proof of Lemma 10.6.2. The new algorithm is a slight adaptation of the algorithm from [Wil18, Theorem 6.1]. It is instructive to describe the original algorithm here. Given a degree-d $\mathbb{F}_2$-polynomial P, the algorithm from [Wil18] computes the sum $\sum_{x\in\{0,1\}^n} P(x)$ over *integers* as follows:

1. Choose K to be a sufficiently large constant and let $\delta = \frac{1}{K\cdot d}$. Let $\ell = \lfloor \delta n \rfloor$. We will work with the ring $\mathbb{Z}_{2^\ell}$. Recall that there is a modulus-amplifying polynomial ([Yao90, BT94]) $V(x)$ of degree $2\ell - 1$ such that the following hold: (1) if $x \equiv 0 \pmod 2$ then $V(x) \equiv 0 \pmod{2^\ell}$; (2) if $x \equiv 1 \pmod 2$ then $V(x) \equiv 1 \pmod{2^\ell}$.
2. By the modulus-amplifying property of V, for every $(x_1, \ldots, x_{n-\ell+1}) \in \{0,1\}^{n-\ell+1}$, it holds that

$$\sum_{(x_{n-\ell+2},\ldots,x_n)\in\{0,1\}^{\ell-1}} P(x_1,\ldots,x_n) \equiv \sum_{(x_{n-\ell+2},\ldots,x_n)\in\{0,1\}^{\ell-1}} V(P(x_1,\ldots,x_n)) \pmod{2^\ell}.$$

   Note that in above, the two sums on both sides of the equality are taken over *integers* and not $\mathbb{F}_2$.

[18] That is, the algorithm is given a description of the polynomial part P of C by listing all coefficients in P, and a description of the free part Q of C by giving the truth-table of Q. Such description takes roughly $\sum_{i=0}^{d} \binom{n}{i} + 2^{n-p}$ bits.

3. We construct an $(n-\ell+1)$-variable polynomial P' over $\mathbb{Z}_{2^\ell}$ as

$$P'(x_1,\ldots,x_{n-\ell+1}) := \sum_{(x_{n-\ell+2},\ldots,x_n)\in\{0,1\}^{\ell-1}} V(P(x_1,\ldots,x_n)).$$

   Since P' is a $\mathbb{Z}_{2^\ell}$-polynomial, the sum on the right side above is taken over $\mathbb{Z}_{2^\ell}$ (not $\mathbb{F}_2$). For the sufficiently large K, the number of non-zero coefficients in the polynomial P' is bounded by $2^{0.01n}$. Fix one such K. We can first compute the description of the polynomial $P(x)^i$ (that is, a list of all its coefficients) for $i \in [2\ell-1]$, and then compute the description of the polynomial $V(P(x))$ as well. Last, we enumerate all possible assignments to last $\ell-1$ input bits and then take a sum to compute the description of P'. This takes $2^{0.02n}\cdot\mathrm{poly}(n) + 2^{0.01n+\ell} \le 2^{0.1n}$ time.

4. By the modulus-amplifying property of V, we know that $P'(x_1,\ldots,x_{n-\ell+1})$ is exactly the number of $(x_{n-\ell+2},\ldots,x_n)$ such that $P(x_1,\ldots,x_n)=1$. Using dynamic programming, we can then produce the table $(P'(x_1,\ldots,x_{n-\ell+1}))_{x_1,\ldots,x_{n-\ell+1}}$ in $O(2^{n-\ell}\cdot\mathrm{poly}(n)) \le O(2^{n-n/Kd})$ time. We take a summation over the table and report the answer, which completes the description of algorithm.

Our adaptation modifies the Step (4). Recall that we want to evaluate $\sum_x(-1)^{P(x)}\cdot Q(x_{\le n-p})$, where Q is an arbitrary Boolean function depending only on the length-$(n-p)$ prefix. Observe that Q just applies a "global XOR" to all the inputs sharing the same $(n-p)$-length prefix, which can fit perfectly in Step (3) of the algorithm above.

More precisely, note that in Step (4), entries of the table $(P'(x_1,\ldots,x_{n-\ell+1}))_{x_1,\ldots,x_{n-\ell+1}}$ correspond to disjoint sets of inputs. For ease of presentation, we say that an entry "contains" its corresponding inputs. A crucial observation is, as long as $\ell-1 \le p$ [19], the effect of Q is the same on every inputs in one entry, since all inputs in a single entry share the same $(n-\ell+1)$-prefix. Therefore, we can easily handle the effect of Q to the summation after producing the table in Step (4) for the polynomial P: If $Q(x_{\le n-p})=1$, then the contribution from the entry does not change. Otherwise, all outputs in this entry should change the sign. In conclusion, the evaluation of the summation can be done in $2^{n-\Omega(\min(n/d,p))+O(\log n)}$ time. ∎

Proof of Lemma 10.6.3. For a given input $r\in\{0,1\}^m$ to the generator $\mathcal{G}$, we write $r = y\circ s_1\circ\cdots\circ s_k$,

[19] We can assume that this condition holds. Otherwise, we apply the algorithm for a larger degree parameter $d' = \Theta(n/p)$

where $|y| = n - p$ and $|s_i| = p$ for each $i \in [k]$ (recall that $m = n + (k-1)p$). We simply define

$$\mathcal{G}(r) = (y \circ s_1, \ldots, y \circ s_k).$$

That is, all the generated instances share the same $(n-p)$-length prefix, and each of them holds an independent p-length suffix.

We now verify that this is an $\mathcal{F}^{n,d,p}$-restrictable generator. According to Definition 10.2.6, we need to define the functions T_i and verify the three items of Definition 10.2.6. For every $(m-n)$-bit string α, we write $\alpha = \alpha_1 \circ \cdots \circ \alpha_{k-1}$ where each of α_i has length p. For every $i \in [k]$, set

$$T_i(x, \alpha) = x_{\leq n-p} \circ \alpha_1 \circ \cdots \circ \alpha_{i-1} \circ x_{>n-p} \circ \alpha_i \circ \cdots \circ \alpha_k.$$

Here recall that $x_{\leq n-p}$ and $x_{>n-p}$ denote the length-$(n-p)$ prefix and length-p suffix of x, respectively.

It is straightforward to verify that Item (1) and (2) of Definition 10.2.6 hold. We establish Item (3) below. For every list of functions $(u_j \colon \{0,1\}^n \to \{-1,1\})_{j \in [k] \setminus \{i\}}$, $\alpha \in \{0,1\}^{m-n}$ and $C \in \mathcal{H}_m^{d,(m-n)+p}$, we consider the function

$$D(x) \coloneqq C(T_i(x, \alpha)) \cdot \prod_{j \in [k] \setminus \{i\}} u_j(\mathcal{G}(T_i(x, \alpha))_j).$$

Clearly $C(T_i(x, \alpha))$ is an $\mathcal{H}_n^{d,p}$-function in x: the polynomial part of C does not increase its degree in a restriction, and the free part of C only depends on the length-$(n-p)$ prefix of its input, which is also the length-$(n-p)$ prefix of x in computing $C(T_i(x, \alpha))$. Moreover, for every $j \in [k] \setminus \{i\}$, the function $u_j(\mathcal{G}(T_i(x, \alpha))_j)$ only depends on the $(n-p)$-length prefix of x, which is also an $\mathcal{H}_n^{d,p}$-function. Since $\mathcal{H}_n^{d,p}$ is closed under XOR (that is, multiplication over $\{-1,1\}$), the function $D(x)$ is in $\mathcal{H}_n^{d,p}$. This completes the proof. ■

10.6.2 Applying the New XOR Lemma

Let $\beta > 0$ be a sufficiently enough constant. We will consider the function collection

$$\mathcal{F}^{\mathsf{poly}} = \bigcup_{n \geq 1} \mathcal{H}_n^{\beta\sqrt{n/\log n}, \sqrt{n \log n}/\beta}.$$

Also let $S(n) := 2^{\beta\sqrt{n \log n}}$. By Lemma 10.6.2, there exists $\beta > 0$ such that $\mathcal{F}^{\mathsf{poly}}$ is $S(n)$-applicable: $\mathcal{F}^{\mathsf{poly}}$ is closed under negation; there is a $2^{n-\Omega(\sqrt{n \log n}/\beta)} \leq 2^n/S(n)$-time algorithm solving #SAT for $\mathsf{AND}_4 \circ \mathcal{F}^{\mathsf{poly}}$ (which can reduce to solving 16 #SAT tasks for $\mathsf{XOR}_4 \circ \mathcal{F}^{\mathsf{poly}}$ by standard Fourier analysis); $\mathcal{F}^{\mathsf{poly}}$ contains all parity functions, which are just polynomials of degree 1. Applying Theorem 10.5.2 to $\mathcal{F}^{\mathsf{poly}}$, we obtain the following.

Corollary 10.6.4. *There are constants $\alpha, \delta > 0$ and $K \geq 1$ such that the following hold. There is an E^{NP} machine which, for all sufficiently large n, on input 1^n, outputs in $2^{O(n)}$ time a Boolean function $f\colon \{0,1\}^\ell \to \{-1,1\}$ where $\ell \in [n, Kn]$ such that one of the following holds.*

1. *f cannot be computed by $2^{\alpha\sqrt{\ell \log \ell}}$-size general circuits.*
2. *For every $\mathsf{Sum} \circ \mathcal{F}^{\mathsf{poly}}_\ell$-function $H\colon \{0,1\}^\ell \to \mathbb{R}$ such that $\mathsf{complexity}(H) \leq 2^{3\alpha\sqrt{\ell \log \ell}}$ and $\|H\|_4 \leq 1$ it holds that*

$$\langle f, H \rangle < (1 - \delta).$$

Using the algorithm of Corollary 10.6.4 we can construct in $2^{O(n)}$ time a function f that meets one of two conditions above. Now we apply hardness amplification on the function f, depending on which case in Corollary 10.6.4 holds.

Case 1: f is worst-case hard against general circuits. In this case, we apply Theorem 3.3.1 to the function f_ℓ. In $2^{O(n)}$ time we can get an $s = \Theta(\ell)$-bit function g'_s such that g'_s cannot be $\left(\frac{1}{2} + 2^{-o(\sqrt{n/\log n})}\right)$-approximated by (general) circuits of size $2^{o(\sqrt{n \log n})}$. Since every n-variable $\mathbb{F}_2$-polynomial of degree at most b can be simulated by a circuit of size $2^{O(b \log n)}$. It follows that g'_s cannot be $\left(\frac{1}{2} + 2^{-o(\sqrt{n/\log n})}\right)$-approximated by $\mathbb{F}_2$-polynomials of degree at most $o(\sqrt{n/\log n})$.

Case 2: f is weakly average-case hard against $\mathsf{Sum} \circ \mathcal{F}^{\mathsf{poly}}_\ell$. In this case, note that

$$\mathcal{H}^{\beta\sqrt{\ell/\log \ell}, \sqrt{\ell \log \ell}/\beta}_\ell \subseteq \mathcal{F}^{\mathsf{poly}} \cap \mathcal{F}^{\ell, \beta\sqrt{\ell/\log \ell}, \sqrt{\ell \log \ell}/\beta}.$$

We apply Lemma 10.4.1 to the function f_ℓ with $\mathcal{F}^{\ell, \beta\sqrt{\ell/\log \ell}, \sqrt{\ell \log \ell}/\beta}$-restrictable generator of Lemma 10.6.3, where we set the inapproximability parameter as $\varepsilon = 2^{-\alpha\sqrt{\ell/\log \ell}}$. Let the number of instances generated in applying Lemma 10.4.1 be $k = \Theta(\log \varepsilon^{-1}) = \Theta(\alpha\sqrt{\ell/\log \ell})$. Also let the instance generator be $\mathcal{G}\colon \{0,1\}^s \to \{0,1\}^{nk}$. Since the seed length to the restrictable generator is $m = \ell + k\sqrt{\ell \log \ell}/\alpha \leq O(\ell)$, it follows from Lemma 10.4.1 that $s \leq O(m) = O(\ell)$. Now, we define a function $g'_s\colon \{0,1\}^s \to \{-1,1\}$ as $g'_s := f^{\oplus k} \circ \mathcal{G}$. By Lemma 10.4.1, it follows that g'_s can-

not be $\left(\frac{1}{2}+2^{-o(k)}\right)$-approximated by $\mathcal{F}^{\mathsf{poly}}$-functions. We have in particular that g'_s cannot be $\left(\frac{1}{2}+2^{-o(\sqrt{n/\log n})}\right)$-approximated by $\mathbb{F}_2$-polynomials of degree at most $\beta\sqrt{n/\log n}$.

Padding. In both cases, given n, we can find in $2^{O(n)}$ time a Boolean function $g'_s\colon \{0,1\}^s \to \{-1,1\}$ with $s=\Theta(n)$, such that g'_s cannot be $\left(\frac{1}{2}+2^{-o(\sqrt{n/\log n})}\right)$-approximated by $\mathbb{F}_2$-polynomials of degree at most $o(\sqrt{n/\log n})$. Now we choose a large enough constant C so that $s \le Cn$ for all sufficiently large n.

Design of the E^{NP} function. We design the final E^{NP} algorithm $\mathcal{A}$ as follows. Given an input z, $\mathcal{A}$ sets $n=|z|/C$ and finds the function g'_s as described before. Since $s \le nC \le |z|$, we just let $\mathcal{A}$ output $g'_s(z_{\le s})$. If follows that for sufficiently large input length m, $\mathcal{A}$ on m-bit inputs computes a function that cannot be $\left(\frac{1}{2}+2^{-o(\sqrt{m/\log m})}\right)$-approximated by $\mathbb{F}_2$-polynomials of degree at most $o(\sqrt{m/\log m})$. This completes the proof of Theorem 1.7.4.

10.7 Better Degree-error Trade-off against $\mathbb{F}_2$-Polynomials and P^{NP} Construction of Extremely Rigid Matrices

In this section, we prove a degree-error trade-off for E^{NP} against $\mathbb{F}_2$-polynomials (Theorem 10.0.1) and present P^{NP} construction of extremely rigid matrices (Theorem 1.7.3).

Before we proceed, we remark that one can already combine known techniques from [CLW20] and [BHPT20] to prove the following (which is also independently utilized by [HV21]).

Theorem 10.7.1. *For every $\beta \in (0,1)$, there is an E^{NP} function f such that, for every sufficiently large n, it holds that $\mathrm{corr}(f, n^\beta/\log n) \le \exp(-\Omega(n^{\frac{1}{2}(1-\beta)}))$.*

Using our new derandomized XOR lemma, we can substantially improve the correlation parameters from Theorem 10.7.1 to those stated in Theorem 10.0.1 and Theorem 1.7.3 (restated below).

Reminder of Theorem 10.0.1. *For every $\beta \in (0,1)$, there is an E^{NP} function f such that, for every sufficiently large n, it holds that $\mathrm{corr}(f, n^\beta/\log n) \le \exp(-\Omega(n^{\frac{2}{3}(1-\beta)}))$.*

Reminder of Theorem 1.7.3. *For every constant $\varepsilon \in (0,1)$, there is a P^{NP} algorithm which on input 1^n outputs an $n \times n$ $\mathbb{F}_2$-matrix H_n satisfying $\mathcal{R}_{H_n}(2^{\log^{1-\varepsilon} n}) \ge (1/2 - \exp(-\log^{2/3\cdot\varepsilon} n)) \cdot n^2$, for every sufficiently large n.*

In fact, [HV21] stated a more fine-grained trade-off: for every $n, \rho, k \in \mathbb{N}_{\geq 1}$ such that $\log \rho \leq \delta \log n/k(\log\log n + k)$ for a sufficiently small $\delta > 0$, they constructed an $n \times n$ matrix $H_n \in \mathbb{F}_2^{n\times n}$ such that $\mathcal{R}_{H_n}(\rho) \geq (1/2 - 2^{-k}) \cdot n^2$. Through a more careful calculation, our approach (which is based on the derandomized XOR Lemma) can recover their results and indeed applies to a wider regime $\log \rho \leq \delta \log n/\sqrt{k}(\log\log n + k)$. For the sake of a clearer exposition, in this section we will only focus on the case where ρ, the rank parameter, is set to $2^{(\log n)^\beta}$ for a constant $\beta \in (0,1)$.

Now, let us formally define the function collection we will work with.

Definition 10.7.2 (Collection of low-rank functions)**.** *For every $n, r \in \mathbb{N}$ such that $r \leq 2^n$, let $\mathcal{M}^r_{2n}$ denote the collection of functions such that, for every $f \in \mathcal{M}$, there is a $2^n \times 2^n$ matrix M over $\mathbb{F}_2$ of rank at most r such that $f(x,y) = (-1)^{M(x,y)}$ for every $(x,y) \in \{0,1\}^{2n}$. Typically, we describe an $\mathcal{M}^r_{2n}$-function by giving the low-rank decomposition of the matrix $M := A \cdot B^T$ where $A, B \in \mathbb{F}^{2^n\times r}$, which only requires $O(2^n \cdot r)$ bits. We also let $\mathcal{M}^r_{2n+1}$ be an empty collection for every $n \in \mathbb{N}$.*

In this section, we will frequently view a $2n$-bit input function as a $2^n \times 2^n$ matrix and vice versa. For convenience, for every $n \in \mathbb{N}_{\geq 1}$ and every $2n$-bit function $f\colon \{0,1\}^{2n} \to \{-1,1\}$, for every input x to f, we call the first n bits of x (*i.e.*, $x_{\leq n}$) as the *row* index, and the last n bits of x (*i.e.*, $x_{>n}$) as the *column* index.

We will prove the following theorem, which implies Theorem 1.7.3 and Theorem 10.0.1.

Theorem 10.7.3. *For every $\beta \in (0,1)$, there is an E^{NP} function f such that, for every sufficiently large n, f_{2n} is a function that cannot be $\left(\frac{1}{2} + 2^{-o(n^{\frac{2}{3}(1-\beta)})}\right)$-approximated by $\mathcal{M}^{2^{2n^\beta}}_{2n}$-functions.*

We sketch how we obtain Theorem 10.0.1 and Theorem 1.7.3 from Theorem 10.7.3,

Proof of Theorem 10.0.1. We show that $\mathcal{M}^r_{2n}$ contains all degree-d $\mathbb{F}_2$-polynomials on $2n$ variables, given that $\sum_{i=0}^{d} \binom{2n}{i} \leq r$. In fact, for every $\mathbb{F}_2$-polynomial $P(x_1, \ldots, x_n, y_1, \ldots, y_n)$ of degree at most d, if we write the truth-table of P as a $2^n \times 2^n$ matrix, then each monomial corresponds to a rank-1 matrix, and the matrix is a sum of at most $\sum_{i=0}^{2n} \binom{2n}{i} \leq r$ rank-1 matrix (*i.e.*, the monimials).

We consider the function f constructed in Theorem 10.7.3, given that f_{2n} cannot be $\left(\frac{1}{2} + 2^{-o(n^{\frac{2}{3}(1-\beta)})}\right)$-approximated by $\mathcal{M}^{2^{2n^\beta}}_{2n}$-functions, it follows naturally that $\mathrm{corr}(f_{2n}, n^\beta/\log n) \leq \exp(-\Omega(n^{\frac{2}{3}(1-\beta)}))$. To handle the odd input lengths, we simply define $f_{2n+1}(x) := f_{2n}(x_{\leq 2n})$ for every $x \in \{0,1\}^{2n+1}$, and this completes the proof. ■

Proof of Theorem 1.7.3. Let $\varepsilon \in (0,1)$ be a constant. First, we set $\beta = 1 - \varepsilon$ and let f be the function constructed in Theorem 10.7.3 with parameter β.

We will use a padding argument. Given a sufficiently large integer $n \geq 1$, we set $\ell = \lceil \log \sqrt{n} \rceil$. Consider the function $f_{2\ell}$ constructed in Theorem 10.7.3. We view $f_{2\ell}$ as a $2^\ell \times 2^\ell$ matrix A, and "pad" it into a larger matrix. We set $k = \left\lfloor \frac{n}{2^\ell} \right\rfloor$ and defining a $(k2^\ell) \times (k2^\ell)$ matrix $M' := 1_k \otimes A$ where 1_k denotes all-ones $k \times k$ matrix and $\otimes$ denotes the Kronecker product of matrices. We then further pad the $(k2^\ell) \times (k2^\ell)$ matrix M' to an $n \times n$ matrix M by filling zeros in the empty entries.

The rigidity of M' follows from the fact that $\mathcal{R}_{1_k \otimes A}(r) = \mathcal{R}_A(r) \cdot k^2$ (see, *e.g.*, [AC19, Lemma 2.7]). The rigidity of M follows from the rigidity of M', and the observation that we only add $O(n\sqrt{n}) \leq n^2 \cdot \exp\left(-\log^{\frac{2}{3}(1-\beta)}(n)\right)$ zeros in M. ■

10.7.1 Technical Ingredients

We collect some crucial technical ingredients required by the proof of this section. First, we need a fast #SAT-algorithm for $\mathcal{M}_n^r$-functions.

Lemma 10.7.4 ([AC19, CW16])**.** *For every even integer $n \geq 1$ and $r \leq 2^{o(n)}$, there is a $2^{n-\Omega(n/\log r)}$-time deterministic algorithm for solving* #SAT *problem for $\mathcal{M}_n^r$-functions.*

Second, we introduce a restrictable generator for $\mathcal{M}$-functions with seed length shorter than nk.

Lemma 10.7.5. *For every $n, r, k \in \mathbb{N}_{\geq 1}$ such that $r \leq 2^n$ and $2 \leq k \leq r$, we define*

$$\mathcal{N}^{2n,r,k} := \mathcal{M}_{2n}^r \cup \mathcal{M}_{2n\lceil\sqrt{k}\rceil}^{r-k}.$$

There is an $\mathcal{N}^{2n,r,k}$-restrictable generator $\mathcal{G} \colon \{0,1\}^m \to \{0,1\}^{nk}$ with seed length $m = 2n\left\lceil\sqrt{k}\right\rceil$.

Proof. For brevity, let $t = \left\lceil\sqrt{k}\right\rceil$. We choose an arbitrary but fixed injective mapping $\rho \colon [k] \to [t] \times [t]$, denoted by $\rho(i) = (\rho(i)_u, \rho(i)_v)$. For every $z \in \{0,1\}^m$, we write $z = x_1 \circ \cdots \circ x_t \circ y_1 \circ \cdots \circ y_t$ where $|x_i| = |y_j| = n$ for every $i, j \in [t]$. We design our generator as

$$\mathcal{G}(z) := (x_{\rho(1)_u} \circ y_{\rho(1)_v}, \ldots, x_{\rho(k)_u} \circ y_{\rho(k)_v}).$$

For every $\alpha \in \{0,1\}^{m-2n}$, write $\alpha = \alpha_1 \circ \cdots \circ \alpha_{2t-2}$ where $|\alpha_j| = n$ for each $j \in [2t-2]$. For every $i \in [k]$, $x \in \{0,1\}^{2n}$ and $\alpha \in \{0,1\}^{m-2n}$, we construct the embedding function as

$$T_i(x, \alpha) := (\alpha_1, \ldots, \alpha_{\rho(i)_u - 1}, x_{\leq n}, \ldots, \alpha_{t-1}) \circ (\alpha_t, \ldots, \alpha_{t+\rho(i)_v - 2}, x_{>n}, \ldots, \alpha_{2t-2}).$$

It is easy to verify that Item (1) and (2) of Definition 10.2.6 both hold. We establish Item (3) below. For every list of functions $(u_j\colon \{0,1\}^{2n} \to \{-1,1\})_{j\in[k]\setminus\{i\}}$, $\alpha \in \{0,1\}^{m-2n}$ and $C \in \mathcal{M}_m^{r-k}$, we consider the function

$$D(x) \coloneqq C(T_i(x,\alpha)) \cdot \prod_{j\in[k]\setminus\{i\}} u_j(\mathcal{G}(T_i(x,\alpha))_j).$$

By the design of T_i, the first and last n bits of x occur in the first and last $m/2$ bits of $T_i(x,\alpha)$ respectively, and consequently we have $C(T_i(x,\alpha))$ is an $\mathcal{M}_{2n}^{r-k}$-function in x. Moreover, for every $j \in [k]\setminus\{i\}$, the function $u_j(\mathcal{G}(T_i(x,\alpha))_j)$ only depends on either $x_{\le n}$ or $x_{>n}$, so it is an $\mathcal{M}_{2n}^{1}$-function. Therefore, the function $D(x)$ is in $\mathcal{M}_{2n}^{r-k+(k-1)} \subset \mathcal{M}_{2n}^{r}$, which completes the proof. ■

We also need the rectangular PCP of [BHPT20], stated below.

Theorem 10.7.6 ([BHPT20, Theorem 8.2 and Remark 8.3])**.** *Let M be an algorithm running in time $T = T(n) \ge n$ on inputs of the form (z,y) where $|z| = n$. For any odd constant integer $m \in \mathbb{N}$ such that $T(n)^{1/m} \ge n$, given $z \in \{0,1\}^n$ one can output in time $\mathrm{poly}(n, T^{1/m})$ a PCP verifier $\mathsf{VrecPCP}_z$ with proof length 2^ℓ, randomness γ, completeness 1 and soundness $s \in (0,1)$ such that the following hold.*

- ***Shortness.*** *$T(n) \le 2^\ell \le 2^\gamma \le T \cdot \mathrm{polylog}(T)$, and ℓ is even.*
- ***Query complexity.*** *There is a constant q such that $\mathsf{VrecPCP}_z$ makes only q queries to the proof.*
- ***Rectangular.*** *The randomness $r \in \{0,1\}^\gamma$ can be split into three parts $r = (r_{\mathsf{row}}, r_{\mathsf{col}}, r_{\mathsf{shared}})$ such that $|r_{\mathsf{row}}| = |r_{\mathsf{col}}| \ge \frac{m-6}{2m}\log T(n)$. For a given proof $\pi\colon \{0,1\}^\ell \to \{-1,1\}$ and fixed r_{shared}, we write $\mathsf{VrecPCP}_z^\pi(r_{\mathsf{row}}, r_{\mathsf{col}}, r_{\mathsf{shared}})$ to denote the output of $\mathsf{VrecPCP}_z$ given $(r_{\mathsf{row}}, r_{\mathsf{col}}, r_{\mathsf{shared}})$ as randomness.*

  For every $i \in [q]$, the row index of the i-th query of $\mathsf{VrecPCP}_z$ only depends on the pair $(r_{\mathsf{shared}}, r_{\mathsf{row}})$, and the column index of the i-th query of $\mathsf{VrecPCP}_z$ only depends on $(r_{\mathsf{shared}}, r_{\mathsf{col}})$. After reading the proof, the decision of $\mathsf{VrecPCP}_z$ only depends on r_{shared}, the q queried bits, and p parity check bits over $(r_{\mathsf{row}}, r_{\mathsf{col}})$, where the specification of each parity check is determined only by r_{shared} and z. Here p is a constant.
- ***Completeness.*** *If there is a y such that $M(z,y)$ accepts, then there is a proof H such that*

$$\Pr_{r\leftarrow U_\gamma}[\mathsf{VrecPCP}_z^H(r) = 1] = 1.$$

- ***Soundness.*** *If no y causes $M(z,y)$ to accept, then for every proof H, we have*

$$\Pr_{r\leftarrow U_\gamma}[\mathsf{VrecPCP}_z^H(r) = 1] \le s.$$

- ***Smoothness.*** *For every $p \in \{0,1\}^\ell$, the quantity*

$$\left|\{(r,i) : \mathsf{VrecPCP}_z^H(r) \text{ makes the } i\text{-th query to } H(p)\}\right|$$

is the same.

Arithmetization. By the rectangular property, fixing r_{shared}, the output of $\mathsf{VrecPCP}_z$ depends only on the q queried bits and p parity check bits over $(r_{\mathsf{row}}, r_{\mathsf{col}})$. For each r_{shared}, and $j \in [p]$, we define a function $C_j^{r_{\mathsf{shared}}} : \{0,1\}^{|r_{\mathsf{row}}|+|r_{\mathsf{col}}|} \to \{-1,1\}$ which maps the random bits $(r_{\mathsf{row}}, r_{\mathsf{col}})$ to its parity check result. Note that $C_j^{r_{\mathsf{shared}}}$ can be written as $C_j^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}}) = (-1)^{\mathrm{Par}_j^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}})}$ where $\mathrm{Par}_j^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}})$ computes an affine function over $(r_{\mathsf{row}}, r_{\mathsf{col}})$. Hence $\mathrm{Par}_j^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}})$ can be written as a rank-2 matrix[20], and $C_j^{r_{\mathsf{shared}}} \in \mathcal{M}^2_{|r_{\mathsf{row}}, r_{\mathsf{col}}|}$.

Now, given a randomness $r = (r_{\mathsf{shared}}, r_{\mathsf{row}}, r_{\mathsf{col}})$, we can write the output of $\mathsf{VrecPCP}_z$ as a multi-linear polynomial (over $\mathbb{R}$) of the query answers $(v_1, \ldots, v_q)$ and the parity check $(c_1, \ldots, c_p)$, denoted by $Q_{r_{\mathsf{shared}}} : \mathbb{R}^{q+p} \to \mathbb{R}$, which maps $(v_1, \ldots, v_q, c_1, \ldots, c_p)$ to a bit from $\{0,1\}$. Moreover, for every $r = (r_{\mathsf{shared}}, r_{\mathsf{row}}, r_{\mathsf{col}})$, we can write the decision as a multi-linear polynomial of queried bits, denoted by $P_r : \mathbb{R}^q \to \mathbb{R}$ (see Section 10.3 for details). Since both of P_r and $Q_{r_{\mathsf{shared}}}$ are multi-linear polynomials, we have

$$P_r(v_1, \ldots, v_q) = Q_{r_{\mathsf{shared}}}(v_1, \ldots, v_q, C_1^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}}), \ldots, C_p^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}})). \tag{10.35}$$

Then we can define an "arithmetized verifier", denoted by $\widetilde{\mathsf{VrecPCP}}_z$, which takes a real-valued function $H : \{0,1\}^\ell \to \mathbb{R}$ as proof. $\widetilde{\mathsf{VrecPCP}}_z$ samples a random string $(r_{\mathsf{row}}, r_{\mathsf{col}}, r_{\mathsf{shared}}) \leftarrow U_\gamma$, reads some values $(v_1, \ldots, v_q)$ from H according to the randomness, and outputs $P_r(v_1, \ldots, v_q)$. Intuitively, for a given real-valued function $H : \{0,1\}^\ell \to \mathbb{R}$ as a proof to $\widetilde{\mathsf{VrecPCP}}_z$, as long as H is close to some Boolean function $H' : \{0,1\}^\ell \to \{-1,1\}$, we will have that $\mathbb{E}_r\left[\widetilde{\mathsf{VrecPCP}}_z^H(r)\right] \approx \mathbb{E}_r\left[\mathsf{VrecPCP}_z^{H'}(r)\right]$.

In fact, by the smoothness of $\mathsf{VrecPCP}_z$, we have the following lemma.

Lemma 10.7.7. *For two functions $H : \{0,1\}^\ell \to \mathbb{R}$ and $H' : \{0,1\}^\ell \to \{-1,1\}$, it holds that*

$$\mathop{\mathbb{E}}_{r \leftarrow U_\gamma}\left|\widetilde{\mathsf{VrecPCP}}_z^H(r) - \mathsf{VrecPCP}_z^{H'}(r)\right| \le 2^q \cdot q \cdot \left(2 + q^{1/(2q-2)}\|H\|_{2q-2}\right)^{2q-2} \|H - H'\|_2.$$

[20] Note that every affine function over only r_{row} or r_{col} can be written as a rank-1 matrix. Therefore, being an XOR over two such affine functions, $\mathrm{Par}_j^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}})$ can be written as a rank-2 matrix.

We prove Lemma 10.7.7 in Section 10.9.

10.7.2 Proof of Theorem 10.7.3 via the Algorithmic Method

Now we try to prove Theorem 10.7.3 by the algorithmic method. First, we choose a constant K and set $T(n) = n^K$. We also choose a proper parameter m for Theorem 10.7.6. We choose K and m properly so that the construction of $\mathsf{VrecPCP}$ of Theorem 10.7.6 can be done in $T(n)^{1/2}$ time.

We prove the trade-off for every $\beta \in (0,1)$. We first set $c = \frac{1-\beta}{1+\beta/2}$. We let $\alpha, \delta > 0$ be two small enough constants and set $\tau \geq 1$ to be a *large enough* constant. For the algorithm $\mathcal{A}^T_{\mathsf{FS}}$ defined in Theorem 10.5.4, we design a cheating algorithm $\mathcal{A}_{\mathsf{matrix}}$ to speed up the computation of $\mathcal{A}^T_{\mathsf{FS}}$ as follows:

- Given an input $z \in \{0,1\}^n$, $\mathcal{A}_{\mathsf{matrix}}$ applies the rectangular PCP of Theorem 10.7.6 (with the constant m set to 13) to $\mathcal{A}^T_{\mathsf{FS}}(z)$ and obtains $\mathsf{VrecPCP}_z$. Let the proof length to $\mathsf{VrecPCP}_z$ be 2^ℓ. Note that $2^\ell = T(n) \cdot \mathrm{polylog}(T(n))$.
- Then $\mathcal{A}_{\mathsf{matrix}}$ guesses a $\mathsf{Sum} \circ \mathcal{M}_\ell^{2^{\tau\ell^{1-c}}}$-function of complexity at most $2^{\alpha\ell^c}$, denoted by H, as the proof. It then applies the following tests:

$$\mathop{\mathbb{E}}_{(x,y)\leftarrow U_\ell} (1 - H(x,y))^2 (1 + H(x,y))^2 \leq \delta, \tag{10.36}$$

$$\mathop{\mathbb{E}}_{(x,y)\leftarrow U_\ell} H(x,y)^{2q-2} \leq 1, \tag{10.37}$$

$$\mathop{\mathbb{E}}_{r\leftarrow U_\gamma} \left[\widetilde{\mathsf{VrecPCP}}^H_z(r)\right] \geq \frac{1+s}{2}. \tag{10.38}$$

  It accepts the proof H if all the above three tests pass, and reject otherwise. We will show in Lemma 10.7.8 (its proof is deferred to Section 10.9) that for fixed $\tau \geq 1$, each of these tests can be done in $o(n^K)$-time for sufficiently small α. This completes the design of $\mathcal{A}_{\mathsf{matrix}}$.

Lemma 10.7.8. *For every constant $\tau \geq 1$, there is a sufficiently small $\alpha > 0$ such that the following is true. For every $z \in \{0,1\}^*$ and H being a $\mathsf{Sum} \circ \mathcal{M}_\ell^{2^{\tau\ell^{1-c}}}$-function with $\mathsf{complexity}(H) \leq 2^{\alpha\ell^c}$, evaluations of the left-hand sides of (10.36)-(10.38) can be done in $2^{\ell - \Omega(\ell^c/\alpha)} \leq o(n^K)$ time.*

Useful Lemmas. The tests (10.36)-(10.38) play a role that is similar to that of the tests (10.30)-(10.32) in the proof of Theorem 10.5.2. Analogously, the following lemmas can be established, and we defer their proofs to Section 10.9.

First, we can verify that the algorithm $\mathcal{A}_{\mathsf{matrix}}$ only makes one-sided error.

Lemma 10.7.9. *For every sufficiently small $\delta > 0$, it holds that $\mathcal{A}_{\mathsf{matrix}}(z) \leq \mathcal{A}^T_{\mathsf{FS}}(z)$ for every $z \in \{0,1\}^*$.*

Second, for every $z \in \{0,1\}^*$ such that $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$ and $\mathcal{A}_{\mathsf{matrix}}(z) = 0$, the correct proof for $\mathsf{VrecPCP}_z$ is rigid in the following sense. (We say a proof H is correct for $\mathsf{VrecPCP}_z$ if it makes $\mathsf{VrecPCP}_z$ always accepts.)

Lemma 10.7.10. *For every sufficiently small $\alpha > 0$, the following is true. For every z such that $\mathcal{A}_{\mathsf{matrix}}(z) = 0$ and $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$, every correct proof for $\mathsf{VrecPCP}_z$ is a function $H' \colon \{0,1\}^\ell \to \{-1,1\}$ such that, for every $\mathsf{Sum} \circ \mathcal{M}^{2^{\tau\ell^{1-c}}}_\ell$-function H with $\mathsf{complexity}(H) \leq 2^{\alpha\ell^c}$, it holds that $\langle H, H' \rangle \leq (1 - \delta/5)\|H\|_{2(q-1)}$.*

Applying the derandomized XOR lemma. In the following we will combine the derandomized XOR lemma Lemma 10.4.1 and Lemma 10.7.10 to finish the proof of Theorem 10.7.3. First, we show that there is a constant $C \geq 1$ and an E^{NP} algorithm $\mathcal{A}_{\mathsf{xor}}$ such that, for every sufficiently large n, $\mathcal{A}_{\mathsf{xor}}$ constructs a function $g \colon \{0,1\}^s \to \{-1,1\}$ that cannot be $\left(\frac{1}{2} + 2^{-o(s^{\frac{2}{3}(1-\beta)})}\right)$-approximated by $\mathcal{M}^{2^{o(s^\beta)}}_s$-functions, where $s \in [n, Cn]$ is an even integer.

Given $n \in \mathbb{N}_{\geq 1}$, $\mathcal{A}_{\mathsf{xor}}$ sets $n' = n^{2/(2+c)}$ and $m = 2^{n'/K}$ (for simplicity, we ignore the rounding issue here and pretend both n' and m are integers). By Theorem 10.5.4, $\mathcal{A}_{\mathsf{xor}}$ can find in $\mathrm{poly}(m) \leq 2^{O(n)}$ time an input z of length $|z| \in [m, m + m^K]$ such that $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$ and $\mathcal{A}_{\mathsf{matrix}}(z) = 0$. We consider the rectangular PCP system $\mathsf{VrecPCP}_z$. Let ${N_1}^2$ be the proof length of $\mathsf{VrecPCP}_z$. It holds that $N_1 \leq O(|z|^{K/2}) \leq 2^{O(n')}$. Also let $\ell = 2 \log N_1 \leq O(n')$.

Since $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$, there exists a function $H' \colon \{0,1\}^\ell \to \{-1,1\}$ being a correct proof for $\mathsf{VrecPCP}_z$. *i.e.*, $\mathsf{VrecPCP}^{H'}_z$ is a tautology. Using an NP oracle, $\mathcal{A}_{\mathsf{xor}}$ can find the lexicographically first such H' in $\mathrm{poly}(m) \leq 2^{O(n)}$ time. By Lemma 10.7.10, it follows that for every $\mathsf{Sum} \circ \mathcal{M}^{2^{\tau\ell^{1-c}}}_{2\ell}$-function C with complexity bounded by $2^{\alpha\ell^c}$, we have $\langle H', C \rangle \leq (1 - \delta/5)\|C\|_{2q-2}$.

We will apply the derandomized XOR Lemma (Lemma 10.4.1) with the restrictable generator given by Lemma 10.7.5. We set the inapproximability parameter as $\varepsilon = 2^{-\frac{\alpha}{3}\ell^c}$. Then we let $k = \Theta(\log \varepsilon^{-1}) = \Theta(\ell^c)$ be the number of instances given by Lemma 10.4.1. Consider the function collection $\mathcal{N}^{2\ell, 2^{\tau\ell^{1-c}}, k}$. We let $\mathcal{A}_{\mathsf{xor}}$ apply Lemma 10.4.1 to the function H' with the $\mathcal{N}^{2\ell, 2^{\tau\ell^{1-c}}, k}$-restrictable generator of Lemma 10.7.5. $\mathcal{A}_{\mathsf{xor}}$ obtains from H' a function $g \colon \{0,1\}^s \to \{-1,1\}$ where $s = \Theta(n\sqrt{k}) = \Theta(\ell^{1+c/2})$, such that g cannot be $\left(\frac{1}{2} + 2^{-o(\ell^c)}\right)$-approximated by $\mathcal{M}^{2^{\tau'\ell^{1-c}}}_s$-functions for some $\tau' \geq \Omega(\tau)$.

Note that $\ell = \Theta(n')$, $s = \Theta(\ell^{1+c/2}) = \Theta(n)$ and $\beta = \frac{2-2c}{2+c}$. So we have that $\ell^c = \Theta(s^{\frac{2}{3}(1-\beta)})$

and $\ell^{1-c} = \Theta(s^\beta)$. Hence, it follows that g cannot be $\left(\frac{1}{2} + 2^{-o(s^{\frac{2}{3}(1-\beta)})}\right)$-approximated by $\mathcal{M}_s^{2^{\tau'' s^\beta}}$-functions for some $\tau'' \geq \Omega(\tau')$. Since $s \leq O(n)$, there is a constant $C \geq 1$ such that $s \leq Cn$ holds for every sufficiently large n.

Padding. Now we design the final E^{NP} algorithm $\mathcal{A}$. On an input x, $\mathcal{A}$ sets $n = |x|/C$ and invokes $\mathcal{A}_{\mathsf{xor}}$ to find a function $g\colon \{0,1\}^s \to \{-1,1\}$ as shown before. Since $s \leq Cn \leq |x|$, $\mathcal{A}$ just outputs $g(x_{\leq s/2}, x_{>n-s/2})$. For every sufficiently large n, $\mathcal{A}$ on $2n$-bit inputs computes a function that cannot be $\left(\frac{1}{2} + 2^{-o(n^{\frac{2}{3}(1-\beta)})}\right)$-approximated by $\mathcal{M}_{2n}^{2^{\tau^{(3)} n^\beta}}$-functions for $\tau^{(3)} \geq \Omega(\tau'') \geq \Omega(\tau)$.

Choice of parameters. Finally, we specify our choice of the parameters, and which completes the proof. We first choose τ such that $\tau^{(3)} \geq 1$. Then we choose α accordingly such that $\mathcal{A}_{\mathsf{matrix}}$ runs in time $o(T(n))$. We also choose δ such that Lemma 10.7.9 and Lemma 10.7.10 hold.

10.8 Missing Proofs in Section 10.5

10.8.1 A Useful Lemma

Before we proceed, we state and prove the following useful lemma. It will be used frequently in this section and Section 10.9.

Lemma 10.8.1. *Let $\mathcal{S}$ be a set. Let $P\colon \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$ be such that, for every $x \in \mathcal{S}$, $P(x, \star)$ is a multi-linear polynomial with absolute values of coefficients bounded by M. Let $f_1, \ldots, f_d\colon \mathcal{S} \to \mathbb{R}$ and $g_1, \ldots, g_d\colon \mathcal{S} \to \mathbb{R}$ be two list of functions. Then, it holds that*

$$\begin{aligned}&\mathop{\mathbb{E}}_{x \leftarrow \mathcal{S}} |P(x, f_1(x), \ldots, f_d(x)) - P(x, g_1(x), \ldots, g_d(x))| \\ &\leq M \cdot \prod_{j=1}^{d} \left(\|g_j\|_{2(d-1)} + \|f_j\|_{2(d-1)} + 1\right) \cdot \sum_{i=1}^{d} \|f_i - g_i\|_2 .\end{aligned}$$

To prove Lemma 10.8.1, the following inequality will be used. It can be proved by iteratively applying Hölder's inequality.

Claim 10.8.2. *Let $h_1, \ldots, h_k\colon \mathcal{S} \to \mathbb{R}$ be k functions, it holds that $\left\|\prod_{i=1}^k h_i\right\|_2 \leq \prod_{i=1}^k \|h_i\|_{2k}$.*

Proof. Use induction on k. For $k = 1$, the statement is trivial. Assuming it is true for $k-1$, we have

$$\begin{aligned}
\left\| \prod_{i=1}^{k} h_i \right\|_2 &\le \left\| \prod_{i=1}^{k-1} h_i \right\|_{2k/(k-1)} \cdot \|h_k\|_{2k} && \text{(Hölder's inequality with } (p,q) = (2k/(k-1), 2k)) \\
&\le \prod_{i=1}^{k} \|h_i\|_{2k}. && \text{(induction hypothesis)}
\end{aligned}$$

■

Then we prove Lemma 10.8.1.

Proof of Lemma 10.8.1. We write

$$P(x, v_1, \ldots, v_d) = \sum_{T \subseteq [d]} \alpha_{x,T} \prod_{i \in T} v_i,$$

where $|\alpha_{x,T}| \le M$ for each $T \subseteq [d]$. Fixing an $T \subseteq [d]$, we consider

$$\mathop{\mathbb{E}}_{x \leftarrow \mathcal{S}} |\alpha_{x,T}| \cdot \left| \prod_{i \in T} f_i(x) - \prod_{i \in T} g_i(x) \right|.$$

It follows that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x \leftarrow \mathcal{S}} |\alpha_{x,T}| \left| \prod_{i \in T} f_i(x) - \prod_{i \in T} g_i(x) \right| \\
&\le M \cdot \mathop{\mathbb{E}}_{x \leftarrow \mathcal{S}} \left| \sum_{i \in T} \left[(f_i(x) - g_i(x)) \left(\prod_{j : j \in T, j < i} f_j(x) \prod_{j : j \in T, j > i} g_j(x) \right) \right] \right| \\
&\le M \cdot \sum_{i \in T} \left\langle f_i - g_i, \prod_{j : j \in T, j < i} f_j \prod_{j : j \in T, j > i} g_j \right\rangle \\
&\le M \cdot \sum_{i \in T} \|f_i - g_i\|_2 \cdot \left\| \prod_{j : j \in T, j < i} f_j \prod_{j : j \in T, j > i} g_j \right\|_2 && \text{(By Cauchy-Schwartz)} \\
&\le M \cdot \sum_{i \in T} \|f_i - g_i\|_2 \left[\prod_{j \in T, j < i} \|f_j\|_{2(|T|-1)} \prod_{j \in T, j > i} \|g_j\|_{2(|T|-1)} \right] && \text{(By Claim 10.8.2)} \\
&\le M \cdot \sum_{i \in T} \|f_i - g_i\|_2 \left[\prod_{j \in T, j < i} \|f_j\|_{2(d-1)} \prod_{j \in T, j > i} \|g_j\|_{2(d-1)} \right]. && \text{(By } \|f\|_{2(|T|-1)} \le \|f\|_{2(d-1)})
\end{aligned}$$

Finally, taking a summation over $T \subseteq [d]$ proves the lemma.

■

10.8.2 $\mathcal{A}_{\mathsf{cheat}}$ Makes Only One-sided Error

In this subsection, we verify Lemma 10.5.9.

Reminder of Lemma 10.5.9. *For every small enough constants $\alpha, \delta \in (0,1)$ and for every sufficiently large constant $K \geq 1$, the following holds: $\mathcal{A}_{\mathsf{cheat}}(z) \leq \mathcal{A}_{\mathsf{FS}}^{T}(z)$ for all but finitely many inputs z.*

Proof of Lemma 10.5.9. Suppose that $\mathcal{A}_{\mathsf{FS}}^{T}(z) = 0$. We consider the execution of $\mathcal{A}_{\mathsf{cheat}}(z)$. Let $\ell = \log T(n) + O(\log\log T(n))$. By Item 2 of Claim 10.5.7, for every circuit $C\colon \{0,1\}^{\ell} \to \{-1,1\}$ fed to VPCP_z, the circuit $\mathsf{VPCP}_z^C\colon \{0,1\}^{\ell} \to \{0,1\}$ evaluates to 1 on at most $2^{\ell}/\mathrm{poly}(n)$ many inputs. Recall in the verification of $\mathcal{A}_{\mathsf{cheat}}$, it applies the following tests(rewriting (10.30) - (10.32)):

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} P_{i,j}(x) \leq \delta, \tag{10.39}$$

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} H(i,j,x)^2 \leq 1, \tag{10.40}$$

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} F_i(x) \geq c_{\mathsf{pcpp}} - \frac{1}{2}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}), \tag{10.41}$$

where P_{ij} is defined as

$$P_{ij}(x) = \begin{cases} (1+T_{ij}(x))^2(1-T_{ij}(x))^2, & \text{if } T_{ij} \in Z, \\ (\mathsf{Enc}_s(x) - T_{ij}(x))^2, & \text{if } T_{ij} \in Y. \end{cases}$$

Now suppose that there is a $\mathsf{Sum} \circ \mathcal{F}$ function $H(i,j,x)$ which can pass the tests above. We will derive a contradiction.

Let (Y,Z) be the real-valued proof constructed by $\mathcal{A}_{\mathsf{cheat}}$. We define from (Y,Z) a list of Boolean valued proof circuits $(\widehat{Y},\widehat{Z})$ as follows. First, let $\widehat{Y}_i := \mathsf{Enc}_i(x)$. For each Z_i, let $\widehat{Z}_i(x) := \mathsf{sign}(Z_i(x))$. We can then analogously define $\widehat{T}_{ij} \in (\widehat{Y},\widehat{Z})$ and $\widehat{F}_i(x) := \widetilde{\mathsf{Cons}_i}(\widehat{T}_{i,1},\widehat{T}_{i,2})$. Note that since $\mathcal{A}_{\mathsf{FS}}^{T}(z) = 0$, we have

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} \widehat{F}_i(x) \leq (1 - 1/\mathrm{poly}(n)) \cdot s_{\mathsf{pcpp}} + c_{\mathsf{pcpp}}/\mathrm{poly}(n) \leq c_{\mathsf{pcpp}} - \frac{9}{10} \cdot (c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}) \tag{10.42}$$

by properties of PCP and PCPP. Also note from (10.41) that $\mathbb{E}_{i\in[m]}\mathbb{E}_{x\leftarrow U_\ell} F_i(x)$ is bounded below by $\frac{c_{\mathsf{pcpp}}+s_{\mathsf{pcpp}}}{2}$.

In the following, we further bound $\mathbb{E}_{i\in[m]}\mathbb{E}_{x\leftarrow U_\ell}[\widehat{F}_i(x) - F_i(x)]$, which will lead to a contradiction.

We will apply Lemma 10.8.1. We let $\mathcal{S}$ be $[m] \times \{0,1\}^\ell$, and define a function $P_F \colon [m] \times \{0,1\}^\ell \times \mathbb{R}^2 \to \mathbb{R}$ as $P_F(i, x, v_1, v_2) := \widetilde{\mathsf{Cons}_i}(v_1, v_2)$. For $j \in \{1,2\}$, we set function $f_j \colon [m] \times \{0,1\}^\ell \to \mathbb{R}$ as $f_j(i,x) := T_{ij}(x)$, and function g_j as $g_j(i,x) := \widehat{T}_{ij}(x)$.

Applying Lemma 10.8.1 with the function P_F, the set $\mathcal{S}$ and two list of functions (f_1, f_2), (g_1, g_2), it follows that[21]

$$\begin{aligned} &\mathop{\mathbb{E}}_{x \leftarrow \mathcal{S}} |P_F(x, f_1(x), f_2(x)) - P_F(x, g_1(x), g_2(x))| \\ &\leq 4 \cdot \prod_{j=1}^{2} \left(\|g_j\|_2 + \|f_j\|_2 + 1 \right) \cdot \max_{j \in [2]} \left\{ \|f_j - g_j\|_2 \right\}. \end{aligned} \tag{10.43}$$

By definition, we have:

$$\begin{aligned} \mathop{\mathbb{E}}_{x \leftarrow \mathcal{S}} |P_F(x, f_1(x), f_2(x)) - P_F(x, g_1(x), g_2(x))| &= \mathop{\mathbb{E}}_{i \in [m]} \mathop{\mathbb{E}}_{x \leftarrow U_\ell} |\widehat{F}_i(x) - F_i(x)|, \\ \max_{j \in [2]} \left\{ \|f_j - g_j\|_2 \right\} &\leq \sum_{j \in [2]} \left(\left(\mathop{\mathbb{E}}_{i \in [m]} \|T_{ij} - \widehat{T}_{ij}\|_2^2 \right)^{1/2} \right), \\ \|f_j\|_2 &= \left(\mathop{\mathbb{E}}_{i \in [m]} \|T_{ij}\|_2^2 \right)^{1/2}, \\ \|g_j\|_2 &= 1. \end{aligned}$$

Therefore, (10.43) translates to

$$\begin{aligned} &\mathop{\mathbb{E}}_{i \in [m]} \mathop{\mathbb{E}}_{x \leftarrow U_\ell} |\widehat{F}_i(x) - F_i(x)| \\ &\leq 4 \cdot \sum_{j \in [2]} \left(\left(\mathop{\mathbb{E}}_{i \in [m]} \|T_{ij} - \widehat{T}_{ij}\|_2^2 \right)^{1/2} \right) \cdot \prod_{j \in [2]} \left(\left(\mathop{\mathbb{E}}_{i \in [m]} \|T_{ij}\|_2^2 \right)^{1/2} + 2 \right). \end{aligned} \tag{10.44}$$

Now, we make two observations. First, we have

$$\sum_{j \in [2]} \left(\left(\mathop{\mathbb{E}}_{i \in [m]} \|T_{ij} - \widehat{T}_{ij}\|_2^2 \right)^{1/2} \right) \leq 2 \left(\mathop{\mathbb{E}}_{i,j \in [m] \times [2]} \|T_{ij} - \widehat{T}_{ij}\|_2^2 \right)^{1/2} \leq 2 \left(\mathop{\mathbb{E}}_{i,j \in [m] \times [2]} \mathop{\mathbb{E}}_{x \leftarrow U_\ell} P_{ij}(x) \right)^{1/2}, \tag{10.45}$$

where the first step is due to $\sqrt{a} + \sqrt{b} \leq 2\sqrt{\frac{a+b}{2}}$, and the second step follows from $(1 - T_{ij}(x))^2(1 +$

[21] Recall that for each $x \in S$, P_F is the multi-linear extension of some Boolean function. So its coefficients are bounded by 4 (see Section 10.3).

$T_{ij}(x))^2 \geq (T_{ij}(x) - \widehat{T}_{ij}(x))^2$. We also have

$$\prod_{j\in[2]} \left(\left(\mathop{\mathbb{E}}_{i\in[m]} \|T_{ij}\|_2^2 \right)^{1/2} + 2 \right) \leq \left(\left(2 \mathop{\mathbb{E}}_{ij\in[m]\times[2]} \|T_{ij}\|_2^2 \right)^{1/2} + 2 \right)^2, \tag{10.46}$$

since $(\sqrt{a}+2)(\sqrt{b}+2) \leq (\sqrt{a}+\sqrt{b}+2)^2$. Finally, combining (10.44)-(10.46) with the conditions (10.39)-(10.40), it follows that

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} |\widehat{F}_i(x) - F_i(x)| \leq 200\sqrt{\delta}. \tag{10.47}$$

Now, it is clear that for $\delta < \sqrt{\frac{c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}}{1000}}$, combining (10.41), (10.42) and (10.47) leads to a contradiction. This completes the proof. ■

10.8.3 Extract Hardness

Now we prove Lemma 10.5.10.

Recall that for an input z of length n, we let $\ell = \log T(n) + O(\log\log T(n))$ be the input length to VPCP_z^C and its oracle. For a circuit C, we apply a PCPP construction to VPCP_z^C and use $m = \mathrm{poly}(|\mathsf{VPCP}_z^C|)$ to denote the number of clauses.

Reminder of Lemma 10.5.10. *For every small enough constants $\alpha, \delta \in (0,1)$ and for every sufficiently large constant $K \geq 1$, the following holds: Suppose for some input z of length n such that $\mathcal{A}_{\mathsf{cheat}}(z) = 0$, there is a circuit C of size at most $S(\ell)^\alpha$ such that VPCP_z^C is a tautology with $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ being its correct Boolean-valued proof. Then the function $H^{\widehat{Y},\widehat{Z}}$ defined by*

$$\begin{aligned} H^{\widehat{Y},\widehat{Z}} \colon \{0,1\}^{\log(m)+1+\ell} &\to \{-1,1\} \\ (i,j,x) &\mapsto \widehat{T}_{ij}(x) \end{aligned} \tag{10.48}$$

is hard in the following sense: letting $r = \log(m) + 1 + \ell$, for every $\mathsf{Sum} \circ \mathcal{F}_r$-functions H such that $\mathsf{complexity}(H) \leq S(r)^{3\alpha}$ and $\|H\|_4 \leq 1$, it holds that

$$\langle H^{\widehat{Y},\widehat{Z}}, H\rangle < (1 - \delta/5). \tag{10.49}$$

Proof of Lemma 10.5.10. Suppose on the contrary that there exists a $\mathsf{Sum} \circ \mathcal{F}_r$-function $H \colon \{0,1\}^r \to$

$\mathbb{R}$ with $\mathsf{complexity}(H) \leq S(r)^{3\alpha}$ and $\|H\|_4 \leq 1$ such that (10.49) does not hold. We show that the algorithm $\mathcal{A}_{\mathsf{cheat}}$ can pass the final verification given H being the guessed function, and this contradicts the assumption. We assume without loss of generality that $\|H\|_4 = 1$. (If not, just do some scaling.)

Recall that for every $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, $\mathcal{A}_{\mathsf{cheat}}$ constructed the following functions.

$$\begin{aligned} Y_s(x) &:= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathcal{T}_{ij}=\mathcal{Y}_s} H(i,j,x), \\ Z_t(x) &:= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathcal{T}_{ij}=\mathcal{Z}_t} H(i,j,x). \end{aligned} \tag{10.50}$$

Recall that $T_{ij} \in (Y \cup Z)$ denotes the proof function corresponding to the variable $\mathcal{T}_{ij} \in \mathcal{Y} \cup \mathcal{Z}$ and $F_i = \widetilde{\mathsf{Cons}}_i(T_{i1}, T_{i2})$. Also recall that we defined $\widehat{T}$ and $\widehat{F}$ for $(\widehat{Y}, \widehat{Z})$ similarly.

To show that H can pass the verification, we need to verify the following (rewriting (10.30) - (10.32)):

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} P_{i,j}(x) \leq \delta, \tag{10.51}$$

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} H(i,j,x)^2 \leq 1, \tag{10.52}$$

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} F_i(x) \geq c_{\mathsf{pcpp}} - \frac{1}{2}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}). \tag{10.53}$$

Here $P_{ij}(x)$ is defined as

$$P_{ij}(x) = \begin{cases} (1 - T_{ij}(x)^2)^2, & \text{if } T_{ij} \in Z, \\ (\mathsf{Enc}_s(x) - T_{ij}(x))^2, & \text{if } T_{ij} \in Y. \end{cases}$$

(10.52) clearly holds since $\|H\|_4 \leq 1$. (10.51) is a little bit tricky. First, by definition (10.50) we observe that

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} [T_{i,j}(x) \cdot \widehat{T}_{i,j}(x)] = \langle H^{\widehat{Y},\widehat{Z}}, H \rangle \geq (1 - \delta/5). \tag{10.54}$$

$$\begin{aligned} \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} |T_{ij}(x)|^2 &\geq \left(\mathop{\mathbb{E}}_{i,j} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} |T_{ij}(x)| \right)^2 && \text{(by Jensen's inequality)} \\ &\geq \left(\mathop{\mathbb{E}}_{i,j} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} [T_{i,j}(x) \cdot \widehat{T}_{i,j}(x)] \right)^2 && (\widehat{T}_{i,j}(x) \in \{-1, 1\}) \\ &\geq (1 - 2\delta/5). \end{aligned} \tag{10.55}$$

By definition of P_{ij}, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{ij\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} P_{ij}(x) &\le \mathop{\mathbb{E}}_{ij} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} \left((1-T_{ij}(x)^2)^2 + (T_{ij}(x)-\widehat{T}_{ij}(x))^2\right) \\
&\le \mathop{\mathbb{E}}_{ij} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} \left(T_{ij}(x)^4 + 2 - 2\cdot T_{ij}(x)\cdot \widehat{T}_{ij}(x) - T_{ij}(x)^2\right) \\
&\le \|H\|_4^4 + 2 - \mathop{\mathbb{E}}_{ij} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} \left(2\cdot T_{ij}(x)\cdot \widehat{T}_{ij}(x) + T_{ij}(x)^2\right) \\
&\le 1 + 2 - (2(1-\delta/5) + (1-2\delta/5)) && \text{(by (10.54) and (10.55))} \\
&\le \delta.
\end{aligned}$$

Finally we verify (10.53). We note that

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \|T_{ij}\|_2^2 \le \|H\|_2^2 \le 1. \tag{10.56}$$

Combining this fact with (10.54), we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \|T_{ij}-\widehat{T}_{ij}\|_2^2 &= \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} (T_{ij}(x)-\widehat{T}_{ij}(x))^2 \\
&= \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left(\|T_{ij}\|_2^2 + \|\widehat{T}_{ij}\|_2^2 - 2\langle T_{ij}, \widehat{T}_{ij}\rangle\right) \\
&\le 1 + 1 - 2(1-\delta/5) && \text{(by (10.54) and (10.56))} \\
&\le \frac{2}{5}\delta. && (10.57)
\end{aligned}$$

Now, given (10.52) and (10.57), we can bound $\mathbb{E}_{i\in[m]} \mathbb{E}_{x\leftarrow U_\ell} |F_i - \widehat{F}_i|$ by utilizing Lemma 10.8.1. This step is very similar to the application of Lemma 10.8.1 in the proof of Lemma 10.5.9. In particular, we can show that

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} [|F_i(x) - \widehat{F}_i(x)|] \le 200\sqrt{\delta}.$$

Since $(\widehat{Y}, \widehat{Z})$ is the correct proof to VPCP^C_z, it follows that

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} [\widehat{F}_i(x)] \ge c_{\mathsf{pcpp}}.$$

Hence, for sufficiently small $\delta > 0$, it holds that

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\leftarrow U_\ell} [F_i(x)] \ge c_{\mathsf{pcpp}} - \frac{1}{2}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).$$

In conclusion, we showed that $\mathcal{A}_{\mathsf{cheat}}$ on the PCPP of VPCP_z^C accepts H as a proof, and $\mathcal{A}_{\mathsf{cheat}}(z) = 1$. This contradicts the assumption. ■

10.9 Missing Proofs in Section 10.7

10.9.1 The Proof of Lemma 10.7.7

Reminder of Lemma 10.7.7. *For two matrices $H \in \mathbb{R}^{N_1 \times N_1}$ and $H' \in \{-1,1\}^{N_1 \times N_1}$, it holds that*

$$\mathop{\mathbb{E}}_{r \leftarrow U_\gamma} \left| \widetilde{\mathsf{VrecPCP}}_z^H(r) - \mathsf{VrecPCP}_z^{H'}(r) \right| \leq 2^q \cdot q \cdot \left(2 + q^{1/(2q-2)} \|H\|_{2q-2}\right)^{2q-2} \|H - H'\|_2.$$

Proof. For every $r \in \{0,1\}^\gamma$ and $i \in [q]$, let the index of the i-th query to the proof be $(x_{r,i}, y_{r,i})$. Then we define q functions $H_1, \ldots, H_q \colon \{0,1\}^\gamma \to \mathbb{R}$ by $H_i(r) := H(x_{r,i}, y_{r,i})$ for every $i \in [q]$. We also define q Boolean functions $H'_1, \ldots, H'_q \colon \{0,1\}^\gamma \to \{-1,1\}$ by $H'_i(r) = H'(x_{r,i}, y_{r,i})$ for every i.

By the smoothness property of $\mathsf{VrecPCP}_z$ (Theorem 10.7.6), we have

$$\mathop{\mathbb{E}}_{i \leftarrow [q]} \|H_i\|_{2q-2}^{2q-2} = \|H\|_{2q-2}^{2q-2},$$
$$\mathop{\mathbb{E}}_{i \leftarrow [q]} \|H_i - H'_i\|_2^2 = \|H - H'\|_2^2.$$

Hence, for every $i \in [q]$, it holds that

$$\|H_i - H'_i\|_2 \leq q \cdot \|H - H'\|_2, \tag{10.58}$$
$$\|H_i\|_{2(q-1)} \leq q^{1/(2q-2)} \|H\|_{2(q-1)}. \tag{10.59}$$

Recall that $\widetilde{\mathsf{VrecPCP}}_z^H$ is the natural arithmetization of $\mathsf{VrecPCP}$: for each $r = (r_{\mathsf{row}}, r_{\mathsf{col}}, r_{\mathsf{shared}})$, $\widetilde{\mathsf{VrecPCP}}_z^H$ reads q values from the proof H, which are $H_1(r), \ldots, H_q(r)$ by definition, and outputs $P_r(H_1(r), \ldots, H_q(r))$ where $P_r \colon \mathbb{R}^q \to \mathbb{R}$ is the polynomial mapping the query answers to the decision. Observe that P_r is the multi-linear extension of a Boolean function[22]. So its coefficients are bounded by 2^q.

[22] Check Section 10.3 for the relevant discussion.

Therefore, we can apply Lemma 10.8.1 to the function $\widetilde{\mathsf{VrecPCP}}_z$, with the set $\mathcal{S} := \{0,1\}^\gamma$ and two lists of functions $(H_1, \ldots, H_q)$ and $(H'_1, \ldots, H'_q)$. This completes the proof. ■

10.9.2 The Proof of Lemma 10.7.8

Reminder of Lemma 10.7.8. *For every $\tau \geq 1$, there is a sufficiently small $\alpha > 0$ such that the following is true. For every $z \in \{0,1\}^*$ and H being a $\mathsf{Sum} \circ \mathcal{M}_\ell^{2^{\tau \ell^{1-c}}}$-function with $\mathsf{complexity}(H) \leq 2^{\alpha \ell^c}$, evaluations of the left-hand sides of (10.36)-(10.38) can be done in $2^{\ell - \Omega(\ell^c/\alpha)} \leq o(n^K)$ time.*

Before we proceed, we introduce the following lemma. It will be proved in the end of this subsection.

Lemma 10.9.1. *For every $n, d, r, t \in \mathbb{N}_{\geq 1}$, let $P \colon \mathbb{R}^d \to \mathbb{R}$ be a multi-linear polynomial. Let $H_1, \ldots, H_d \colon \{0,1\}^n \to \mathbb{R}$ be a list of $\mathsf{Sum} \circ \mathcal{M}_n^r$-functions with $\mathsf{complexity}(H_i) \leq t$ for every $i \in [d]$. Then the evaluation*

$$\mathop{\mathbb{E}}_{x \leftarrow U_n} P(H_1(x), \ldots, H_d(x))$$

reduces to at most $(2t)^d$-many $\#\mathsf{SAT}$ tasks for $\mathcal{M}_n^{dr}$-functions. The reduction can be computed in $O((2t)^d \cdot 2^{n/2} \cdot d \cdot r)$ time.

Proof of Lemma 10.7.8. First, we recall the left-hand sides of (10.36)-(10.38):

$$\mathop{\mathbb{E}}_{(x,y) \leftarrow U_\ell} (1 - H(x,y))^2 (1 + H(x,y))^2 \tag{10.60}$$

$$\mathop{\mathbb{E}}_{(x,y) \leftarrow U_\ell} H(x,y)^{2q-2} \tag{10.61}$$

$$\mathop{\mathbb{E}}_{r \leftarrow U_\gamma} \left[\widetilde{\mathsf{VrecPCP}}_z^H(r) \right] \tag{10.62}$$

Note that for (10.60) and (10.61), we have to calculate expectations of constant-degree polynomials (over $\mathbb{R}$) of H.

For (10.62), we introduce some notation. For every $r = (r_{\mathsf{row}}, r_{\mathsf{col}}, r_{\mathsf{shared}})$ and for $i \in [q]$, let $x_{r,i}, y_{r,i}$ be the row index and column index of the i-th query given randomness r. Recall that we wrote the output of $\mathsf{VrecPCP}_z$ as a multi-linear polynomial (over $\mathbb{R}$) of the query answers $(v_1, \ldots, v_q)$ and the parity check $(c_1, \ldots, c_p)$, denoted by $Q_{r_{\mathsf{shared}}} \colon \mathbb{R}^{q+p} \to \mathbb{R}$, which maps $(v_1, \ldots, v_q, c_1, \ldots, c_p)$ to a bit in $\{0,1\}$. Moreover, for every $r = (r_{\mathsf{shared}}, r_{\mathsf{row}}, r_{\mathsf{col}})$, we wrote the decision as a multi-linear polynomial of queried bits, denoted by $P_r \colon \mathbb{R}^q \to \mathbb{R}$.

Now, fixing r_{shared}, for every $i \in [q]$, we define a function $H_i^{r_{\mathsf{shared}}}$ as

$$H_i^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}}) := H(x_{r,i}, y_{r,i}).$$

Since $x_{r,i}$ (resp. $y_{r,i}$) only depends on r_{row} (resp. r_{col}), we conclude that $H_i^{r_{\mathsf{shared}}} \in \mathsf{Sum} \circ \mathcal{M}^{2^{\tau\ell^{1-c}}}_{|r_{\mathsf{row}}|+|r_{\mathsf{col}}|}$ with $\mathsf{complexity}(H_i^{r_{\mathsf{shared}}}) \leq 2^{\alpha\ell^c}$, and the description of $H_i^{r_{\mathsf{shared}}}$ can be computed in $\widetilde{O}(2^{\ell/2})$ time. Also recall that for every $j \in [p]$ and $(r_{\mathsf{row}}, r_{\mathsf{col}})$, we defined $C_j^{r_{\mathsf{shared}}}(r_{\mathsf{row}}, r_{\mathsf{col}})$ to be the result of the j-th parity check given randomness r. Note that we have $C_j^{r_{\mathsf{shared}}} \in \mathcal{M}^2_{|r_{\mathsf{row}}|+|r_{\mathsf{col}}|}$. What we want to compute can be written as

$$\mathop{\mathbb{E}}_{(r_{\mathsf{row}}, r_{\mathsf{col}})}[P_r(H_1^{r_{\mathsf{shared}}}, \ldots, H_q^{r_{\mathsf{shared}}})] = \mathop{\mathbb{E}}_{(r_{\mathsf{row}}, r_{\mathsf{col}})}[Q_{r_{\mathsf{shared}}}(H_1^{r_{\mathsf{shared}}}, \ldots, H_q^{r_{\mathsf{shared}}}, C_1^{r_{\mathsf{shared}}}, \ldots, C_p^{r_{\mathsf{shared}}})]. \quad (10.63)$$

Fixing $\tau \geq 1$, we can choose $\alpha > 0$ being sufficiently small constant such that the following argument holds. First, assuming Lemma 10.9.1 and given $\mathsf{complexity}(H_i) \leq 2^{\alpha\ell^c}$, the evaluations of (10.60) and (10.61) reduce to solving $2^{O(\alpha\ell^c)}$ #SAT tasks for $\mathcal{M}_\ell^{2q2^{\tau\ell^{1-c}}}$-functions, which can be done in

$$2^{\ell - \Omega(\ell^c/\tau) + O(\alpha\ell^c)} = 2^{\ell - \Omega(\ell^c/\tau)} \leq o(n^K)$$

time, by Lemma 10.7.4.

For (10.62), note that its evaluation reduces to calculating for each r_{shared} the right-hand side of (10.63). For every fixed r_{shared}, by Lemma 10.9.1, the evaluation of (10.63) reduces to solving $2^{O(\alpha\ell^c)}$ #SAT tasks for $\mathcal{M}^{(q+1)2^{\tau\ell^{1-c}}}_{|r_{\mathsf{row}}|+|r_{\mathsf{col}}|}$-functions (since p and q are constants). Since $|r_{\mathsf{row}}| \geq \Omega(\ell)$, again by Lemma 10.7.4, evaluating (10.63) can be done in

$$2^{|r_{\mathsf{row}}|+|r_{\mathsf{col}}| - \Omega(\ell^c/\tau) + O(\alpha\ell^c)}$$

time. Enumerating r_{shared}, the evaluation of (10.62) can be done in

$$2^{|r_{\mathsf{shared}}|+|r_{\mathsf{row}}|+|r_{\mathsf{col}}| - \Omega(\ell^c/\tau) + O(\alpha\ell^c)} = 2^{\gamma - \Omega(\ell^c/\tau)} \leq o(n^K)$$

time, which completes the proof.

■

Now we prove Lemma 10.9.1.

Proof of Lemma 10.9.1. We first show that for the special case when $P(v_1, \ldots, v_d) = \prod_{i=1}^{d} v_i$, the desired evaluation in the lemma can be reduced to solving t^d #SAT tasks. The general case can be handled by considering monomials in P one by one (there are at most 2^d monomials in a multi-linear polynomials on d variables). First, for each $i \in [q]$, we write

$$H_i(x) = \sum_{j=1}^{t} \alpha_{i,j} H_{i,j}(x).$$

Note that here we assume without loss of generality that all of $H_i(x)$ have sparsity exactly t. Then we have

$$\mathop{\mathbb{E}}_{x \leftarrow U_n} \prod_{i=1}^{d} H_i(x) = \sum_{(\ell_1, \ldots, \ell_d) \in [t]^d} \left[\prod_{i \in [d]} \alpha_{i,\ell_i} \left(\mathop{\mathbb{E}}_{x \leftarrow U_n} \prod_{i=1}^{d} H_{i,\ell_i}(x) \right) \right].$$

For two functions $f, g \in \mathcal{M}_n^r$, we define two matrices $M_f, M_g \in \mathbb{F}_2^{2^{r/2} \times 2^{r/2}}$ such that $f(x,y) = (-1)^{M_f(x,y)}$ and $g(x,y) = (-1)^{M_g(x,y)}$ for $(x,y) \in \{0,1\}^n$. Then we have $f(x,y) \cdot g(x,y) = (-1)^{M_f(x,y) + M_g(x,y)}$, where the addition of M_f and M_g is over $\mathbb{F}_2$. It follows that $f \cdot g \in \mathcal{M}_n^{2r}$. Moreover, if we have the descriptions of f and g, denoted by $M_f = A_f \cdot B_f^T$ and $M_g = A_g \cdot B_g^T$, then the description of $f(x,y) \cdot g(x,y)$ is just $(A_f, A_g) \cdot (B_f, B_g)^T$, where we use $(A_f, A_g), (B_f, B_g)$ to denote the concatenations of matrices with same number of rows.

Hence, for every tuple $(\ell_1, \ldots, \ell_d) \in [n^d]$, the function $\prod_{i=1}^{d} H_{i,\ell_i}(x)$ is in $\mathcal{M}_n^{dr}$, and its description can be computed in $O(2^{n/2} \cdot r \cdot d)$ time. This completes the proof. ■

10.9.3 The Proof of Lemma 10.7.9

Reminder of Lemma 10.7.9. *For every sufficiently small $\delta > 0$, it holds that $\mathcal{A}_{\mathsf{matrix}}(z) \leq \mathcal{A}_{\mathsf{FS}}^T(z)$.*

Proof of Lemma 10.7.9. Suppose on the contrary that there exists $z \in \{0,1\}^*$ such that $\mathcal{A}_{\mathsf{FS}}^T(z) = 0$ but $\mathcal{A}_{\mathsf{matrix}}(z) = 1$. We show a contradiction. By $\mathcal{A}_{\mathsf{matrix}}(z) = 1$ we know that there is a function $H \colon \{0,1\}^\ell \to \mathbb{R}$ such that the following hold (rewriting (10.36) - (10.38)):

$$\mathop{\mathbb{E}}_{(x,y) \leftarrow U_\ell} (1 - H(x,y))^2 (1 + H(x,y))^2 \leq \delta, \tag{10.64}$$

$$\mathop{\mathbb{E}}_{(x,y) \leftarrow U_\ell} H(x,y)^{2q-2} \leq 1, \tag{10.65}$$

$$\mathop{\mathbb{E}}_{r \leftarrow U_\gamma} \left[\widetilde{\mathsf{VrecPCP}}_z^H(r) \right] \geq \frac{1+s}{2}. \tag{10.66}$$

We define from H a Boolean function H' as $H'(x,y) = \mathrm{sign}(H(x,y))$. It follows from (10.64) that $\|H - H'\|_2 \le \|(1-H)(1+H)\|_2 \le \sqrt{\delta}$.

Using Lemma 10.7.7, it follows from (10.65) that

$$\mathop{\mathbb{E}}_{r\leftarrow U_\gamma}\left|\widetilde{\mathsf{VrecPCP}}_z^H(r) - \mathsf{VrecPCP}_z^{H'}(r)\right| \le 2^q \cdot q \cdot \left(2 + q^{1/(2q-2)}\|H\|_{2q-2}\right)^{2q-2}\|H - H'\|_2 \le 2^{O(q)}\sqrt{\delta}. \tag{10.67}$$

Since $\mathcal{A}_{\mathsf{FS}}^T(z) = 0$, by the soundness property of Theorem 10.7.6, it follows that

$$\mathop{\mathbb{E}}_{r\leftarrow U_\gamma}\left[\mathsf{VrecPCP}_z^{H'}(r)\right] < s. \tag{10.68}$$

Note that for sufficiently small δ ($\delta \ll (1-s)$), combining (10.66)-(10.68) leads to a contradiction. This completes the proof. ■

10.9.4 The Proof of Lemma 10.7.10

Reminder of Lemma 10.7.10. *For every sufficiently small $\alpha > 0$, the following is true. For every z such that $\mathcal{A}_{\mathsf{matrix}}(z) = 0$ and $\mathcal{A}_{\mathsf{FS}}^T(z) = 1$, every correct proof for $\mathsf{VrecPCP}_z$ is a function $H'\colon \{0,1\}^\ell \to \{-1,1\}$ such that, for every $\mathsf{Sum}\circ\mathcal{M}_\ell^{2^{\tau\ell^{1-c}}}$-function H with $\mathsf{complexity}(H) \le 2^{\alpha\ell^c}$, it holds that $\langle H, H'\rangle \le (1-\delta/5)\|H\|_{2(q-1)}$.*

Proof. Suppose on the contrary that there exists an $H\colon \{0,1\}^\ell \to \mathbb{R}$ guessed by $\mathcal{A}_{\mathsf{matrix}}(z)$ that violates the lemma statement. In the following, we assume without loss of generality that $\|H\|_{2(q-1)} = 1$. (If not, just do a scaling.) We show that $\mathcal{A}_{\mathsf{matrix}}(z)$ accepts H as a proof, which contradicts the assumption that $\mathcal{A}_{\mathsf{matrix}}(z) = 0$.

We verify that after guessing H, $\mathcal{A}_{\mathsf{matrix}}(z)$ can pass the following tests (rewriting (10.36)-(10.38)):

$$\mathop{\mathbb{E}}_{(x,y)\leftarrow U_\ell}(1 - H(x,y))^2(1 + H(x,y))^2 \le \delta, \tag{10.69}$$

$$\mathop{\mathbb{E}}_{(x,y)\leftarrow U_\ell} H(x,y)^{2q-2} \le 1, \tag{10.70}$$

$$\mathop{\mathbb{E}}_{r\leftarrow U_\gamma}\left[\widetilde{\mathsf{VrecPCP}}_z^H(r)\right] \ge \frac{1+s}{2}. \tag{10.71}$$

In the following, we assume that $q \geq 3$. If it is not the case, we can construct an equivalent PCP which issues some additional queries to proof and ignore the answer.

First, (10.70) holds since we have assumed that $\|H\|_{2q-2} = 1$. For (10.71), we observe that

$$\|H\|_2^2 \leq \|H\|_{2q-2}^2 \leq 1$$

and

$$\langle H, H' \rangle > (1 - \delta/5)\|H\|_{2q-2} \geq 1 - \delta/5.$$

Therefore, we have that

$$\|H - H'\|_2^2 = \|H\|_2^2 + \|H'\|_2^2 - 2\langle H, H' \rangle \leq \frac{2}{5}\delta. \tag{10.72}$$

Note that δ measures the distance between H and H'. The less δ is, the closer H is to H'. Hence, for sufficiently small δ, it follows from Lemma 10.7.7, (10.70) and (10.72) that

$$\mathop{\mathbb{E}}_{r \leftarrow U_\gamma}\left[\widetilde{\mathsf{VrecPCP}}_z^H(r)\right] \geq \mathop{\mathbb{E}}_{r \leftarrow U_\gamma}\left[\mathsf{VrecPCP}_z^{H'}(r)\right] - 2^{O(q)} \cdot \delta \geq \frac{1+s}{2}.$$

Lastly, we verify (10.69). Note that $\|H\|_2 \geq \|H\|_1 \geq \langle H, H' \rangle \geq 1 - \delta/5$. It follows that

$$\begin{aligned}
\|(H+1)(H-1)\|_2^2 &= \|1 - H^2\|_2^2 \\
&= \mathop{\mathbb{E}}_{(x,y) \leftarrow U_\ell}\left[H(x,y)^4 + 1 - 2H(x,y)^2\right] \\
&= \|H\|_4^4 + 1 - 2\|H\|_2^2 \\
&\leq 2 - 2(1 - 2\delta/5) && (\|H\|_4 \leq \|H\|_{2q-2} = 1) \\
&\leq \delta.
\end{aligned}$$

This shows that H can pass the tests and consequently $\mathcal{A}_{\mathsf{matrix}}(z) = 1$, a contradiction.

■

Part III

Hardness vs. Randomness Revised: Superfast and Non-black-box Derandomization

Chapter 11

Near-optimal Derandomization from Very Hard Functions

Contents

11.1 Introduction

Can we replace all randomized algorithms for decision problems by deterministic algorithms with roughly similar runtime? The long line of works typically referred to as "hardness-to-randomness", which was initiated by [Yao82, BM84, NW94], gives one way of answering the foregoing question: These works show that certain *lower bounds* for non-uniform circuits imply derandomization with bounded *runtime overhead*.

The fastest conditional derandomization in the classical line-of-works was proved by Impagliazzo and Wigderson [IW97], who showed that if $\mathsf{E} \not\subset SIZE[2^{.01 \cdot n}]$, then $\mathsf{prBPP} = \mathsf{prP}$. By a padding argument, this conclusion implies that randomized time-T algorithms can be simulated in deterministic time $T(n)^c$ for some large constant $c \in \mathbb{N}$. In other words, they showed that when solving decision problems, randomness can be deterministically simulated with a *polynomial runtime overhead*.

In a recent exciting work, Doron, Moshkovitz, Oh, and Zuckerman [DMOZ20] asked if an *even faster* derandomization is possible: Could we prove that derandomization with a *small* polynomial overhead (*i.e.*, derandomization in time $T(n)^c$ for a small value of c) follows from plausible hypotheses? Taken to the extreme, could it be that randomized algorithms for decision problems can be deterministically simulated with *almost no runtime overhead*? Their main result is that derandomization with only a *quadratic* time overhead (*i.e.*, $c \approx 2$) is possible under a certain lower

bound hypothesis; specifically, this conclusion follows from the hypothesis that $\mathsf{DTIME}[2^n]$ is hard for *randomized and nondeterministic circuits* of very large size $\approx 2^{.99 \cdot n}$. That is:

Theorem 11.1.1 (derandomization with quadratic overhead [DMOZ20])**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \in \mathsf{DTIME}[2^n]$ that cannot be computed by randomized SVN circuits[1] of size $2^{(1-\delta) \cdot n}$, even infinitely-often. Then, for every time-constructible $T \colon \mathbb{N} \to \mathbb{N}$ such that $T(n) \geq n$ we have that $\mathsf{prBPTIME}[T(n)] \subseteq \mathsf{prDTIME}[T(n)^{2+\varepsilon}]$.*

The result of Doron *et al.* [DMOZ20] provides evidence that extremely fast derandomization might be possible, and moreover opens the door to asking if an even *faster* derandomization, namely with overhead $\approx T(n)^{1.01}$, might also be possible. However, the hypothesis in Theorem 11.1.1 is considerably stronger than the hypotheses in the classical works: It refers to circuits that are not only larger (*i.e.*, of size $2^{.99 \cdot n}$ rather than $2^{.01 \cdot n}$), but that also use randomness and non-determinism; that is, it refers to a lower bound for the non-uniform analogue of $\mathsf{MA} \cap \mathsf{coMA}$. Needless to say, these additional resources might significantly increase the power of non-uniform circuits.[2] Moreover, in contrast to classical "hardness-to-randomness" works, the hypothesis in Theorem 11.1.1 is not known to be *necessary* for the existence of the corresponding PRG that they construct.

11.1.1 Our Contributions: Bird's Eye View

This chapter extends the line of inquiry opened by [DMOZ20], focusing on the possibility of extremely fast derandomization, and provides answers to several open questions that arose from their work. We now describe our results informally and in high-level, and later on we will elaborate in more detail.

1. Simulating randomness with very small overhead. The main open question following Theorem 11.1.1 is whether or not we can derandomize probabilistic algorithms in time that is close to linear $T(n)^{1+\varepsilon}$, rather than in quadratic time $T(n)^{2+\varepsilon}$.

We provide an *affirmative answer to this question*, conditioned on a plausible hypothesis, which is formally incomparable to the one in Theorem 11.1.1 but is arguably more standard. Specifically, we show that probabilistic algorithms running in time $T(n)$ can be deterministically sim-

[1] Randomized SVN circuits are the non-uniform analogue of $\mathsf{MA} \cap \mathsf{coMA}$; see Definition 11.3.4 for details.

[2] For a recent demonstration of the power of MA algorithms that run in exponential time, see [Wil16b]; we refer the reader to the corresponding discussion in [DMOZ20, Section 1.6].

ulated in time $n \cdot T(n)^{1+\varepsilon}$, conditioned on the following: There exist one-way functions secure against polynomial-sized circuits, and there exists a problem in time $2^{k \cdot n}$ that is hard for algorithms that run in time $2^{(k-.01) \cdot n}$ and use $2^{.99n}$ bits of non-uniform advice, where k is a sufficiently large constant (see Theorem 11.1.2, and see the subsequent discussion for a comparison with Theorem 11.1.1).

The second assumption may be viewed as a stronger version of the classical time-hierarchy theorem, asserting that the time-hierarchy holds even when the "weaker" class is given a near-maximal amount of non-uniform advice. This assumption (or, more accurately, a relaxation of it that we use) is essentially *necessary* to obtain the derandomization conclusion using PRGs (see Section 11.1.2), and is a natural extension of hardness hypotheses from classical "hardness-to-randomness" results.

2. Optimizing the time overhead. We further improve the derandomization time to $n^{1+\varepsilon} \cdot T(n)$ for natural special cases, conditioned on hypotheses that are slightly more technically involved and/or mildly stronger (see Theorem 11.1.4). This time bound almost matches the straightforward *non-uniform* derandomization, and we prove that it is essentially optimal, under a "counting" version of the nondeterministic strong exponential time hypothesis (*i.e.*, under #NSETH; see Theorem 11.1.3).

3. Average-case derandomization with almost no overhead. Bypassing the conditional lower bound in Theorem 11.1.3, we show a faster derandomization for polynomial-time algorithms that succeeds on *average case*, rather than in the worst-case. Specifically, our derandomization algorithm runs in time $n^{\varepsilon} \cdot T(n)$, and succeeds with probability $1 - n^{-\omega(1)}$ with respect to all distributions samplable in time $T(n)$ (see Theorem 11.1.7).

4. Fast derandomization via simple proof paradigms. The proof of Theorem 11.1.1 in [DMOZ20] is highly non-trivial, relying on refined technical notions and on complicated analyses. In contrast, all of our results rely on *simple and intuitive proofs* that use only standard technical tools. In particular, the results mentioned above rely on proof strategies that are significantly different than the ones in [DMOZ20].

We also present a proof for Theorem 11.1.1 that is simpler than the one in [DMOZ20] and relies on an observation of independent interest: Any PRG construction that is analyzed as "extracting randomness from a pseudoentropic string" (in particular, the construction of [DMOZ20]) can be

analyzed via a different proof strategy that is both *simpler* and *more general* (see Section 11.1.4). As one application, we extend Theorem 11.1.1 by showing that derandomization with either *cubic* or *quartic* overhead is possible, under hardness assumptions that are similar to the one in Theorem 11.1.1, yet refer only to SVN circuits that *do not use randomness* (see Theorem 11.1.8).

11.1.2 Worst-case Derandomization with Very Small Overhead

Our first result is that under a plausible hardness hypothesis, probabilistic algorithms that run in time $T(n)$ can be deterministically simulated in time $n \cdot T(n)^{1+\varepsilon}$, for an arbitrarily small constant $\varepsilon > 0$ and for all time bounds $T(n)$.

Similarly to other results in the "hardness-to-randomness" line-of-works, our derandomization relies on a PRG construction. However, the standard approach of constructing a PRG that "fools" non-uniform circuits of size $T(n)$ cannot work here, because such a PRG requires a seed of length at least $\log(T(n))$ (and evaluating a time-T algorithm at each output of the PRG requires time $T(n)^2$). Our way to bypass this obstacle is to observe that the standard approach is an "overkill": When transforming a probabilistic algorithm into a distinguisher for the PRG, the distinguisher runs in time $T(n)$ but *only uses n bits of non-uniform advice* (*i.e.*, the advice corresponds to the input, which is of length n rather than $T(n)$). Thus, it suffices to "fool" the foregoing class of distinguishers (see Proposition 11.4.7), and indeed there exists a non-explicit PRG for this class with *sub-logarithmic* seed length; that is, the distinguisher class can be modeled by $\mathsf{DTIME}[N]/T^{-1}(N)$ (where the notation alludes to $N = T(n)$), and it can be "fooled" by a (non-explicit) PRG with seed length $(1 + o(1)) \cdot \log(T^{-1}(N))$. [3]

In our first result we construct a PRG that yields derandomization in time $n \cdot T(n)^{1+\varepsilon}$, conditioned on the following. First, we assume that there exist non-uniformly secure one-way functions; recall that this assumption is not known to imply (by itself) derandomization in less than sub-exponential time. Secondly, we assume that there exists a problem decidable in time $2^{k \cdot n}$ that cannot be solved in time $2^{(k-\delta) \cdot n}$ with $2^{(1-\delta) \cdot n}$ bits of non-uniform advice, where δ is sufficiently small and k is sufficiently large. As mentioned above, the latter hypothesis can be interpreted as a strengthening of the classical time-hierarchy theorem, since it asserts that (loosely speaking) there are problems solvable in time $T(n) = 2^{k \cdot n}$ that cannot be solved in time slightly smaller than $T(n)$ even with a near-maximal amount of non-uniform advice.

[3] To see this, for every input length N we "fool" the first $\varepsilon(N)$ machines that run in time N, instantiated with every possible advice, where ε is any super-constant function. This yields at most $\varepsilon(N) \cdot 2^{T^{-1}(N)}$ distinguishers, and therefore a seed of length $\log(T^{-1}(N)) + \log(\varepsilon(N))$ suffices.

Theorem 11.1.2 (derandomization with very small overhead for all probabilistic algorithms)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Let $T\colon \mathbb{N} \to \mathbb{N}$ be any time-constructible non-decreasing function, and let $k = k_{\varepsilon,T} \geq 1$ be a sufficiently large constant.*[4] *Assume that there exist one-way functions that are secure against polynomial-sized circuits, and that there exists $L \in \mathsf{DTIME}[2^{k \cdot n}]$ such that $L \notin$ i.o.-$\mathsf{DTIME}[2^{(k-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, we have that $\mathsf{prBPTIME}[T(n)] \subseteq \mathsf{prDTIME}[n \cdot T(n)^{1+\varepsilon}]$.*

The non-cryptographic hypothesis in Theorem 11.1.2 can be relaxed, to only require that the *amortized* time-complexity of L when printing its entire truth-table will be $2^{k \cdot n}$ (rather than requiring each entry in the truth-table to be computable in such time; see Theorem 11.4.10). The reason that we mention this relaxation is that *the existence of such an L is necessary for the PRG conclusion* (see Proposition 11.1.6 and Theorem 11.4.11). This relaxation will apply to all results in the current section, but for simplicity we will avoid mentioning it explicitly, and defer the full result statements to the technical section.

Thus, one of our assumptions in Theorem 11.1.2 is *necessary* for the conclusion, and the other is a standard and fundamental one. Moreover, both assumptions refer only to standard computational models that *do not use non-determinism or randomness* (rather than to non-uniform analogues of $\mathsf{MA} \cap \mathsf{coMA}$). Thus, we argue that our hypothesis is more standard and appealing than the one in Theorem 11.1.1. Intuitively, the main part in our hypothesis that is stronger (and allows for faster derandomization) is that our hardness assumption refers to a separation of uniform algorithms from non-uniform procedures in a *"higher" time bound* (*i.e.*, in time $2^{k \cdot n}$ rather than in time 2^n).

Derandomizing "better-than-brute-force" algorithms. The benefit in derandomization as in Theorem 11.1.2 (compared to, say, derandomization in quadratic time) is particularly salient when considering randomized algorithms that run in time close to that of a "brute-force" algorithm. Many such "better-than-brute-force" randomized algorithms are known, for example for NP-complete graph problems (see, *e.g.*, [Bjö14, CKN18]), for satisfiability of formulas and circuits (see, *e.g.*, [PPSZ05, CSS16]), and for NP-complete algebraic problems (see, *e.g.*, [LPT+17]). For all these problems, derandomization in time $T(n)^{1+\varepsilon}$ would yield a better-than-brute-force deterministic algorithm, but derandomization in time $T(n)^2$ is trivial.[5]

[4] When T is a polynomial the constant k will be very close to the polynomial power of T, and when T is super-polynomial the constant k will be linear in $1/\varepsilon$; see Theorem 11.4.10.

[5] For incomparable results regarding derandomization of slow probabilistic algorithms, see [AIKS16].

Derandomization as in Theorem 11.1.2 also implies that lower bounds for deterministic algorithms yield near-identical lower bounds for randomized algorithms: For example, if for every $\varepsilon > 0$ the hypothesis of Theorem 11.1.2 holds, then the Strong Exponential-Time Hypothesis (SETH) is equivalent to its randomized version (*i.e.*, to rSETH).

Can we further reduce the overhead?

The derandomization time overhead of $T(n) \mapsto n \cdot T(n)^{1+\varepsilon}$ is small, but is it as small as it can be? We prove several additional results that address this question, both by improving the upper-bound and by showing a near-matching conditional lower bound.

Recall that the known *non-uniform* derandomization of $\mathsf{prBPTIME}[T]$ yields circuits of size $O(n \cdot T(n))$.[6] We first show that such an overhead is unavoidable, at least for uniform algorithms,[7] conditioned on a *counting* version of the Non-Deterministic Strong Exponential-Time Hypothesis (NSETH), which is weaker than NSETH itself. Specifically, extending a result of Williams [Wil16b], we prove that derandomization in time $O(n \cdot T(n))$ is optimal if for every $\varepsilon > 0$ there does not exist a nondeterministic machine that counts the number of satisfying assignments of a k-SAT formula over n bits in time $2^{(1-\varepsilon)\cdot n}$, assuming that $k = k_\varepsilon$ is sufficiently large (see Section 11.6).

Theorem 11.1.3 (conditional necessity of the multiplicative overhead of n)**.** *Assuming* #NSETH *we have that* $\mathsf{BPTIME}[T] \not\subseteq \mathsf{DTIME}[n^{1-\varepsilon} \cdot T(n)]$*, for every polynomial* T *and* $\varepsilon > 0$*.*

Matching Theorem 11.1.3, we improve the derandomization overhead in Theorem 11.1.2 to $n^{1+\varepsilon} \cdot T(n)$ for a large class of probabilistic algorithms, under hypotheses similar to the ones in Theorem 11.1.2. We state the following result in a slightly suboptimal way for simplicity, and mention afterwards how it can be improved.

Theorem 11.1.4 (derandomization with near-optimal overhead; informal)**.** *For every* $\varepsilon > 0$ *there exists* $\delta > 0$ *such that for every "nice"* $T(n) \leq 2^{n^{o(1)}}$ *the following holds. Assume that for some* $\gamma > 0$ *there exist one-way functions that are secure against circuits of size* 2^{n^γ}*, and that there exists* $L \in \mathsf{DTIME}[2^{\delta \cdot n} \cdot T'(n)]$ *such that* $L \notin$ *i.o.*-$\mathsf{DTIME}[T']/2^{(1-\delta)\cdot n}$*, where* $T'(n) = 2^{O(\delta \cdot n)} \cdot T(2^{(1-\delta)\cdot n})$*. Then,* $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[n^{1+\varepsilon} \cdot T(n)]$*.*

[6] This is since for every probabilistic algorithm and $n \in \mathbb{N}$, by a Chernoff bound there exist $O(n)$ fixed random strings that lead the algorithm to a correct decision on all inputs.

[7] Note that many problems can be solved in constant probabilistic time $T(n) = O(1)$ but require linear deterministic time $O(n)$ (*e.g.*, estimating the Hamming weight of the input). Nevertheless, the lower bound for this specific time bound does not rule out an *additive* derandomization overhead of n.

The hypothesis in Theorem 11.1.4 can be improved for polynomial time functions $T(n) = \text{poly}(n)$. Specifically, in this case we only need the hypothesized one-way function to be secure against polynomial-sized circuits (see Theorem 11.4.8 for details).

Batch-computable PRGs and problems with bounded *amortized* complexity

Recall that the results in this section are obtained by constructing PRGs for the class $\mathsf{DTIME}[O(N)]/T^{-1}(N)$, which (ideally) have seed length $\ell \approx \log(T^{-1}(N)) = \log(n)$. Also recall that our goal is to obtain derandomization in time $n^{1+\varepsilon} \cdot N$.

Unlike classical results, we do not prove that our PRGs are computable in time close to N on each of the 2^ℓ seeds, but rather only *"batch-computable"* in time close to $2^\ell \cdot N$ on *all seeds* at once. Indeed, such PRGs still suffice for the standard derandomization approach of enumerating over all seeds. And moreover, as mentioned after Theorem 11.1.2, the "batch-computable" PRGs that we construct also follow from the relaxed hypothesis that asserts an upper-bound on the *amortized* time-complexity of the hard problem when computing its entire truth-table (rather than a worst-case time bound); and the existence of such a problem is in fact necessary to get "batch-computable" PRGs.

The point is that the relationship that we show between the latter two objects is *quantitatively tighter* than the known relationship between standard PRGs (that are efficiently-computable on each seed) and hard problems (with bounded worst-case complexity). Thus, in the context of derandomization with almost no time overhead, it turns out to be more fruitful to study the two weaker objects. To be more explicit about this point, let us state a special case of the connection between these two objects. (The result follows from the technical versions of results that were already stated or mentioned above, but its parametrization is less clean since we wish to highlight the parametric tightness.)

Definition 11.1.5 (amortized time complexity)**.** *For $f\colon \{0,1\}^* \to \{0,1\}$, we say that $f \in \mathtt{amort}\text{-}\mathsf{DTIME}[T]$ if for every $n \in \mathbb{N}$, the truth-table of f on n-bit inputs can be printed in time $2^n \cdot T(n)$.*

Proposition 11.1.6 (near-equivalence between batch-computable PRGs and problems with bounded amortized time complexity, the polynomial setting)**.** *There exists a universal constant $c > 1$ such that the following holds. Assume there exists one-way functions that are secure against polynomial-sized cir-*

cuits. Then, for every $\varepsilon > 0$ there exist $\delta, \delta' > 0$ such that for any fixed constant $k \geq 1$:

$$\exists L \in \texttt{amort-DTIME}[2^{\delta \cdot n} \cdot T_k(n)] \setminus \textit{i.o.-DTIME}[T_k(n)]/2^{(1-\delta)\cdot n},$$

where $T_k(n) = 2^{(1-\delta)\cdot kn + c\delta \cdot n}$

$$\Downarrow \qquad \text{(Theorem 11.4.8)}$$

$\exists$ $(1/3)$-*PRG for* $\mathsf{DTIME}[O(n)]/n^{1/k}$ *with seed length* $(1+\varepsilon)\cdot(1/k)\cdot\log(n)$

that is batch-computable on all seeds in time $n^{(1+2\varepsilon)/k} \cdot n$

$$\Downarrow \qquad \text{(Theorem 11.4.11)}$$

$$\exists L \in \texttt{amort-DTIME}[2^{\delta' \cdot n} \cdot T_k(n)] \setminus \textit{i.o.-DTIME}[T_k(n)]/2^{(1-\delta')\cdot n},$$

where $T_k(n) = 2^{(1-\delta')\cdot kn}$

11.1.3 Average-case Derandomization with Almost No Overhead

Bypassing the conditional lower bound in Theorem 11.1.3, we show that a faster derandomization is possible if we are willing to settle for derandomization that *succeeds only on most inputs*, rather than in worst-case. Recall that for any $L \in \mathsf{BPTIME}[T]$ and any distribution $\mathcal{D}$ over $\{0,1\}^n$ there exists a circuit family of size $O(T(n))$ that correctly decides L_n with high probability over choice of input from $\mathcal{D}$ (say, with probability 0.99). Moreover, when we restrict the class of distributions only to those that are samplable in time $T(n)$, then there exists a *single* circuit family of size $\tilde{O}(T(n))$ that correctly decides L with high probability with respect to *all* distributions in this class.[8](Note that this class of distributions is quite natural, and in particular contains the uniform distribution.)

We show a near-matching explicit (*i.e.*, uniform) derandomization under a hypothesis that is similar to the one in Theorem 11.1.2. Specifically, under this hypothesis, for every $L \in \mathsf{BPTIME}[T]$ we construct a deterministic algorithm A_L that runs in time $n^{\varepsilon} \cdot T(n)$ and succeeds with high probability with respect to every T-time samplable distribution. (In particular, with respect to uniform distribution.) We will focus on the setting of polynomial time bounds $T(n) = n^k$, since this is the more interesting setting for improving the worst-case bound of $n^{1+\varepsilon} \cdot T(n)$ to the average-case bound $n^{\varepsilon} \cdot T(n)$.

[8]To see this, for every $n \in \mathbb{N}$, consider the first n Turing machines that run in time T. By a Chernoff bound, there exist $O(\log(n))$ random strings that lead the probabilistic algorithm to be correct with probability 0.99 on each of the n distributions sampled by the n Turing machines. The average-case error (of 0.99) can be decreased by first applying error-reduction to the original probabilistic machine.

Theorem 11.1.7 (average-case derandomization with overhead n^ε; see Theorem 11.4.13)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that for every $T(n) = n^k$ the following holds. Assume that there exist one-way functions secure against polynomial-sized circuits, and that there exists $L_0 \in \mathsf{DTIME}[2^{\delta \cdot n} \cdot T'(n)]$ such that $L_0 \notin$ i.o.-$\mathsf{DTIME}[T']/2^{(1-\delta)\cdot n+1}$, where $T'(n) = 2^{(1-\delta)\cdot(2k/\varepsilon)\cdot n+O(\delta \cdot n)}$. Then, for every $L \in \mathsf{BPTIME}[T]$ there exists a deterministic algorithm A_L that runs in time $n^\varepsilon \cdot T(n)$ such that for every T-time samplable distribution $\mathcal{D}$ it holds that $\Pr_{x \sim \mathcal{D}_n}[A_L(x) = L(x)] = 1 - n^{-\omega(1)}$.*

An appealing interpretation for the derandomization in Theorem 11.1.7 is that there exists a deterministic algorithm A_L such that every T-time algorithm that tries to find an input x such that $A_L(x) \neq L(x)$ succeeds only with negligible probability.

11.1.4 Fast Derandomization via a Simple Paradigm

The starting point for our other contributions is an alternative and *considerably simpler* proof for Theorem 11.1.1. Using a different high-level proof strategy, we rely on the hypothesis to construct a very simple PRG, and analyze it in a way that avoids essentially all of the involved technical work that was carried out in [DMOZ20]. This alternative proof leads us to two further contributions:

1. Our simple proof is flexible, and allows us to extend the original result in several directions. In particular, we can relax the hardness hypothesis while settling on derandomization with cubic or quartic overhead (see below).
2. Our proof strategy simplifies a well-known proof strategy for PRG constructions *in general*. Specifically, we show that any PRG construction that relies on "extracting randomness from a pseudoentropic string" (as in [DMOZ20], following [HILL99, BSW03, FSUV13]) can be analyzed in a simpler way.

The details of our alternative proof strategy are presented in Section 11.2.2. In a gist, given a PRG construction $G(s_0, s_1) = \mathsf{Ext}(G_0(s_0), s_1)$ that is analyzed as "extracting randomness from a pseudoentropic string", we show that G can be analyzed in the following way: We first show that Ext reduces the derandomization problem to the problem of quantified derandomization (via a non-standard reduction); and then we show that G_0 solves the latter problem, concluding that G is a PRG. This analysis follows a classical idea of Sipser [Sip88], which was recently highlighted in [GW14] and in a sequence of follow-up works concerning quantified derandomization. We also argue that this simpler analysis applies to a potentially-larger class of constructions (see Sec-

tion 11.2.2).

As mentioned above, our proof allows us to mildly relax the hypothesis of Theorem 11.1.1 while deducing an only mildly slower derandomization. Specifically, we show that the hypothesis in Theorem 11.1.1 can be relaxed to refer only to SVN circuits, which are non-uniform analogues of $\mathsf{NP} \cap \mathsf{coNP}$, while deducing derandomization with either *cubic* overhead $T(n)^{3+\varepsilon}$ or *quartic* overhead $\approx T(n)^{4+\varepsilon}$ (rather than quadratic overhead), depending on the specific hardness hypothesis. In more detail:

Theorem 11.1.8 (fast derandomization from hardness for SVN circuits)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds.*

1. *Assume that there exists $L \subseteq \{0,1\}^*$ whose entire truth-table on n-bit inputs can be printed in time $2^{(3/2)\cdot n}$, but that cannot be computed (on an input-by-input basis) by SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, for every time-constructible $T\colon \mathbb{N} \to \mathbb{N}$ we have that $\mathsf{prBPTIME}[T(n)] \subseteq \mathsf{prDTIME}[T(n)^{3+\varepsilon}]$.*
2. *Assume that there exists $L \in \mathsf{DTIME}[2^n]$ such that L cannot be computed by SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, for every time-constructible $T\colon \mathbb{N} \to \mathbb{N}$ we have that $\mathsf{prBPTIME}[T(n)] \subseteq \mathsf{prDTIME}[T(n)^{4+\varepsilon}]$.*

We also construct a near-optimal algorithm for a natural setting of quantified derandomization, under a plausible hypothesis that refers to hardness for SVN circuits. Loosely speaking, we show that randomized time-T algorithms that err on at most $2^{T(n)^{.99}}$ of their random choices can be deterministically simulated in time $T(n)^{1.01}$, if there exists a function whose truth-table on n-bit inputs can be printed in time $2^{(1.01)\cdot n}$, but that is hard to compute (on an input-by-input basis) for SVN circuits of size $2^{.99\cdot n}$. See Theorem 11.5.4 for details.

New light on bypassing the hybrid argument. A well-known challenge in proving results such as the ones in Theorem 11.1.1 and Theorem 11.1.2 is that the analysis of the PRG construction needs to avoid a certain hybrid argument (for a detailed explanation see [BSW03, Sec. 1.2], and also [FSUV13]). Indeed, the proofs in [BSW03, DMOZ20] as well as our proofs avoid such an argument. However, we believe that the observations in the current section allow to better understand *how* this barrier was bypassed in all these works, which sheds new light on this challenge. Further details appear in Section 11.2.2.

11.1.5 Organization

In Section 11.2 we describe our proofs in high-level. In Section 11.3 we present preliminary definitions and state some well-known results. In Section 11.4 we show the PRG constructions that correspond to the results in Section 11.1.2, and in Section 11.5 we show the PRG constructions that correspond to the results in Section 11.1.4. Finally, in Section 11.6 we show the conditional lower bound for derandomization that was mentioned in Section 11.1.2.

11.2 Proof Overviews

Throughout the section we will be somewhat informal with respect to the precise values of parameters, and in particular denote by $\varepsilon > 0$ an unspecified constant that should be thought of as arbitrarily small. Also, for simplicity we explain how to construct PRGs with constant error; the extensions to PRGs with error $n^{-.01}$ are straightforward.

11.2.1 Proof of Theorem 11.1.2, Theorem 11.1.4 and Theorem 11.1.7

Let us first see what is the core difficulty that yields a large polynomial overhead in existing "hardness-to-randomness" proofs. Recall that for a probabilistic algorithm M with running time $T(n)$ and a fixed input $x \in \{0,1\}^n$, we consider a distinguisher D_x that gets as input random coins $r \in \{0,1\}^N$, where $N = T(n)$, runs in time N, and outputs the decision of M at input x with randomness r. If we can construct a PRG that "fools" all potential distinguishers D_x, then we can use this PRG to derandomize M.

Classical "hardness-to-randomness" proofs (following [NW94]) rely on *reconstructive* PRGs. Such a PRG is an oracle machine G that gets a random seed and access to a function f and satisfies the following: Any efficient procedure D that distinguishes the output distribution of G^f from the uniform distribution can be transformed to an efficient procedure C_f that computes f. As a contrapositive, if f is hard for every efficient procedure, then G^f "fools" every potential efficient distinguisher D.

The key bottleneck in the reconstructive proof approach is the overhead in transforming D into C_f. In the state-of-the-art construction of Umans [Uma03] (following [NW94, IW97, STV01, TSZS06, SU05]), the transformation overhead is a large polynomial, say $|C_f| = |D|^c = N^c$. Thus, we must assume that f is hard for circuits of size N^c, which means that the time it takes to compute

G^f is larger than $N^c = T(n)^c$. This is where the large derandomization overhead comes from, and this is what we want to avoid.

The first idea: Composing two "low-cost" PRGs. We will construct a PRG with optimal seed length $(1+\varepsilon)\cdot\log(N)$ and running time $N^{1+\varepsilon}$ by composing two suboptimal yet "low-cost" PRGs; that is, each of the two PRGs falls short of achieving the parameters we need by itself, but is nevertheless computable in time $N^{1+\varepsilon}$. The first PRG will have short seed length $(1+\varepsilon)\cdot\log(N)$ but also short output length N^ε; and the second PRG will have a relatively-long seed N^ε but a sufficient output length N.

The first PRG follows from the observation that the *transformation of hardness into randomness entails very little overhead when the PRG outputs a relatively-short string*. In fact, for this purpose we can even use an instantiation of the classical construction of Nisan and Wigderson [NW94] (with variations a-la [RRV02]): Using a function f whose truth-table is of size $|f| = N^{1+O(\varepsilon)}$, a suitable instantiation of the NW PRG outputs $M = N^\varepsilon$ pseudorandom bits using seed length $(1+O(\varepsilon))\cdot\log(N)$, in time $N^{1+O(\varepsilon)}$, and with a small reconstruction overhead such that $|C_f| = |f|^\varepsilon\cdot|D| + |f|^{1-2\varepsilon} < |f|^{1-\varepsilon}$ (*i.e.*, the reconstruction procedure is an oracle machine that gets access to D, runs in time $|f|^\varepsilon$, and uses $|f|^{1-2\varepsilon}$ bits of advice; see Theorem 11.4.1).[9] Thus, if f is hard for circuits of size $|f|^{1-\varepsilon}$, then the PRG "fools" D.

The observation underlying the second PRG is that *standard cryptographic assumptions yield a very fast PRG with polynomial stretch*. Specifically, assuming the existence of a one-way function that is secure against polynomial-sized circuits, there exists a PRG $G^{\mathtt{cry}}$ with stretch $M \mapsto N$ that can be computed in time $N^{1+\varepsilon}$ and "fools" any distinguisher of size N (see Section 11.4.1). The composed PRG will take a seed $w \in \{0,1\}^{(1+O(\varepsilon))\cdot\log(N)}$, map it to $NW(w) \in \{0,1\}^M$, and output $G(w) = G^{\mathtt{cry}}(NW(w)) \in \{0,1\}^N$. The composition of $G^{\mathtt{cry}}$ and of NW indeed "fools" D, where the crucial point is that applying the "outer" PRG $G^{\mathtt{cry}}$ can be thought of as yielding a distinguisher $D' = D\circ G^{\mathtt{cry}}$ of approximately the same size as D (and therefore NW "fools" D').[10]

This argument yields derandomization with *quadratic overhead*; that is, with running time $N^{2+O(\varepsilon)} = T(n)^{2+O(\varepsilon)}$. This is because we will evaluate D, which is of size N, on each of the

[9] The parameters stated here are not fully accurate (*e.g.*, we need $M = N^{\Omega(\varepsilon^2)}$ for this to be true), but the difference is immaterial for this high-level discussion. See Section 11.4 for the full details.

[10] To see this, observe that $\Pr[D'(NW(\mathbf{u}_\ell)) = 1] = \Pr[D(G^{\mathtt{cry}}(NW(\mathbf{u}_\ell))) = 1]$, where $\ell = (1+O(\varepsilon))\cdot\log(N)$. Since D' is of approximately the same size as D, we can instantiate the NW PRG with parameters very similar to the ones above, and assuming that f is hard for circuits of size $|f|^{1-\varepsilon}$, the distribution $NW(\mathbf{u}_\ell)$ is pseudorandom for D'. Since $\Pr[D'(\mathbf{u}_M) = 1] \approx \Pr[D(\mathbf{u}_N) = 1]$, the distribution $G^{\mathtt{cry}}(NW(\mathbf{u}_\ell))$ is pseudorandom for D.

$N^{1+O(\varepsilon)}$ outputs of the PRG. Thus, so far we deduced the conclusion of Theorem 11.1.1 from incomparable hypotheses (which refer only to standard circuits), but still fall short of proving the stronger conclusion of Theorem 11.1.2.

The second idea: Using small and extremely hard truth-tables to fool distinguishers with "a little" non-uniformity. The bottleneck in the argument above is that the seed length of the NW PRG is $(1+O(\varepsilon)) \cdot \log(N)$. In general, the seed length of the NW PRG is proportional to the truth-table size of the hard function, and in our instantiation of the NW PRG the seed length is $(1+\varepsilon) \cdot \log(|f|)$. However, it a-priori seems impossible to use a smaller truth-table, since our distinguisher is a non-uniform circuit of size N, and we cannot assume hardness of a function with truth-table size $|f| < N$ for circuits of size N (*i.e.*, it is trivial to compute such a function with N bits of advice).

As mentioned in Section 11.1, the pivotal observation to solve this problem is that when the distinguisher $D = D_x$ models the execution of a probabilistic algorithm with running-time $N = T(n)$ on an input x, then D can be thought of as a time-N algorithm that uses only $|x| = n = T^{-1}(N)$ bits of non-uniform advice. In other words, to derandomize time-T algorithms it suffices to "fool" the class $\mathsf{DTIME}[N]/T^{-1}$, rather than all circuits of size N. This opens the door to instantiating the PRG using a *very hard function whose truth-table is of size significantly smaller than the running time of our distinguisher*; in other words, we will be using a function with super-exponential time complexity.

Let us spell out the resulting argument, and for concreteness let us focus on the setting of $T(n) = n^k$ for some constant k. The outer PRG $G^{\mathtt{cry}}$ will have stretch $n^\varepsilon \mapsto n^k$, and the inner PRG NW will use a function of truth-table size $|f| = n^{1+O(\varepsilon)}$ and a seed of length $(1+O(\varepsilon)) \cdot \log(n)$ to output n^ε bits. Recall that the reconstruction procedure of NW is an oracle machine that runs in time $|f|^\varepsilon$ and uses $|f|^{1-2\varepsilon}$ bits of non-uniform advice; thus, a distinguisher that runs in time $T(n)$ and uses n bits of advice yields an algorithm A_f for f that runs in time $T(n) \cdot n^{O(\varepsilon)}$ and uses $n + |f|^{1-2\varepsilon} < |f|^{1-\varepsilon}$ bits of advice (see Theorem 11.4.1). When we consider the complexity of A_f as a function of the input size $\ell = \log(|f|)$ to f, we get an algorithm that runs in time $T(2^{(1-O(\varepsilon))\cdot\ell}) \cdot 2^{O(\varepsilon\cdot\ell)}$ and uses $2^{(1-\varepsilon)\cdot\ell+1}$ bits of advice. Thus, for this to work we need to assume that $f \notin \text{i.o.-}\mathsf{DTIME}[T(2^{(1-\varepsilon)\cdot\ell}) \cdot 2^{O(\varepsilon\cdot\ell)}]/2^{(1-\varepsilon)\cdot\ell+1}$. [11]

[11] To see that this is consistent with the time bound stated in Theorem 11.1.2, plug-in $T(n) = n^k$ to the time bound to see that $T(2^{(1-\varepsilon)\cdot\ell}) \cdot 2^{O(\varepsilon\cdot\ell)} = 2^{(1-\varepsilon)\cdot k\ell + O(\varepsilon\cdot\ell)} = 2^{(1-(1-O(1/k))\cdot\varepsilon)\cdot k\ell}$.

The last necessary missing piece: Batch-computable PRGs and hard functions with bounded *amortized* time complexity. The resulting derandomization algorithm computes the output-set of the PRG, and evaluates D at each of the strings in this set. Now, recall that we want this algorithm to run in time $n^{1+O(\varepsilon)} \cdot T(n)$. It is not clear if we can compute the PRG at each seed in time close to $T(n)$ (as is the standard approach); nevertheless, we can still *compute the entire output-set of the PRG "in a batch" in time* $n^{1+O(\varepsilon)} \cdot T(n)$, which suffices for our purposes.

For concreteness, let us still focus on the setting of $T(n) = n^k$. We first compute the entire truth-table of f, and we need to do so in time at most $n^{1+O(\varepsilon)} \cdot T(n) = 2^\ell \cdot 2^{O(\varepsilon \cdot \ell)} \cdot T(2^{(1-O(\varepsilon)) \cdot \ell})$. Thus, we have to assume that when computing the entire truth-table of f, the amortized time cost per entry is no more than $2^{O(\varepsilon \cdot \ell)} \cdot T(2^{(1-O(\varepsilon)) \cdot \ell})$. (Note that the constant hidden inside the O-notation in $2^{O(\varepsilon \cdot \ell)}$ may be larger than the constant hidden inside the O-notation in $2^{O(\varepsilon \cdot \ell)}$ in our lower bound.) As mentioned in Section 11.1, the existence of such a function is necessary in order to construct a "batch-computable" PRG as the one that we are constructing (see Theorem 11.4.11). Now, given access to f, we can compute the output-set of the NW PRG in time $n^{1+O(\varepsilon)}$ (*i.e.*, we compute the combinatorial designs once in advance, and similarly apply an error-correcting code to the truth-table of f once in advance; see Section 11.7). Finally, we evaluate $D' = D \circ G^{\mathtt{cry}}$ on each of the $n^{1+O(\varepsilon)}$ resulting strings. The PRG $G^{\mathtt{cry}}$ can be assumed to run in time $T(n) \cdot n^\varepsilon$ (see Section 11.4.1), and thus our final running-time is indeed $n^{1+O(\varepsilon)} \cdot T(n)$.

The above argument proves the special case of Theorem 11.1.2 for polynomial time functions $T(n) = n^k$. (In fact, it even proves the stronger result for this special case, in which the derandomization time is $n^{1+\varepsilon} \cdot T(n)$.) In Section 11.4 we explain how to extend the argument to super-polynomial time functions $T(n) = n^{\omega(1)}$. In a gist, we will either reduce the super-polynomial case to the polynomial case using a padding argument, which yields derandomization in time $n \cdot T(n)^{1+\varepsilon}$; or rely on a stronger cryptographic hypothesis to obtain an outer PRG $G^{\mathtt{cry}}$ with super-polynomial stretch, which yields derandomization in time $n^{1+\varepsilon} \cdot T(n)$. See further details in Section 11.4.

Proof of Theorem 11.1.7: Faster average-case derandomization. For simplicity, let us prove that we can derandomize algorithms that run in time $T(n) = n^k$ in time $n^\varepsilon \cdot T(n)$ with respect to the *uniform* distribution (rather than any T-time samplable distribution).

Given a probabilistic machine M running in time T, consider the machine M' that gets input $w \in \{0,1\}^m$, maps it to $x = G^{\mathtt{cry}}(w) \in \{0,1\}^n$ where $n = m^{1/\varepsilon}$, and outputs $M(x)$. We now

instantiate Theorem 11.1.2 with the time function $T'(m) = m^{k/\varepsilon}$, which bounds the running time of M'. Using appropriate hypotheses (to apply Theorem 11.1.2 with the function T'), we deduce there exists a PRG G with seed length $(1+\varepsilon)\cdot\log(m)$ that is batch-computable in time $m^{1+\varepsilon}\cdot m^{k/\varepsilon}$ and can be used to replace the random coins of M'. Now, a key point to note is that M' only uses its random coins as random coins for M. Hence, we deduce that *the PRG G can be used to replace the random coins of M on any input $x \in \{0,1\}^n$ in the output-set of $G^{\mathtt{cry}}$*. Note that as a function of $|x| = n$, the PRG has seed length $(1+\varepsilon)\cdot\log(n^\varepsilon) < 2\varepsilon\cdot\log(n)$ and running time at most $n^{\varepsilon(1+\varepsilon)}\cdot n^k < n^{k+2\varepsilon}$.

Of course, this still doesn't mean that the G can be used to replace the random coins of M on inputs $x \in \{0,1\}^n$ that are not in the output-set of $G^{\mathtt{cry}}$. The crucial observation is that we can now *reuse the pseudorandomness properties of $G^{\mathtt{cry}}$ to "fool" a second test* (the first use of the pseudorandomness of $G^{\mathtt{cry}}$ was in our instantiation of Theorem 11.1.2). Specifically, recall that $G^{\mathtt{cry}}$ is a cryptographic PRG was obtained relying on the existence of one-way functions, and hence it can be shown to "fool" circuits of arbitrary polynomial size (see Proposition 11.4.3). Now, consider an algorithm T that gets input $x \in \{0,1\}^n$ and checks whether or not the pseudorandom coins produced by G can be used to replace the random coins of M at x.[12] Since T can be implemented by a circuit of size $\mathrm{poly}(n^k)$, it is "fooled" by $G^{\mathtt{cry}}$. And since for all x in the output-set of $G^{\mathtt{cry}}$ it holds that $T(x) = 1$, it follows that for almost all $x \in \{0,1\}^n$ it holds that $T(x) = 1$. Hence, for almost all $x \in \{0,1\}^n$ we can use G to replace the random coins of M at x.

The foregoing argument extends naturally to any T-time samplable distribution. For any T-time sampling algorithm S, we use $G^{\mathtt{cry}}$ with stretch $m \mapsto T(n)$, and define $M'(x) = M(S(G^{\mathtt{cry}}(x)))$ (this generalizes the foregoing uniform case, which is obtained using $S(z) = z$). The running time of M' is still less than $T'(m) = m^{k/\varepsilon}$, and therefore the rest of the argument proceeds without change.

11.2.2 Proofs of Theorem 11.1.1 and Theorem 11.1.8

The proof of Theorem 11.1.1 in [DMOZ20] follows a well-known strategy for constructing PRGs, which dates back to [HILL99, BSW03]. We now show how to simplify the analysis of any such construction, while pointing out that our simpler analysis also applies to a potentially-larger class

[12] To be more accurate, T solves a promise-problem wherein the "yes" instances are x such that the gap between the acceptance probability of $M(x,\cdot)$ with random coins and the acceptance probability of $M(x,\cdot)$ with pseudorandom coins is less than $1/6$, and "no" instances are those in which the said gap is at least $1/8$. This problem can be solved probabilistically in time $O(n^{k+2\varepsilon})$ (by sampling from the seeds of G and from uniform coins for M and comparing the two estimates), and the probabilistic algorithm can then be converted to a deterministic circuit of size $\mathrm{poly}(n^k)$. See the proof of Theorem 11.4.13 for details.

of constructions.

In the well-known approach introduced by [HILL99, BSW03], to construct a PRG they first construct a *pseudoentropy generator* (PEG), which takes as input a small random seed and outputs a string that appears to any efficient distinguisher as though it came from a distribution with high min-entropy; and then apply a *seeded extractor* to this pseudoentropic string. An extractor converts distributions with high *statistical* min-entropy to distributions that are *statistically* close to uniform, and the main idea is that we expect that a complexity-theoretic analogue of this statement will also hold: When the input distribution to the extractor *looks* to any efficient distinguisher as though it has high min-entropy, we intuitively expect its output distribution to *look* to any efficient distinguisher as close to uniform. This yields a PRG of the form

$$G(s_0, s_1) = \mathsf{Ext}\left(G_0(s_0), s_1\right) \ , \tag{11.2.1}$$

where G_0 is a pseudoentropy generator and Ext is an extractor.

The main challenge in materializing this approach is that constructing PEGs for standard notions of pseudoentropy (*e.g.*, HILL pseudoentropy [HILL99]) is challenging, and few constructions are known (see, *e.g.*, [HILL99, BSW03, STV01]). Doron *et al.* [DMOZ20] bypassed this obstacle by constructing a PEG for a weaker notion of pseudoentropy, called *metric pseudoentropy*;[13] however, they then faced the problem of proving that extracting from a string with high *metric* pseudoentropy yields a pseudorandom string, which required a lot of technical work. (In fact, to prove this they needed the PEG to "fool" a stronger and non-standard class of distinguishers; see [DMOZ20].)

We show that *any construction as in Eq.* (11.2.1) *is a special case of a class of constructions that can be analyzed in a different and significantly simpler way*. To do so we follow an idea of Sipser [Sip88], which was recently highlighted again in [GW14] and in a sequence of follow-up works concerning quantified derandomization (see, *e.g.*, [Tel17, Tel18, KL18, CT19, CJW20]). Specifically, consider any potential distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ of size $O(N)$. We show that G from Eq. (11.2.1) is ε-pseudorandom for D, as follows:

1. **Non-standard reduction to quantified derandomization:** Let $\bar{D}$ be a circuit that accepts its input z iff $\Pr_r[\mathsf{Ext}(z,r) \in D^{-1}(1)] \in \mu \pm \varepsilon/2$, where $\mu = \Pr_r[D(r) = 1]$. Note that $\bar{D}$ is *not*

[13] This means that for every potential distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ there *exists* some distribution $\mathbf{w}$ over $\{0,1\}^N$ with high entropy such that D does not distinguish between $\mathbf{w}$ and the output distribution of G_0^f (for a precise definition see [DMOZ20, Sec. 2]).

the circuit that is obtained by applying standard (extractor-based) error-reduction to D, but rather a circuit that tests whether or not its input causes the extractor to sample the event $D^{-1}(1)$ correctly, up to error $\varepsilon/2$. (In particular, the circuit $\bar{D}$ has the value μ hard-wired, but this does not cause a problem since $\bar{D}$ is only part of the analysis.)

The two crucial points are that $\bar{D}$ accepts all but a tiny number of exceptional inputs (since Ext samples any event correctly, with extremely high probability); and that any $(\varepsilon/2)$-PRG G_0 for $\bar{D}$ yields an ε-PRG $G(s,r) = \mathsf{Ext}(G_0(s),r)$ for D. [14]

2. **Solving the quantified derandomization problem:** Thus, the only missing part in the proof is to construct a generator G_0 that is $(\varepsilon/2)$-pseudorandom for the extremely-biased circuit $\bar{D}$ (*i.e.*, such that $\Pr_s[\bar{D}(G(s)) = 1] \geq 1 - \varepsilon/2$). This is indeed a quantified derandomization problem, where the number of exceptional inputs (of $\bar{D}$) is dictated by the parameters of the extractor Ext.

   Now, it follows from [BSW03] that metric PEGs are *equivalent* to PRGs that solve the quantified derandomization problem (*i.e.*, a metric PEG for min-entropy k is equivalent to a PRG for distinguishers with at most 2^k exceptional inputs, where in both cases this parameter corresponds to the min-entropy of Ext; see Proposition 11.3.11). Thus, any PEG G_0 is also a PRG for quantified derandomization with the parameters induced by Ext, and therefore G is indeed an ε-PRG.

The argument above shows that the composition of a metric PEG with an extractor as in Eq. (11.2.1) yields a PRG.[15] We note that this argument applies to a *potentially-larger class of constructions*, compared to the class of constructions that can be analyzed via the pseudoentropy-based approach, since the known pseudoentropy-based analysis requires the metric PEG to "fool" a stronger distinguisher class (see [DMOZ20, Sec. 6]).

Applying the proof strategy in our setting. The alternative proof of Theorem 11.1.1 and our proof of Theorem 11.1.8 follow by applying the strategy above with a very simple construction of a PRG for quantified derandomization, which is presented in Section 11.5.1. This simple construction is inspired by a technical idea from Sipser's [Sip88] original paper that introduced the approach of error-reduction and quantified derandomization,[16] but to obtain the PRG that we

[14] To see this, call a string z good if $\Pr_r\left[\mathsf{Samp}(z,r) \in D^{-1}(1)\right] \in \mu \pm \varepsilon/2$. Then, for any $\sigma \in \{0,1\}$ it holds that $\Pr_{s,r}[\mathsf{Samp}(G(s),r) \in D^{-1}(\sigma)] \leq \Pr[G(s) \text{ is not good}] + \mu + \varepsilon/2 < \varepsilon$.

[15] Since this holds for constructions that use a metric PEG, it also holds for constructions that use stronger PEGs.

[16] The original paper did not use the term quantified derandomization, which was only coined later by Goldreich and Wigderson [GW14].

need we instantiate the idea using more recent technical tools, such as locally list-decodable error-correcting codes (see Section 11.5.1 for details).

As pointed out by Dean Doron, the latter construction is identical to a simplified construction of a pseudoentropy generator for the "higher-error" setting in a recent revision of [DMOZ20]; and as pointed out by an anonymous reviewer, constructions of HSGs using essentially the same ideas date back to [ACR98, MV05].

The main difference between the proofs of Theorem 11.1.1 and Theorem 11.1.8 boils down to the construction of a circuit for the function $\bar{D}$. In more detail, note that $\bar{D}$ is a more complicated function than D, and that we need to solve the quantified derandomization problem for $\bar{D}$ rather than for D; intuitively, the overhead in implementing $\bar{D}$ as a circuit yields an overhead in our derandomization time.

The straightforward way of implmenting $\bar{D}$ yields derandomization either in cubic time or in quartic time, depending on the specific hardness hypothesis (see Theorem 11.5.5). An alternative way is to implement $\bar{D}$ using *randomness*, which mitigates the overhead and allows to obtain derandomization in quadratic time, at the cost of having to assume that the underlying hard function is hard for *randomized SVN circuits* (this yields the alternative proof of Theorem 11.1.1; see Theorem 11.5.6). For further details see Section 11.5.

New light on bypassing the hybrid argument. As mentioned in Section 11.1.4, proofs of results such as Theorem 11.1.1 and Theorem 11.1.2 need to avoid a certain hybrid argument (see, *e.g.*, [BSW03, FSUV13]). In [BSW03, Sec 1.2] it was suggested that the proof strategy of "extracting from a pseudoentropic string" allows to bypass this barrier, since the analysis of a reconstruction procedure for a PEG might be easier than the analysis of a reconstruction procedure for a PRG. (To be more accurate, [BSW03] suggested that the connection of pseudoentropy to unpredictability might be closer than the connection of pseudorandomness to unpredictability.)

Our main point is that in light of the above, the explanation can be reframed as suggesting that *the analysis of PRGs for quantified derandomization might allow avoiding a hybrid argument* more easily than the analysis of PRGs for standard derandomization.[17] This suggestion is at least as plausible as their original suggestion, since any proof that avoids a hybrid argument using the PEG-based approach also yields a proof that avoids a hybrid argument using the quantified-derandomization-based approach. Moreover, all the reconstruction procedures whose analyses

[17] This is consistent with the fact that the unconditionally-known constructions of PRGs for quantified derandomization indeed avoid a hybrid argument; see, *e.g.*, [GW14, Tel17].

avoid a hybrid argument in [BSW03, DMOZ20] and in our work can be viewed as solving a corresponding quantified derandomization problem.[18]

We also point out the fact that in all three works, the reconstruction procedures that avoid a hybrid argument used "strong" resources (*i.e.*, either used non-determinism or referred only to space-bounded computation, disregarding time). Thus, it is useful to recall that an additional potential explanation for the success so far might simply be that the analysis of such "strong" reconstruction procedures is easier.

11.3 Preliminaries

11.3.1 Complexity Classes

We first recall standard definitions of $\mathsf{prBPTIME}[T]$ and of $\mathsf{DTIME}[T]/s$. Our reason for formally stating these definitions is to clarify that we refer to problems solvable with time and randomness complexity *precisely* $T(n)$ (rather than $O(T(n))$ as in some sources).

Definition 11.3.1. *For a time-constructible function $T\colon \{0,1\}^{\mathbb{N}} \to \{0,1\}^{\mathbb{N}}$, we say a promise problem $\Pi = (Y, N)$ is in $\mathsf{prBPTIME}[T]$ if there is a randomized RAM machine M such that for every $x \in \{0,1\}^*$, the machine runs in time $T(|x|)$ and satisfies the following:*

1. *If $x \in Y$ then $\Pr[M(x) = 1] \geq 2/3$.*
2. *If $x \in N$ then $\Pr[M(x) = 0] \geq 2/3$.*

Definition 11.3.2. *We use $\mathsf{DTIME}[T]/s$ to denote the class of languages that are computable by a deterministic RAM machine that on inputs of length $n \in \mathbb{N}$ runs in time $T(n)$ and uses $s(n)$ bits of non-uniform advice.*

We also define the following notion, which generalizes $\mathsf{NP} \cap \mathsf{coNP}$ to computational models other than Turing machines (*e.g.*, to circuits, oracles machines, etc.). We then extend this notion to a general form of $\mathsf{MA} \cap \mathsf{coMA}$.

[18] In this chapter this fact is explicit, and in [DMOZ20] this is because the reconstruction procedure is part of a construction of a metric PEG, which is equivalent to a PRG for quantified derandomization. In [BSW03] there is no direct construction of a PEG or of a PRG for quantified derandomization, but the reconstruction procedures are ones that correspond to such construction: This is since the reconstruction algorithms in [BSW03] transform a *very biased distinguisher* (of a pseudoentropic distribution $\mathbf{w}$ over $\{0,1\}^n$ from the uniform distribution $\mathbf{u}_n$) into an algorithm that predicts a bit in the pseudoentropic distribution with high success probability. (Indeed, the point is that the reconstruction algorithm only works for very biased distinguishers; see [BSW03, Section 7] for further details.)

Definition 11.3.3 (nondeterministic unambiguous computation)**.** *We say that a nondeterministic procedure R* computes a function f nondeterministically and unambiguously *if for every $x \in \{0,1\}^*$ the following holds:*

1. *There exist nondeterministic choices for R such that $R(x) = f(x)$.*
2. *For all nondeterministic choices for R it holds that $R(x) \in \{f(x), \perp\}$.*

Definition 11.3.4 (SVN circuits)**.** *We say that $f\colon \{0,1\}^N \to \{0,1\}$ can be* computed by an SVN circuit of size S *if there exists a nondeterministic circuit D of size S that computes f nondeterministically and unambiguously.*

Definition 11.3.5 (randomized SVN circuits)**.** *We say that f can be* computed by randomized SVN circuits of size S *if the there exists a randomized nondeterministic circuit D of size S such that for every $x \in \{0,1\}^N$ the following holds:*

1. *There exists w such that $\Pr[D(x,w) = f(x)] \geq 2/3$.*
2. *For every w it holds that $\Pr[D(x,w) \in \{f(x), \perp\}] \geq 2/3$.*

11.3.2 Pseudorandomness, PRGs and HSGs

We recall the standard definition of pseudorandom generators and of hitting-set generators. For simplicity, when we do not specify the size of a distinguisher, we assume that this size is identical to the number of output bits of the PRG.

Definition 11.3.6 (distinguisher)**.** *For a distribution $\mathbf{w}$ over $\{0,1\}^n$, we say that $D\colon \{0,1\}^n \to \{0,1\}$* distinguishes $\mathbf{w}$ from the uniform distribution with advantage $\varepsilon > 0$ *(or that D is an ε-distinguisher for $\mathbf{w}$, in short) if $\left|\Pr_{x\in\{0,1\}^n}[D(x)=1] - \Pr[D(\mathbf{w})=1]\right| \geq \varepsilon$. If D is* not *an ε-distinguisher for $\mathbf{w}$, we say that D is* ε-fooled *by $\mathbf{w}$.*

Definition 11.3.7 (pseudorandom generator)**.** *Let G be an algorithm that gets input 1^n and a random seed of length $\ell(n)$ and outputs an n-bit string, and let $\mathcal{F}$ be a class of Boolean functions. We say that G is a* pseudorandom generator with error μ for $\mathcal{C}$ *(μ-PRG for $\mathcal{C}$, in short) if for every $f \in \mathcal{F}$ and sufficiently large $n \in \mathbb{N}$ it holds that f is μ-fooled by $G(1^n, \mathbf{u}_{\ell(n)})$.*

Definition 11.3.8 (hitting-set generator)**.** *Let H be an algorithm that gets input 1^n and a random seed of length $\ell(n)$ and outputs an n-bit string, and let $\mathcal{F}$ be a class of Boolean functions. We say that H is a* hitting-set generator for $\mathcal{C}$ with density $\mu > 0$ *(μ-HSG for $\mathcal{C}$, in short) if for every $f \in \mathcal{F}$ and sufficiently large $n \in \mathbb{N}$, either $\Pr_{x\in\{0,1\}^n}[f(x)=1] < \mu$ or there exists $s \in \{0,1\}^{\ell(n)}$ such that $f(H(s)) = 1$.*

For both PRGs and HSGs, when we omit an explicit mention of a class $\mathcal{F}$ (of potential distinguishers) we implicitly refer to the class $\mathcal{F}$ of functions that are computable by circuits of size that is identical to the number of input bits (*i.e.*, of size $S(n) = n$).

We will also refer to "batch-computable" PRGs/HSGs, in which we do not measure the computation time on a single seed, but rather the time that it takes to print the entire output-set of the PRG (on all seeds). That is:

Definition 11.3.9 (batch-computable PRGs and HSGs)**.** *Let G be a μ-PRG (resp., μ-HSG) with seed length ℓ for some class $\mathcal{C}$. We say that G is* batch-computable in time T *if for every $n \in \mathbb{N}$, the entire set of strings $\{G(1^n, s) : s \in \ell(n)\}$ can be printed in time $T(n)$.*

We also mention the notion of metric pseudoentropy, which was introduced by Barak, Shaltiel, and Wigderson [BSW03]. The reason for doing so is that we want to explicitly state and prove the claim, which was mentioned in Section 11.2.2, that PRGs for very biased circuits are *equivalent* to metric PEGs. (The proof follows [BSW03], and one direction of it was used in [DMOZ20].)

Definition 11.3.10 (metric pseudoentropy)**.** *We say that a distribution $\mathbf{w}$ over $\{0,1\}^n$ has ε-*metric-pseudoentropy at least k for circuits of size S *if for every circuit $D\colon \{0,1\}^n \to \{0,1\}$ of size S there exists a distribution $\mathbf{x}_D$ over $\{0,1\}^n$ with min-entropy at least k such that $\left|\Pr[D(\mathbf{x}_D) = 1] - \Pr[D(\mathbf{w}) = 1]\right| < \varepsilon$.*

Proposition 11.3.11 (quantified derandomization is equivalent to metric pseudoentropy)**.** *For any distribution $\mathbf{w}$ over $\{0,1\}^n$ and every $k \leq n$ and $\varepsilon > 0$ the following holds:*

1. *If $\mathbf{w}$ has ε-metric-pseudoentropy at least k for circuits of size S, then for every $\delta > 0$ there does not exist a $(\delta \cdot 2^k)$-quantified $(\varepsilon + \delta)$-distinguisher for $\mathbf{w}$.*
2. *If there does not exist a 2^k-quantified ε-distinguisher for $\mathbf{w}$, then $\mathbf{w}$ has ε-metric-pseudoentropy at least k for circuits of size S.*

Proof. Barak, Shaltiel and Wigderson [BSW03] proved that $\mathbf{w}$ has ε-metric-pseudoentropy at least k for circuits of size S if and only if for every circuit $D\colon \{0,1\}^n \to \{0,1\}$ of size S and every $\sigma \in \{0,1\}$ it holds that $\Pr[D(\mathbf{w}) = \sigma] \leq \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \varepsilon$.

Now, assume that $\mathbf{w}$ has ε-metric-pseudoentropy at least k for circuits of size S, and let $D\colon \{0,1\}^n \to \{0,1\}$ be a size-S circuit that evaluates to some $\sigma \in \{0,1\}$ on at most $\delta \cdot 2^k$ of its inputs. Using the result of [BSW03], it follows that $\Pr[D(\mathbf{w}) = \sigma] \leq \delta + \varepsilon$. For the other direction, assume that there does not exist a 2^k-quantified ε-distinguisher of size S for $\mathbf{w}$, and let $D\colon \{0,1\}^n \to \{0,1\}$ be a size-S circuit. We claim that for every $\sigma \in \{0,1\}$ it holds that $\Pr[D(\mathbf{w}) = \sigma] \leq \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \varepsilon$.

To see this, note that if $|D^{-1}(\sigma)| > 2^k$ then the bound is trivial; and otherwise, it cannot be that $\Pr[D(\mathbf{w}) = \sigma] > \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \varepsilon \geq \varepsilon$, since that would mean that D is a 2^k-quantified ε-distinguisher of size S for $\mathbf{w}$. ■

11.3.3 Well-known Algorithmic Constructions

In this section we will state several well-known algorithmic results that we will use in our proofs: Namely, constructions of a hash function, of a randomness extractor, and of an error-correcting code. First, we will need the following construction of a pairwise-independent hash function that is computable in quasilinear time:

Theorem 11.3.12 (quasilinear-time pairwise-independent hashing). *For every $m, m' \in \mathbb{N}$ there exists a family $\mathcal{H} \subseteq \left\{\{0,1\}^m \to \{0,1\}^{m'}\right\}$ of quasilinear-sized circuits such that for every distinct $x, x' \in \{0,1\}^m$ it holds that $\Pr_{h\in\mathcal{H}}[h(x) = h(x')] \leq 2^{-m'}$.*

Proof. We use convolution hashing (see [MNT93]): Any $h = h_{a,b} \in \mathcal{H}$ is uniquely defined by a pair $a \in \{0,1\}^{m+m'-1}$ and $b \in \{0,1\}^{m'}$ such that $h_{a,b}(x) = a * x + b$, where the i^{th} output bit of the convolution operator is $(a * x)_i = \sum_{j\in[m]} a_{i+j-1} \cdot x_i \pmod 2$. Using the Fast Fourier Transform, the complexity of computing $h_{a,b}$ (for a fixed a, b) is $\tilde{O}(n)$ (see [CLRS09, Thm 30.8]). ■

In this chapter we will need an averaging sampler, or equivalently a seeded randomness extractor. As noted in [DMOZ20], a construction of an such an extractor in [TSZS06, Thm 5] can be analyzed with sub-constant min-entropy rates, and this is indeed what we will need in the this chapter. We first define the corresponding notions, then state the well-known equivalence between extractors and averaging samplers, and finally state the result from [DMOZ20], following [TSZS06].

Definition 11.3.13 (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 $\Pr[\mathbf{x} = x] \leq 2^{-k}$.*

Definition 11.3.14 (seeded extractor). *A function $\mathsf{Ext}\colon [N] \times \{0,1\}^\ell \to [M]$ is a (k,δ)-extractor if for every random variable $\mathbf{x}$ over $[N]$ with min-entropy k it holds that $\mathsf{Ext}(\mathbf{x}, \mathbf{u}_\ell)$ is δ-close in statistical distance to $\mathbf{u}_m$. The value ℓ is the* seed length *of the extractor.*

Definition 11.3.15 (averaging samplers). *A function $f : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is an* averaging sampler with accuracy $\varepsilon > 0$ 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 11.3.16 (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* $(n - \log(\varepsilon/\delta), 2\varepsilon)$*-extractor.*

Theorem 11.3.17 (an extractor with near-logarithmic seed length; see [DMOZ20, Lem 7.10], following [TSZS06, Thm 5])**.** *There exists a constant* $c \geq 1$ *such that for every* $\gamma < 1/2$ *the following holds. There exists a strong* (k,ε)*-extractor* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *for* $k = n^{1-\gamma}$ *and* $\varepsilon \geq c \cdot n^{-1/2+\gamma}$ *and* $d \leq (1 + c\cdot\gamma)\cdot\log(n) + c\cdot\log(1/\varepsilon)$ *and* $m = \frac{1}{c}\cdot n^{1-2\gamma}$*. Moreover, the extractor is computable in linear time.*

Finally, we will need a locally list-decodable code, and we will use the Reed-Muller code. We first define this notion then state the result of [STV01], which asserts that the RM code is indeed locally list-decodable.

Definition 11.3.18 (locally list-decodable code)**.** *We say that* $Enc\colon \Sigma^N \to \Sigma^M$ *is* locally list-decodable from agreement ρ with decoding circuit size s and output-list size L *if there exists a randomized oracle circuit* $Dec\colon [N]\times[L] \to \Sigma$ *of size* s *such that following holds. For every* $z \in \Sigma^M$ *that satisfies* $\Pr_{i\in[M]}[z_i = Enc(x)_i]$ *for some* $x \in \Sigma^N$ *there exists* $a \in [L]$ *such that for every* $i \in [N]$ *we have that* $\Pr[Dec^z(i,a) = x_i] \geq 2/3$*, where the probability is over the internal randomness of* Dec*.*

Theorem 11.3.19 (the Reed-Muller code is locally list-decodable; see [STV01, Thm. 29])**.** *Let* $RM\colon \mathbb{F}_q^{\binom{t+d}{d}} \to \mathbb{F}_q^{q^t}$ *be the* t*-variate Reed-Muller code of degree* d *over* $\mathbb{F}_q$*. Then, for every* $\rho \geq \sqrt{8\cdot(d/q)}$ *it holds that* RM *is locally list-decodable from agreement* ρ *with decoding circuit size* $\mathrm{poly}(t, \log(q), d, 1/\rho)$ *and output-list size* $O(1/\rho)$*.*

Corollary 11.3.20 (a locally list-decodable code)**.** *For every constant* $\eta > 0$ *there exists a constant* $\eta' > 0$ *such that the following holds. For every* $m \in \mathbb{N}$ *and* $\rho = \rho(m)$ *there exists a code* $Enc\colon \{0,1\}^m \to \Sigma^{\bar{m}}$*, where* $|\Sigma| = O(m^{\eta'}/\rho^2)$ *and* $\bar{m} = O_{\eta'}\left(m/\rho^{2/\eta'}\right)$*, such that:*

1. *The code is computable in time* $\tilde{O}(\bar{m}\cdot\log(|\Sigma|)) = \tilde{O}(m/\rho^{2/\eta'})$*.*
2. *The code is locally list-decodable from agreement* ρ *with decoding circuit size* $m^{\eta}\cdot(1/\rho)^{1/\eta'}$ *and output list size* $O(1/\rho)$*.*

Proof. For $\eta' < \eta$ be sufficiently small, let $t = 1/\eta'$, let $d = m^{\eta'}$, and let $q = (8/\rho^2) \cdot d$. We use the t-variate Reed-Muller code of degree d over $\mathbb{F}_q$, whose input (of bits) is of length $\binom{t+d}{d} \cdot \log(q) \geq (d/t)^t \cdot \log(q) \geq m$ and output (of elements in $\mathbb{F}_q$) is of length $\bar{m} = q^t = O_{\eta'}\left(m/\rho^{2/\eta'}\right)$. By Theorem 11.3.19, this code is locally list-decodable from agreement ρ with decoding circuit size $\text{poly}(t, \log(q), d, 1/\eta) = \text{poly}(m^{\eta'}/\rho) \leq m^{\eta} \cdot (1/\rho)^{1/\eta'}$ (choosing η' to be sufficiently small) and output list size $O(1/\rho)$. ■

11.4 Derandomization with Almost No Slowdown

In this section we prove the results that were presented in Section 11.1.2, namely Theorem 11.1.2, Theorem 11.1.4, and Theorem 11.1.7, and Proposition 11.1.6. The sole exception is that Theorem 11.1.3 is proved separately in Section 11.6.

As mentioned in Section 11.2, one of our basic ideas will be to compose two PRGs that are computable in near-linear time but have certain (respective) shortcomings, in order to obtain one PRG that does not suffer from these shortcomings. In Section 11.4.1 we show that the two foregoing PRGs exist, one of them unconditionally and the other under the hypothesis that non-uniformly secure one-way functions exist. Then, in Section 11.4.2 we show how to compose these PRGs in order to obtain a single (and better) PRG. Next, in Section 11.4.3 we prove Theorem 11.1.2 as well as a converse direction, which asserts that if PRGs with our parameters exist then there exists a hard problem as in the hypothesis of Theorem 11.1.2 (*i.e.*, we prove Proposition 11.1.6). And finally, in Section 11.4.4 we prove several extensions and optimizations of Theorem 11.1.2, namely Theorem 11.1.4 and Theorem 11.1.7.

Throughout the section we will ignore rounding issues for simplicity, since rounding issues do not significantly affect our proofs. Nevertheless, to avoid confusion, let us state in advance that the notation $\mathsf{DTIME}[n]/T^{-1}$ will denote the class of functions computable in linear time with $\lceil T^{-1}(n) \rceil$ bits of advice (*i.e.*, the advice length is the minimal m such that $T(m) \geq n$).

11.4.1 Near-linear-time Computable PRGs

As described in Section 11.2, we will compose a PRG that has a small seed $(1.01) \cdot \log(n)$ but short output length n^{ε}, with a PRG that has a relatively-long seed n^{ε} and a long output length n. The first of the two PRGs can be constructed using an instantiation of the classic Nisan-Wigderson PRG [NW94], with modifications a-la [RRV02]; this instantiation yields the following result:

Theorem 11.4.1 (NW Generator for small output length)**.** *There exists a universal constant c_{nw} such that for all sufficiently small ε_{nw}, there exists an oracle machine G satisfies the following:*

- *When given input $1^{N^{\varepsilon_{nw}}}$ and oracle access to a function $f\colon \{0,1\}^{\log(N)} \to \{0,1\}$, the machine G runs in time $N^{1+c_{nw}\cdot\sqrt{\varepsilon_{nw}}}$ and outputs $2^{\ell_{nw}(N)}$ strings in $\{0,1\}^{N^{\varepsilon_{nw}}}$, where $\ell_{nw}(N) = (1 + c_{nw}\sqrt{\varepsilon_{nw}}) \cdot \log N$.* [19]
- *There exists an oracle machine R that, when given input $x \in \{0,1\}^{\log N}$ and oracle access to an $(N^{-\varepsilon_{nw}})$-distinguisher for $G(1^N, \mathbf{u}_{\ell(N)})^f$ and $N^{1-\sqrt{\varepsilon_{nw}}/c_{nw}}$ bits of advice, runs in time $N^{c_{nw}\cdot\sqrt{\varepsilon_{nw}}}$ and outputs $f(x)$.*

The proof of Theorem 11.4.1 is essentially a different parametrization of well-known proofs, and we provide a proof sketch in Section 11.7 for completeness. The second of the two PRGs that we need is a near-linear time computable PRG with polynomial stretch that "fools" linear-sized circuits. Such a PRG is not known to follow from standard hardness assumptions for non-uniform circuits, where the main challenge is obtaining a near-linear runtime in the output length (cf., the PRG of [BFNW93] has polynomial stretch but runs in sub-exponential time).

Assumption 11.4.2 (near-linear-time computable PRG with arbitrary polynomial stretch)**.** *For every $\varepsilon > 0$, there exists a $(1/n)$-PRG with seed length $\ell(n) = n^{\varepsilon}$ that is computable in time $n^{1+\varepsilon}$.*

We show that a PRG as in Assumption 11.4.2 exists, under the hypothesis that there exist one-way functions secure against polynomial-sized circuits. The proof of this claim amounts to using the classic construction of PRGs from one-way functions (OWFs) [HILL99], and then applying standard techniques to extend the expansion factor of PRGs (see, *e.g.*, [Gol01, Const. 3.3.2]). That is:

Proposition 11.4.3 (OWF $\Rightarrow$ near-linear-time PRG)**.** *If there exists a polynomial-time computable one way function that is secure against circuits of arbitrary polynomial size, then Assumption 11.4.2 is true. Moreover, for some negligible function* neg *and any polynomial p, the resulting PRG is* neg*-pseudorandom for circuits of size $p(n)$ (rather than only $1/n$-pseudorandom for circuits of linear size).*

Proof Sketch. By [HILL99], our hypothesis implies that for some negligible function neg there exists an s-PRG for P/poly with seed length $\ell(n) = n/2$ that is computable in polynomial time. In more detail, for some constant $c \in \mathbb{N}$ there exists G_1 that extends an m-bit seed to a $2m$-bit

[19] For simplicity, we bounded both the seed length and the running time using the same parameter (*i.e.*, $c_{nw} \cdot \sqrt{\varepsilon_{nw}}$) such that the running time is precisely $2^{\ell_{nw}(N)}$. In the actual construction the seed length is smaller than $1 + c_{nw} \cdot \sqrt{\varepsilon_{nw}}$ (see Section 11.7 for details).

output such that G_1 is computable in time m^c, and for all $k \in \mathbb{N}$, no m^k-size circuit can distinguish between $G_1(U_m)$ and U_{2m} with advantage at least $\mathtt{neg}(m)$.

Now, let $\varepsilon > 0$. Our PRG G gets input 1^n and a seed x of length $m = n^{\varepsilon/2c} < n^\varepsilon$ and acts as follows:

- Let $\sigma_1 = x$.
- For all $i \in \{2, 3, \ldots, n/m\}$, we set σ_i to be the last m bits of $G_1(\sigma_{i-1})$.
- The output of $G(x)$ is the concatenation of all $\sigma_1, \sigma_2, \ldots, \sigma_{n/m}$.

Note that G outputs n bits, and its running time is at most $m^c \cdot n + O(n) \leq n^{1+\varepsilon}$. A standard hybrid argument (as in [Gol01, Thm 3.3.3]) shows that for every polynomial p it holds that G is a $(1/p(n))$-PRG for circuits of size $p(n)$. ■

When extending Theorem 11.1.2 to super-polynomial time functions $T(n) = n^{\omega(1)}$ we will use a hypothesis stronger than Assumption 11.4.2; specifically, we will assume that there exists a PRG with sub-exponential (rather than polynomial) stretch that is computable in near-linear time. Analogously to Proposition 11.4.3, such a PRG follows from the existence of a OWF that is secure against circuits of sub-exponential size.

Assumption 11.4.4 (near-linear-time computable PRG with sub-exponential stretch)**.** *For some constant c there exists a $(1/n)$-PRG with seed length $\ell(n) = (\log(n))^c$ that is computable in time $n \cdot (\log(n))^c$.*

Proposition 11.4.5 (OWF against sub-exponential circuits $\Rightarrow$ near-linear-time PRG with sub-exponential stretch)**.** *If there exists a polynomial-time computable one-way function that is secure against circuits of size 2^{n^ε} (for some constant $\varepsilon > 0$), then Assumption 11.4.4 is true.*

We omit the proof of Proposition 11.4.5, since it is nearly identical to the proof of Proposition 11.4.3.

11.4.2 Composing Two Near-linear-time PRGs

We now show that the two PRGs that were mentioned in Section 11.4.1 can be composed to obtain a single PRG that has both a short seed length and small running time. Moreover, we will instantiate this PRG with parameters that are suitable for our application, which involves very small truth-tables that are hard for algorithms with super-exponential time complexity. Specifically, in the following result, our goal will be to "fool" the class $\mathsf{DTIME}[O(n)]/A$, and we will do so using a PRG with seed length $1.01 \cdot \log(A(n))$ and running time $A(n)^{1.01} \cdot n$; that is:

Proposition 11.4.6 (super-exponential hardness to near-optimal randomness by composing two "low-cost" PRGs)**.** *There exists a universal constant c such that for every $\varepsilon > 0$ there exists $\delta_0 > 0$ and $\delta_2 > \delta_1 > 0$ for which the following holds. For any time-constructible non-increasing function $A\colon \mathbb{N} \to \mathbb{N}$ such that $A(\mathbb{N}) = \mathbb{N}$ and $A^{-1}(n) = \min\{N \in \mathbb{N} : A(N) = n\}$ is time-constructible, assume that:*

1. *There exists a $(1/n)$-PRG with seed length $A(n)^{\delta_0}$ and running time $n \cdot A(n)^{\delta_0}$.*
2. *For $T_{A^{-1}}(n) = 2^{(c\cdot\delta_1)\cdot n} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot n}\right)$, there exists $L \in \texttt{amort-DTIME}[2^{\delta_2 \cdot n} \cdot T_{A^{-1}}(n)]$ such that $L \notin$ i.o.-$\mathsf{DTIME}\left[T_{A^{-1}}\right]/2^{(1-\delta_1)\cdot n+1}$.* [20]

Then, there exists an $(n^{-\delta_0/2})$-PRG for $\mathsf{DTIME}[O(n)]/A$ that on input 1^n uses a seed of length $(1+\varepsilon)\cdot\log(A(n))$ and is batch-computable in time $A(n)^{1+\varepsilon}\cdot n$.

Proof. Let c_{nw} be the constant from Theorem 11.4.1. Let $\delta_1, \delta_0 > 0$ be sufficiently small constants to be specified later, and let c be a universal constant to be specified later. We first describe the construction of the PRG G.

Construction of the PRG G. It will be more convenient to denote the input to the PRG by 1^N rather than 1^n. For $N \in \mathbb{N}$, let $n = A(N)$ and let $\ell = \frac{\log(n)}{1-\delta_1}$ (such that $2^\ell = n^{1/(1-\delta_1)}$). By our hypothesis, the truth-table of L_ℓ (*i.e.*, the restriction of the hypothesized L to ℓ-bit inputs) can be printed in time $2^{(1+c\cdot\delta_1+\delta_2)\cdot\ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right) \leq 2^{(1+c\cdot\delta_1+\delta_2)\ell} \cdot N$, whereas $L \notin$ i.o.-$\mathsf{DTIME}[2^{(c\cdot\delta_1)\cdot\ell} \cdot A^{-1}(n)]/2^{\ell(1-\delta_1)+1}$.

Let G_{cry} be the $(1/n)$-PRG from our first hypothesis, and let G_{nw} be the oracle machine from Theorem 11.4.1, instantiated with parameter $\varepsilon_{nw} = (1-\delta_1)\cdot\delta_0$. Note that for $N_{nw} = |L_\ell| = n^{1/(1-\delta_1)}$, when G_{nw} gets input $1^{N_{nw}^{\varepsilon_{nw}}}$ and oracle access to L_ℓ, it uses a seed of length $\ell_{nw} = (1 + c_{nw}\cdot\sqrt{\varepsilon_{nw}})\cdot\ell = \frac{1+c_{nw}\cdot\sqrt{\varepsilon_{nw}}}{1-\delta_1}\cdot\log n$, and outputs a string of length $N_{nw}^{\varepsilon_{nw}} = 2^{\varepsilon_{nw}\cdot\ell} = n^{1/(1-\delta_1)\cdot\varepsilon_{nw}} = n^{\delta_0}$.

On input 1^N and given a seed $w \in \{0,1\}^{\ell_{nw}}$, the machine G outputs

$$G(1^N, w) = G_{cry}(1^N, G_{nw}(1^{n^{\delta_0}}, w))\ ;$$

that is, G outputs the composition of G_{nw} and G_{cry}.

Analysis. Consider any $F \in \mathsf{DTIME}[O(N)]/A$ (to reflect the relationship with the notation above more clearly, we denote the input length to the time function by N rather than n). Assume towards a contradiction that there are infinitely many $N \in \mathbb{N}$ such that $G(1^N, \mathbf{u}_{(1+\varepsilon)\cdot\log(A(N))})$ does not

[20] Recall that the definition of $\texttt{amort-DTIME}[T]$ appears in Definition 11.1.5.

$(N^{-\delta_0/2})$-fool F on inputs of length N. [21]

Let D be an algorithm that gets an input $w \in \{0,1\}^m$, computes $n = m^{1/\delta_0}$ and $N = A^{-1}(n)$, and outputs $F(G_{cry}(1^N, w))$. Note that D can be computed on inputs of length n^{δ_0} in time $O(N \cdot n^{\delta_0})$ and with n bits of advice. Now, recall that G_{cry} is $(1/N)$-pseudorandom for F, and therefore

$$\left| \mathsf{E}_{x \in \{0,1\}^N}[F(x)] - \mathsf{E}_{w \in \{0,1\}^{n^{\delta_0}}}[D(w)] \right| \leq 1/N \,. \tag{11.1}$$

It follows that for any distribution $\mathbf{w}$ over $\{0,1\}^{n^{\delta_0}}$, if F on inputs of length N is an $(N^{-\delta_0/2})$-distinguisher for $G_{cry}(1^N, \mathbf{w})$, then D on inputs of length n^{δ_0} is a $(N^{-\delta_0/2} - 1/N)$-distinguisher for $\mathbf{w}$. (This is since $\mathsf{E}_{w \in \{0,1\}^{n^{\delta_0}}}[D(w)]$ is $1/N$-close to $\mathsf{E}_{x \in \{0,1\}^N}[F(x)]$, but the latter is $(N^{-\delta_0/2})$-far from $G_{cry}(1^N, \mathbf{w})$.) In particular, there are infinitely many $n = A(N)$ such that D on inputs of length n^{δ_0} is a $(N^{-\varepsilon_{nw}})$-distinguisher for $G_{nw}^{L_\ell}(1^{n^{\delta_0}}, \mathbf{u}_{\ell_{nw}})$.

Fix an n as above, and recall the notation $\ell = \frac{\log(n)}{1-\delta_1}$. By Theorem 11.4.1, there is an oracle machine R that computes L_ℓ when given oracle access to $D_{n^{\delta_0}}$. Plugging the time and advice complexity of D_n, we obtain an algorithm that decides L_ℓ in time

$$O\left(N_{nw}^{c_{nw}\sqrt{\varepsilon_{nw}}} \cdot n^{\delta_0} \cdot N\right) \leq 2^{(\varepsilon_{nw} + c_{nw}\sqrt{\varepsilon_{nw}}) \cdot \ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right)$$

when given an advice string of length

$$N_{nw}^{1-\sqrt{\varepsilon_{nw}}/c_{nw}} + n = 2^{(1-\sqrt{\varepsilon_{nw}}/c_{nw})\cdot\ell} + 2^{(1-\delta_1)\cdot\ell} \,.$$

Setting the parameters. The above shows that any distinguisher F can be converted to an algorithm that decides L infinitely-often; we now set the parameters to obtain a contradiction. We set $\delta_0 = (c_{nw} \cdot \delta_1)^2/(1-\delta_1)$, which implies that $\sqrt{\varepsilon_{nw}}/c_{nw} = \delta_1$ (since we defined $\varepsilon_{nw} = (1-\delta_1)\cdot\delta_0$). The number of advice bits is then bounded by $2^{(1-\delta_1)\cdot\ell+1}$, and the running time can be bounded by

$$2^{(\delta_1^2 c_{nw}^2 + \delta_1 c_{nw}^2)\cdot\ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right) \leq 2^{(2\delta_1 c_{nw}^2)\cdot\ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right) \,,$$

which contradicts our assumption about L if we set the universal constant c in the statement to be $2c_{nw}^2$. Therefore, G is an $(N^{-\delta_0/2})$-PRG for $\mathsf{DTIME}[O(N)]/A$.

[21] Recall that the seed length of G for output length N is $\frac{1+c_{nw}\cdot\sqrt{\varepsilon_{nw}}}{1-\delta_1} \cdot \log(A(N))$. We will set the parameters below such that this equals $(1+\varepsilon)\cdot\log(A(N))$.

The seed length of G on input 1^N is $(1+c_{nw}\cdot\sqrt{\varepsilon_{nw}})/(1-\delta_1)\cdot\log(A(N)) = (1+\delta_1\cdot c_{nw}^2)/(1-\delta_1)\cdot\log(A(N))$, which is smaller than $(1+\varepsilon)\cdot\log(A(N))$ if we set $\delta_1 > 0$ to be sufficiently small. The batch-computation time of G on input 1^N is bounded by the time it takes to compute the truth-table of L_ℓ, the time it takes to batch-compute G_{nw}, and the time it takes to apply G_{cry} to each output of G_{nw}; this is at most

$$\begin{aligned}
&O\left(2^{(1+c\cdot\delta_1+\delta_2)\ell}\cdot A^{-1}\left(2^{\ell(1-\delta_1)}\right) + 2^{(1+c_{nw}\cdot\sqrt{\varepsilon_{nw}})\cdot\ell}\cdot N\cdot n^{\delta_0}\right)\\
&\quad\leq N\cdot O\left(n^{(1+c\cdot\delta_1+\delta_2)/(1-\delta_1)} + n^{(1+c_{nw}\cdot\sqrt{\varepsilon_{nw}})/(1-\delta_1)+\delta_0}\right)\\
&\quad= N\cdot O\left(n^{(1+c\cdot\delta_1+\delta_2)/(1-\delta_1)} + n^{(1+\delta_1\cdot c_{nw}^2+(c_{nw}\cdot\delta_1)^2)/(1-\delta_1)}\right)\ ,
\end{aligned}$$

which is at most $N\cdot n^{1+\varepsilon} = N\cdot A(N)^{1+\varepsilon}$ if δ_1 and δ_2 are sufficiently small. ■

11.4.3 Proof of Theorem 11.1.2 And of a Converse Direction

As explained in the introduction, our first observation towards proving Theorem 11.1.2 is that in order to derandomize $\mathsf{prBPTIME}[T]$, we do not actually need to "fool" non-uniform circuits, but only to "fool" the class $\mathsf{DTIME}[O(n)]/T^{-1}$. [22] Let us formally prove this statement:

Proposition 11.4.7 (PRGs for "DTIME with bounded advice" suffice for derandomization)**.** *For any time-constructible and increasing $T\colon\mathbb{N}\to\mathbb{N}$, if there exists a $(1/8)$-PRG for $\mathsf{DTIME}[O(n)]/T^{-1}$ with seed length $\ell(n)$ that is batch-computable in time $W(n)$, then*

$$\mathsf{prBPTIME}[T]\subseteq\mathsf{prDTIME}\left[O\left(W(T(n))+2^{\ell(T(n))}\cdot T(n)\right)\right]\ .$$

Proof. Let $\Pi=(Y,N)\in\mathsf{prBPTIME}[T]$ and let M be a randomized time-T algorithm that solves Π. Given input $x\in\{0,1\}^n$, we invoke the PRG in our hypothesis, denoted G, on input $1^{T(n)}$ and all possible seeds to generate $2^{\ell(T(n))}$ outputs $\{G(1^{T(n)},w)\}_{w\in\{0,1\}^{\ell(T(n))}}$, and accept x if and only if

$$\Pr_{w\in\{0,1\}^{\ell(T(n))}}\left[M_x(G(1^{T(n)},w))=1\right]>1/2\ ,$$

where $M_x\colon\{0,1\}^{T(|x|)}\to\{0,1\}$ accepts its input $r\in\{0,1\}^{T(|x|)}$ if and only if M accepts x when using randomness r.

[22] Recall that we ignore rounding issues for simplicity, and that $\mathsf{DTIME}[O(n)]/T^{-1}$ is the class of linear-time algorithms whose advice length is $\lceil T^{-1}(n)\rceil$.

The foregoing algorithm runs in time $O\Big(W(T(n)) + 2^{\ell(T(n))} \cdot T(n)\Big)$. To show its correctness, assume towards a contradiction that it does not solve Π. Then, there are infinitely many $m \in \mathbb{N}$ (we call them bad m's) such that there exists $x_m \in \{0,1\}^m$ for which M_{x_m} is an $(1/8)$-distinguisher for $G(1^{T(n)}, \mathbf{u}_{\ell(T(n))})$.

Consider the following algorithm: Given $x \in \{0,1\}^n$, if $n \notin \{T(m) : m \in \mathbb{N}\}$, accept; otherwise, when $n = T(m)$ for $m \in \mathbb{N}$, we denote the m-bit advice string by $a_m \in \{0,1\}^m$, and output $M_{a_m}(x)$. Note that this algorithm yields a function $D \in \mathsf{DTIME}[O(n)]/T^{-1}$, and that when given advice $a_m = x_m$ for all bad $m \in \mathbb{N}$ it holds that D on inputs of length $T(m)$ computes M_{x_m}, and is therefore a $(1/8)$-distinguisher for $G(1^{T(m)}, \mathbf{u}_{\ell(T(n))})$. This contradicts our hypothesis about G. ■

Note that in the foregoing proof we only relied on the fact that the PRG "fools" $\mathsf{DTIME}[O(n)]/T^{-1}$ on inputs of length $n = T(m)$ for some $m \in \mathbb{N}$, rather than on all input lengths. For simplicity we did not explicitly state this relaxed hypothesis.

We now prove Theorem 11.1.2. We actually prove the result in two parts, each of which refers to a different parameter setting and asserts a stronger result. The first part refers to probabilistic polynomial-time algorithms, and for this setting we show derandomization in time $n^{1+\varepsilon} \cdot T(n)$, which is better than the time bound of $n \cdot T(n)^{1+\varepsilon}$ stated in Theorem 11.1.2. [23] In more detail:

Theorem 11.4.8 (derandomization with almost no overhead for polynomial-time algorithms)**.** *There exists a universal constant $c > 1$ such that for every $\varepsilon > 0$ there exist $\rho > 0$ and $\delta_2 > \delta_1 > 0$ for which the following holds. For any constant $k \geq 1$, assume that there exist one-way functions that are secure against polynomial-sized circuits, and that for $T_k(n) = 2^{(1-\delta_1)\cdot kn + (c\cdot\delta_1)\cdot n}$ there exists $L \in \mathtt{amort}\text{-}\mathsf{DTIME}[2^{\delta_2 \cdot n} \cdot T_k(n)]$ such that $L \notin$ i.o.-$\mathsf{DTIME}[T_k]/2^{(1-\delta_1)\cdot n+1}$. Then, there exists an $(n^{-\rho})$-PRG for $\mathsf{DTIME}[O(n)]/n^{1/k}$ with seed length $(1+\varepsilon)\cdot(1/k)\cdot\log(n)$ that is batch-computable in time $n^{1+1/k+\varepsilon/k}$. Consequently, we have that $\mathsf{prBPTIME}[n^k] \subseteq \mathsf{prDTIME}[n^{k+1+\varepsilon}]$.*

Proof. We will instantiate Proposition 11.4.6 with parameter value $\varepsilon' = \varepsilon/3$, and denote by $\delta'_0, \delta'_1, \delta'_2$ be the three constants from Proposition 11.4.6 corresponding to ε'. We set $\delta_0 = \delta'_0$, and $\delta_1 = \delta'_1$, and $\delta_2 = \delta'_2/2$. Relying on Proposition 11.4.3 and on our hypothesis, there exists a $(1/n)$-PRG that on input 1^n uses a seed of length $n^{\delta_0/k}$ and is computable in time $n^{1+\delta_0/k}$.

[23] Indeed, the difference between the two can be bridged by taking a sufficiently small $\varepsilon > 0$ that depends on T (*i.e.*, if $T(n) = n^k$ we set $\varepsilon = \varepsilon'/k$), but in this case the hardness hypothesis becomes less natural and does not match Theorem 11.4.11 (*i.e.*, the hypothesized hardness will be $2^{(1-\delta)\cdot kn}$ for $\delta \ll 1/k$).

Let $A(N) = \lceil N^{1/k} \rceil$, and let $A^{-1}(n) = (n-1)^k + 1 = \min\{N \in \mathbb{N} : A(N) = n\}$. Let f be the hard function from our hypothesis, and note that for every $\ell \in \mathbb{N}$, the truth-table of L_ℓ can be printed in time

$$\begin{aligned} 2^\ell \cdot 2^{(1-\delta_1)\cdot k\ell + (c\cdot\delta_1+\delta_2)\cdot\ell} &< 2^{(1+c\delta_1+\delta_2)\cdot\ell} \cdot 2 \cdot A^{-1}(2^{(1-\delta_1)\cdot k\ell}) \\ &< 2^{(1+c\delta_1+2\delta_2)\cdot\ell} \cdot A^{-1}(2^{(1-\delta_1)\cdot k\ell}) \,, \end{aligned}$$

whereas L_ℓ cannot be computed in time $2^{(c\cdot\delta_1)\cdot\ell} \cdot 2^{(1-\delta_1)\cdot k\ell} > 2^{(c\cdot\delta_1)\cdot\ell} \cdot A^{-1}(2^{(1-\delta_1)\cdot k\ell})$ when given at most $2^{(1-\delta_1)\cdot\ell+1}$ bits of advice. Thus, relying on Proposition 11.4.6, we can invoke the proposition with the parameter value ε'.

We deduce that there exists an $(n^{-\delta_0/2})$-PRG for $\mathsf{DTIME}[O(n)]/A$ with seed length $\ell(n) = (1+\varepsilon')\cdot\log(A(n)) < (1+2\varepsilon')\cdot(1/k)\cdot\log(n)$ that is batch-computable in time $W(n) = A(n)^{1+\varepsilon'} \cdot n < n^{1+1/k+2\varepsilon'/k}$. Using Proposition 11.4.7 with $T(n) = n^k$, we deduce that $\mathsf{prBPTIME}[n^k] \subseteq \mathsf{prDTIME}[O(n^{k+1+2\varepsilon'})] \subset \mathsf{prDTIME}[n^{k+1+\varepsilon}]$. ■

Next we turn to the setting of superpolynomial-time algorithms. Note that for such algorithms there is no meaningful difference between derandomization in time $T(n)^{1+\varepsilon}$ and $n \cdot T(n)^{1+\varepsilon}$ (as we take $\varepsilon > 0$ to be arbitrarily small), and we will indeed show a derandomization with the former time bound. Moreover, for this setting the required gaps between the upper-bound and the lower-bound in the hardness hypothesis (represented by parameters δ_1, δ_2) will be universal, rather than having the required gaps depend on the target derandomization overhead (represented by the parameter $\varepsilon > 0$). That is:

Theorem 11.4.9 (derandomization with almost no overhead for superpolynomial-time algorithms)**.** *There exist universal constants $c > 1$ and $\delta_2 > \delta_1 > 0$ such that for every $\varepsilon > 0$ and every $k \geq c/\varepsilon$ the following holds. Assume that there exist one-way functions that are secure against polynomial-sized circuits, and that for $T_k(n) = 2^{(1-\delta_1)\cdot kn + (c\cdot\delta_1)\cdot n}$ there exists $L \in \mathtt{amort}\text{-}\mathsf{DTIME}[2^{\delta_2\cdot n} \cdot T_k]$ such that $L \notin$ i.o.$\text{-}\mathsf{DTIME}[T_k]/2^{(1-\delta_1)\cdot n+1}$. Then, for any time-constructible and increasing $T(n) = n^{\omega(1)}$ we have $\mathsf{prBPTIME}[T(n)] \subseteq \mathsf{prDTIME}[T(n)^{1+\varepsilon}]$.*

Proof. For a sufficiently large $k \geq 1$, we instantiate Theorem 11.4.8 with parameter values $\varepsilon = 1$ and $k+1$, to deduce that our hypothesis implies that $\mathsf{prBPTIME}[n^{k+1}] \subseteq \mathsf{prDTIME}[n^{k+3}]$. Fixing any $T(n) = n^{\omega(1)}$ as in our hypothesis and fixing any promise-problem $\Pi \in \mathsf{prBPTIME}[T(n)]$, we define a padded promise-problem Π' whose "yes" instances are $\left\{ x0^{T(|x|)^{1/k}-|x|} : x \text{ is a yes instance for } \Pi \right\}$

and whose "no" instances are $\left\{x0^{T(|x|)^{1/k}-|x|} : x \text{ is a no instance for } \Pi\right\}$. Note that $\Pi' \in \mathsf{prBPTIME}[O(n^k)] \subseteq \mathsf{prBPTIME}[n^{k+1}]$, and hence $\Pi' \subseteq \mathsf{prDTIME}[O(n^{k+3})]$. By a padding argument it follows that $\Pi \in \mathsf{DTIME}[O(T(n)^{1+\frac{2}{k+3}})]$. Assuming that k is sufficiently large such that $\frac{2}{k+3} < \varepsilon$, we have that $\Pi \in \mathsf{prDTIME}[T(n)^{1+\varepsilon}]$. ■

We now combine Theorems into a proof of Theorem 11.1.2. We will actually state and prove a stronger version of Theorem 11.1.2, in which the assumed upper-bound on the hard function is amortized, rather than in worst-case. That is:

Theorem 11.4.10 (Theorem 11.1.2, stronger version)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Let $T\colon \mathbb{N} \to \mathbb{N}$ be any time-constructible non-decreasing function, and let $k = k_{\varepsilon,T} \geq 1$ be a sufficiently large constant. Assume that there exist one-way functions that are secure against polynomial-sized circuits, and that there exists $L \in \mathtt{amort}\text{-}\mathsf{DTIME}[2^{k\cdot n}]$ such that $L \notin$ i.o.-$\mathsf{DTIME}[2^{(k-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, we have that $\mathsf{prBPTIME}[T(n)] \subseteq \mathsf{prDTIME}[n \cdot T(n)^{1+\varepsilon}]$.*

Proof. Given $\varepsilon > 0$, let $\delta > 0$ be sufficiently small. In particular, we assume that δ is smaller than the values of δ_1 and δ_2 from Theorem 11.4.8 and of δ_1 and δ_2 from Theorem 11.4.9, when the latter two theorems are instantiated with the parameter value $\varepsilon/2$.

Now, if $T(n) = n^{k_0}$ for some $k_0 \geq 1$, we let $k = k_{T,\varepsilon} = (1-\delta)\cdot k_0 + (c+1)\cdot\delta > 1$, where $c > 1$ is the universal constant from Theorem 11.4.8. Our hypothesis is thus strong enough to instantiate Theorem 11.4.8 with parameter k_0 and conclude that $\mathsf{prBPTIME}[n^k] \subseteq \mathsf{prDTIME}[n^{k+1+\varepsilon/2}]$. If $k \leq k_0$ then we rely on the fact that $k_0 < k/(1-\delta)$ and on a padding argument to deduce that $\mathsf{prBPTIME}[n^{k_0}] \subseteq \mathsf{prBPTIME}[n^{k/(1-\delta)}] \subseteq \mathsf{prDTIME}[n^{\frac{k+1+\varepsilon/2}{1-\delta}}] \subseteq \mathsf{prDTIME}[n^{\frac{k_0+1+\varepsilon/2}{1-\delta}}] \subset \mathsf{prDTIME}[n \cdot n^{(1+\varepsilon)\cdot k_0}]$, where the last containment relied on δ being sufficiently small. Otherwise, if $k > k_0$, then we rely on the fact that $k < k_0 + (c+1)\cdot\delta_0$ to deduce that $\mathsf{prBPTIME}[n^{k_0}] \subseteq \mathsf{prBPTIME}[n^k] \subseteq \mathsf{prDTIME}[n^{k+1+\varepsilon/2}] \subset \mathsf{prDTIME}[n^{k_0+(c+1)\cdot\delta+1+\varepsilon/2}] \subset \mathsf{prDTIME}[n^{k_0+1+\varepsilon}]$, where again the last containment relied on δ being sufficiently small.

For a super-polynomial T, let $k_0 \geq c/\varepsilon$ be a sufficiently large constant (where $c > 1$ is the universal constant from Theorem 11.4.9) such that $k = (1-\delta)\cdot k_0 + c\cdot\delta \geq c/\varepsilon$. Our hypothesis suffices to instantiate Theorem 11.4.9 with parameter k_0 and deduce that $\mathsf{prBPTIME}[T(n)] \subseteq \mathsf{prDTIME}[T(n)^{1+\varepsilon}]$. ■

Finally, we show a partial inverse to Theorem 11.4.8. This partial inverse is the last missing piece needed to prove Proposition 11.1.6 (*i.e.*, the proposition follows from Theorem 11.4.8 and

from the partial inverse that we now prove). The proof is an adaptation of the standard proof that a PRG yields a function that is hard for non-uniform circuits, for our setting of a batch-computable PRG and distinguishers in "DTIME with advice".

Theorem 11.4.11 (batch-computable HSG $\Rightarrow$ hard function)**.** *For every $\varepsilon, \varepsilon' > 0$ there exists $\delta_1 > 0$ and $\delta_2 \geq 0$ such that for any time-constructible $T\colon \mathbb{N} \to \mathbb{N}$ the following holds. Assume that there exists a $(1/2)$-HSG for $\mathsf{DTIME}[O(n)]/T^{-1}$ with seed-length $\ell(n) = (1+\varepsilon)\cdot\log(T^{-1}(n))$ that is batch-computable in time $T^{-1}(n)^{1+\varepsilon'}\cdot n$. Then, for $T'(\ell) = T(2^{(1-\delta_1)\cdot\ell})$ there exists $L \in \mathtt{amort}\text{-}\mathsf{DTIME}[O(2^{\delta_2\cdot\ell}\cdot T'(\ell))]$ such that $L \notin$ i.o.-$\mathsf{DTIME}[T']/2^{(1-\delta_1)\cdot\ell}$.*

Proof. For $\varepsilon > 0$, let $\delta_1 = 1 - 1/(1+\varepsilon)$. Let H be the HSG from the hypothesis, and note that for every $n \in \mathbb{N}$ it holds that $2^{\ell(T(n))} = n^{1+\varepsilon}$, which by our choice of parameters implies that $n = 2^{(1-\delta_1)\cdot\ell(T(n))}$.

For $\ell \in \mathbb{N}$, let $n \in \mathbb{N}$ be the largest integer such that $\ell(T(n)) = \ell$.[24] We define L on inputs of length $\ell+1$ such that $x \notin L$ if and only if there exists a seed $w \in \{0,1\}^\ell$ such that $H(1^{T(n)}, w)$ has prefix $\ell+1$. Since H is batch-computable in time $T_H(N) = T^{-1}(N)^{1+\varepsilon'}\cdot N$, it follows that L can be decided in time

$$T_L(\ell) = O\Big(n^{1+\varepsilon'}\cdot T(n)\Big) = O\Big(2^{\ell}\cdot 2^{\delta_2\cdot\ell}\cdot T(2^{(1-\delta)\cdot\ell})\Big)\ ,$$

where $\delta_2 = \frac{\varepsilon'-\varepsilon}{1+\varepsilon}$.

Now, assume towards a contradiction that there exists an algorithm A_L that runs in time $T(2^{(1-\delta_1)\cdot\ell})$ and ues $2^{(1-\delta_1)\cdot\ell}$ bits of advice that for infinitely many ℓ's decides L correctly on inputs of length $\ell+1$ (we assume that A_L always runs in time $T(2^{(1-\delta_1)\cdot\ell})$, but we can only assume that it correctly computes L on infinitely-many input lengths). We define a Boolean function D such that on inputs of length $N = T(n)$ it holds that $D(x) = A_L(x_{\leq\ell+1})$, where $\ell = (1+\varepsilon)\cdot\log(n)$ (we define D trivially on input lengths that are not of the form $T(n)$). Our assumption about A_L implies that D can be computed in time $O(T(2^{(1-\delta_1)\cdot\ell})) = O(T(n)) = O(N)$ with at most $2^{(1-\delta_1)\cdot\ell} \leq n = T^{-1}(N)$ bits of advice; that is, $D \in \mathsf{DTIME}[O(N)]/T^{-1}$. Also, for infinitely many input lengths N it holds that $D(H(1^N, w)) = 0$ for all $w \in \{0,1\}^\ell$, whereas $\Pr_{x\in\{0,1\}^N}[D(x)] \geq 1/2$. This contradicts our hypothesis that H is a $(1/2)$-HSG for $\mathsf{DTIME}[O(N)]/T^{-1}$. ■

[24] Note that such ℓ always exists since $\ell(T(m)) \in \left[(1+\varepsilon)\cdot\log(T^{-1}(T(m))), (1+\varepsilon)\cdot\log(\lceil T^{-1}(T(m))\rceil)\right]$.

11.4.4 Extensions and Optimizations of Theorem 11.1.2

In this section our goal is to optimize the time overhead of the derandomization in Theorem 11.1.2. First, we improve the original deterministic time bound $n \cdot T(n)^{1+\varepsilon}$ to the time bound $n^{1+\varepsilon} \cdot T(n)$. As mentioned in Section 11.4.3, for probabilistic polynomial-time algorithms Theorem 11.4.8 already asserts such a time bound, and therefore it suffices to show such an improvement for superpolynomial-time algorithms.

We will do so for probabilistic algorithms that run in *at most sub-exponential time*, under the stronger hypothesis that there exists a one-way function that is secure against sub-exponential sized circuits. Let us first set up one piece of notation: We say that a function $T\colon \mathbb{N} \to \mathbb{N}$ is a valid γ'-sub-exponential function if T is time-constructible, increasing, satisfies $T(n) = 2^{n^{\gamma'}}$ and $T(n+1) \leq 2 \cdot T(n)$, and if $\lceil T^{-1} \rceil$ is time-constructible and satisfies $T^{-1}(\mathbb{N}) = \mathbb{N}$. Then, the result is the following:

Theorem 11.4.12 (optimizing the derandomization overhead for superpolynomial-time algorithms)**.** *There exists a universal constant $c > 1$ such that for every $\varepsilon > 0$ there exist $\rho > 0$ and $\delta_1, \delta_2 > 0$ for which the following holds. Assume that for some $\gamma > 0$ there exist one-way functions that are secure against circuits of size 2^{n^γ}, and let $T\colon \mathbb{N} \to \mathbb{N}$ be a valid γ'-sub-exponential function, where γ' depends on γ. For $T'(n) = 2^{(c\cdot\delta_1)\cdot n} \cdot T(2^{(1-\delta_1)\cdot n})$, assume that there exists $L \in \mathtt{amort}\text{-}\mathsf{DTIME}[2^{\delta_2 \cdot n} \cdot T'(n)]$ such that $L \notin$ i.o.$\text{-}\mathsf{DTIME}[T']/2^{(1-\delta_1)\cdot n+1}$. Then, there exists a $(n^{-\rho})$-PRG for $\mathsf{DTIME}[O(n)]/T^{-1}$ with seed length $(1+\varepsilon)\cdot\log(T^{-1}(n))$ that is batch-computable in time $T^{-1}(n)^{1+\varepsilon} \cdot n$. Consequently, $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[n^{1+\varepsilon} \cdot T(n)]$.*

Proof. The proof is analogous to the proof of Theorem 11.4.8, and we define the parameters $\varepsilon' = \varepsilon/3$ and δ_0, δ_1, and δ_2 in the exact same way. Relying on Proposition 11.4.5 and on our hypothesis, there exists a $(1/n)$-PRG that on input 1^n uses a seed of length $\log(n)^{c'}$ for some constant c' and is computable in time $\tilde{O}(n)$. We define $A(n) = \lceil T^{-1}(n) \rceil$ and $A^{-1}(n) = T(n-1)+1$, and note that $A^{-1}(n) = \min\{N : A(N) = n\}$, that $A^{-1}(n) \leq T(n)$, and that $A(n) \leq \log(n)^{c'}$ if the constant γ' is sufficiently small.

Let L be the hard problem from our hypothesis, and note that the truth-table of L_ℓ can be

printed in time

$$\begin{aligned} 2^{(1+c\cdot\delta_1+\delta_2/2)\cdot\ell}\cdot T(2^{(1-\delta_1)\cdot\ell}) &= 2^{(1+c\cdot\delta_1+\delta_2/2)\cdot\ell}\cdot\left(A^{-1}(2^{(1-\delta_1)\cdot\ell}+1)+1\right) \\ &\leq 2^{(1+c\cdot\delta_1+\delta_2/2)\cdot\ell}\cdot\left(2\cdot A^{-1}(2^{(1-\delta_1)\cdot\ell})+3\right) \\ &< 2^{(1+c\cdot\delta_1+\delta_2)\cdot\ell}\cdot A^{-1}(2^{(1-\delta_1)\cdot\ell}) \,, \end{aligned}$$

whereas no algorithm that uses at most $2^{(1-\delta_1)\cdot\ell+1}$ bits of non-uniform advice can decide L_ℓ in time $2^{(c\cdot\delta_1)\cdot n}\cdot T(2^{(1-\delta_1)\cdot n}) \geq 2^{(c\cdot\delta_1)\cdot n}\cdot A^{-1}(2^{(1-\delta_1)\cdot n})$.

Using Proposition 11.4.6 with the parameter value ε', there exists an $(n^{-\delta_0/2})$-PRG for $\mathsf{DTIME}[O(n)]/A$ with seed length $\ell(n) = (1+\varepsilon')\cdot\log(A(n)) < (1+2\varepsilon')\cdot\log(T^{-1}(n))$ that is batch-computable in time $W(n) = A(n)^{1+\varepsilon'}\cdot n < T^{-1}(n)^{1+2\varepsilon'}\cdot n$. Proposition 11.4.7 then implies that $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[O(n^{1+2\varepsilon'})\cdot T(n)] \subset \mathsf{prDTIME}[n^{1+\varepsilon}\cdot T(n)]$. ■

Finally, we can further reduce the derandomization overhead for derandomization that works in *average-case*. We will derandomize time-T algorithms in time $n^\varepsilon\cdot T(n)$ with respect to all distributions that are samplable in time $T(n)$, under a hypothesis similar to that of Theorem 11.4.12. Moreover, for every $L\in\mathsf{BPTIME}[T]$ we construct a *single deterministic algorithm* that works for all distributions that are samplable in time T. (The result extends naturally to promise-problems, and we explain this after the proof.)

Theorem 11.4.13 (average-case derandomization with almost no overhead)**.** *There exists a universal constant $c>1$ such that for every $\varepsilon>0$ there exist $\delta_1,\delta_2>0$ for which the following holds. Let $T(n)=n^k$ for a constant $k\geq 1$. Assume that there exist one-way functions secure against polynomial-sized circuits, and that for $T_k(n) = 2^{(c\cdot\delta_1)\cdot n}\cdot 2^{(1-\delta_1)\cdot(2k/\varepsilon)\cdot n}$ there exists $L_0\in\mathtt{amort}\text{-}\mathsf{DTIME}[2^{\delta_2\cdot n}\cdot T_k(n)]$ such that $L_0\notin$ i.o.$\text{-}\mathsf{DTIME}[T_k]/2^{(1-\delta_1)\cdot n+1}$. Then, for every $L\in\mathsf{BPTIME}[T]$ there exists an algorithm A that runs in time $n^\varepsilon\cdot T(n)$ such that for every distribution that can be sampled in time $T(n)$ it holds that $\Pr_{x\sim\mathcal{D}}[A(x)=L(x)] = 1-\mathtt{neg}(n)$, where $\mathtt{neg}$ is a negligible function.*

Proof. Let $L\in\mathsf{BPTIME}[T]$, let M be a probabilistic time-T machine that decides L, and let S be a probabilistic algorithm that gets input 1^n, runs in time $T(n)$, and outputs an n-bit string.

By our hypothesis and the "moreover" part of Proposition 11.4.3, for some negligible function $\mathtt{neg}$, there exists a $\mathtt{neg}$-PRG $G^{\mathtt{cry}}$ for circuits of size $\mathrm{poly}(n)$ that on input $1^{T(n)}$ uses a seed of length at most $n^{\varepsilon'}$ and is computable in time $T(n)\cdot n^{\varepsilon'}$, where $\varepsilon'=\varepsilon/2$. Let $\tilde{S}$ be an algorithm that gets input 1^n and $n^{\varepsilon'}$ bits of randomness, maps its randomness to a string of length $T(n)$ using $G^{\mathtt{cry}}$,

and applies S to the latter string; that is, $\tilde{S}(1^n, w) = S(1^n, G^{\mathtt{cry}}(1^{T(n)}, w))$. Let $\tilde{M}$ be a machine that gets input $w \in \{0,1\}^m$, maps it to a string $x = \tilde{S}(1^n, w)$ of length $n = m^{1/\varepsilon'}$, and outputs $M(x)$. Observe that on m-bit inputs $\tilde{M}$ runs in time $O(m \cdot T(m^{1/\varepsilon'})) < T(m^{2/\varepsilon'})$, and that on each input $\tilde{M}$ either outputs 0 with probability at least $2/3$ or outputs 1 with probability at least $1/3$.

Relying on Proposition 11.4.6 with the parameter ε' and the function $A(n) = \left\lceil n^{\varepsilon'/2k} \right\rceil$, and assuming that $\delta_1, \delta_2 > 0$ are sufficiently small, there exists a $(n^{-\Omega(1)})$-PRG for $\mathsf{DTIME}[O(n)]/A$, denoted G, that on inputs of length n uses a seed of length less than $\ell(n) = (3\varepsilon'/k) \cdot \log(n)$ and is batch-computable in time less than $n^{1+3\varepsilon'/k}$. By a proof analogous to the proof of Proposition 11.4.7, for every sufficiently large $m \in \mathbb{N}$ and $w \in \{0,1\}^m$, the random coins of $\tilde{M}$ can be replaced by the distribution $G(1^N, \mathbf{u}_{\ell(N)})$, where $N = 1^{T(m^{1/\varepsilon'})}$, while changing the acceptance probability by at most $1/8$.[25]

Now, observe that the random coins of $\tilde{M}$ on input $w \in \{0,1\}^m$ are only used as random coins for M on input $x = \tilde{S}(1^{m^{1/\varepsilon'}}, w)$. Therefore, for every sufficiently large $n \in \mathbb{N}$ and $x \in \{0,1\}^n$ such that $x = \tilde{S}(1^n, w)$ for some $w \in \{0,1\}^{n^{\varepsilon'}}$,[26] the random coins of M on input x can be replaced by the distribution $G(1^N, \mathbf{u}_{\ell(N)})$, where $N = T(n)$, while changing the acceptance probability by at most $1/8$.

While the latter claim was stated only for $x = \tilde{S}(1^n, w)$, we now show that this claim actually holds for almost all x's in the distribution $S(1^n, \mathbf{u}_{T(n)})$. That is:

Claim 11.4.14. *With probability at least $1 - \mathtt{neg}(T(n))$ over choice of $x \sim S(1^n, \mathbf{u}_{T(n)})$, the random coins of M can be replaced by the distribution $G(1^N, \mathbf{u}_{\ell(N)})$ while changing the acceptance probability by less than $1/6$.*

Proof. For every $x \in \{0,1\}^n$, denote $\nu(x) = \Pr[M(x, \mathbf{u}_N) = 1]$ and denote $\tilde{\nu}(x) = \Pr[M(x, G(1^N, \mathbf{u}_{\ell(N)})) = 1]$. Let $\mathtt{YES} \subseteq \{0,1\}^{T(n)}$ be the set of z's such that $x = S(1^n, z)$ satisfies $|\nu(x) - \tilde{\nu}(x)| \leq 1/8$, and let $\mathtt{NO} \subseteq \{0,1\}^{T(n)}$ be the set of z's such that $x = S(1^n, z)$ satisfies $|\nu(x) - \tilde{\nu}(x)| > 1/6$. Note that a probabilistic algorithm D can solve the promise-problem $(\mathtt{YES}, \mathtt{NO})$ in time $O(T(n))$,[27], and

[25] To see this, assume that there exist infinitely many $m \in \mathbb{N}$ such that for some $w \in \{0,1\}^m$ it holds that $\tilde{M}(w)$ is a $(1/8)$-distinguisher for $G(1^N, \mathbf{u}_{\ell(N)})$. We define an algorithm D that gets input $x \in \{0,1\}^n$, if $n \notin \left\{T(m^{1/\varepsilon'}) : m \in \mathbb{N}\right\}$ accepts, and otherwise given advice a_m computes $M(a_m, x)$ (*i.e.*, x is used as randomness). The algorithm D runs in linear time, uses $T^{-1}(n)^{\varepsilon'}$ bits of advice, and (by our assumption) $(1/8)$-distinguishes the output of G from uniform infinitely-often. This is a contradiction.

[26] We ignore rounding issues for simplicity. The more accurate statement here would be that x is the n-bit prefix of $\tilde{S}(1^n, w)$ for some $w \in \{0,1\}^{\lceil n^{\varepsilon'} \rceil}$.

[27] Specifically, D that gets input $z \in \{0,1\}^{T(n)}$, maps it to $x = S(1^n, z)$, estimates both $\nu(x)$ and $\tilde{\nu}(x)$ up to accuracy 0.01 with confidence $2/3$, and accepts if and only if $|\nu(x) - \tilde{\nu}(x)| < 1/7$.

therefore there exists a (deterministic) circuit $D' : \{0,1\}^{T(n)} \to \{0,1\}$ of size $O(T(n)^2)$ that solves $(\texttt{YES}, \texttt{NO})$ (*i.e.*, D' is obtained by hard-wiring fixed $O(T(n))$ random strings into D).

We already proved that when $z = G^{\texttt{cry}}(1^{T(n)}, w)$ for some $w \in \{0,1\}^{n^{\varepsilon'}}$, we have that $z \in \texttt{YES}$; hence, in this case $D'(z) = 1$. Assume towards a contradiction that with probability more than $\texttt{neg}(T(n))$ over choice of $x = S(1^n, \mathbf{u}_{T(n)})$ it holds that $|\nu(x) - \tilde{\nu}(x)| > 1/6$. Then, with probability more than $\texttt{neg}(T(n))$ over a uniform choice of $z \in \{0,1\}^{T(n)}$ it holds that $z \in \texttt{NO}$, in which case $D'(z) = 0$. Since $G^{\texttt{cry}}$ is a $\texttt{neg}$-PRG for circuits of arbitrary polynomial size, this is a contradiction. □

Our deterministic algorithm gets input $x \in \{0,1\}^n$, computes the output-set of G, denoted $R = \left\{ G(1^N, w) : w \in \{0,1\}^{\ell(N)} \right\}$, computes $M(x,r)$ for each $r \in R$, and outputs the majority value. Recalling that $\ell(N) = (3\varepsilon'/k) \cdot \log(N)$, this algorithm runs in time $O(n^{3\varepsilon'} \cdot T(n))$, and by Claim 11.4.14, with probability at least $1 - \texttt{neg}(N)$ over choice of $x \sim S(1^n, \mathbf{u}_{T(n)})$ this algorithm outputs $L(x)$. Finally, observe that this algorithm does not depend on the sampling algorithm S, and therefore we obtained a single deterministic algorithm that works for all distributions samplable in time $T(n)$. ■

Remark: Extension to promise-problems. The argument above extends to promise-problems (*i.e.*, to $\mathsf{prBPTIME}[T]$ rather than only $\mathsf{BPTIME}[T]$), under the additional natural hypothesis that the probability that $\mathcal{D}_n$ violates the promise is at most $\texttt{neg}(n)$.

To see this, note that the only place in the proof above where we relied on the fact that M decides a language $L \subseteq \{0,1\}^*$ was in the last paragraph. Specifically, in the first part of the proof we showed that with probability at least $1 - \texttt{neg}(n)$ over choice of $x \sim S(1^n, \mathbf{u}_{T(n)})$, we can replace the random coins of $M(x, \cdot)$ by pseudorandom coins while changing the acceptance probability by less than $1/6$. Now, relying on the hypothesis that for *every* input $x \in \{0,1\}^*$ the acceptance probability of $M(x, \cdot)$ is either at least $2/3$ or at most $1/3$, we deduced that the machine obtained by using pseudorandom coins still accepts every $x \in L$ and rejects every $x \notin L$.

Now, if we replace L by a promise-problem $\Pi = (\texttt{YES}, \texttt{NO})$ and assume that with probability at least $1 - \texttt{neg}(n)$ over $x \sim S(1^n, \mathbf{u}_{T(n)})$ it holds that x does not violate the promise, then with probability at least $1 - \texttt{neg}(n)$ it holds that replacing the random coins by pseudorandom ones changes the acceptance probability by less than $1/6$ *and* that the acceptance probability of $M(x, \cdot)$ is either at least $2/3$ or at most $1/3$. In this case, the deterministic machine outputs the correct decision at x.

11.5 Fast Derandomization via a Simple Paradigm

In this section we present our alternative proof of Theorem 11.1.1, as well as prove Theorem 11.1.8 and our conditional near-optimal quantified derandomization. In Section 11.5.1 we present a construction of a very simple reconstructive PRG for quantified derandomization, which was mentioned in Section 11.2.2. In Section 11.5.2 we explain how our proofs follow using this PRG, in high-level. Then, in Section 11.5.3 prove our results regarding quantified derandomization, and in Section 11.5.4 we prove Theorem 11.1.1 and Theorem 11.1.8. Throughout this section we ignore rounding issues for simplicity (this does not meaningfully affect our proofs).

11.5.1 A Reconstructive PRG for Quantified Derandomization

We first describe the simple PRG construction, in high level, and then provide the full proof. As mentioned in Section 11.2.2, this construction is inspired by a technical idea of Sipser [Sip88], and it is identical to a simplified construction of a PEG for the "higher-error" setting in a recent revision of [DMOZ20], and uses very similar ideas to ones used in [ACR98, MV05].

A high-level description of the construction

We first explain how to construct a hitting-set generator (HSG) with seed length $\varepsilon \cdot \log(N)$ that "hits" any distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ of size N that accepts all but at most $2^{N^{1-\varepsilon}}$ of its inputs; later on we will explain how to easily adapt the construction to obtain a PRG that "fools" all distinguishers with such extreme bias (see below). Our HSG and PRG will be reconstructive, and their reconstruction procedure will be very efficient but will use use non-determinism;[28] thus, they should be compared with known reconstructive PRGs (*e.g.*, [NW94, Uma03]), whose reconstruction procedure does not use non-determinism but has a large polynomial overhead.

The most naive construction of a HSG would be an algorithm H that evaluates f at inputs that are indexed by its random seed. Known constructions are, of course, more complicated, first encoding the truth-table of f by some useful encoding, and then using the seed in a clever way to choose bits from this encoding (see, *e.g.*, [Nis91, NW94, RRV02, TSZS06, SU05, Uma03]). In contrast, in the current context *we show that the naive construction suffices:* The algorithm H gets

[28] Thus, when instantiating this PRG with a function f that is hard for nondeterministic circuits, we deduce that the PRG "fools" all potential distinguishers D. We also comment that the reconstruction procedure actually yields an SVN circuit (rather than an arbtirary nondeterministic circuit), but for simplicity we ignore this fact in the high-level overview.

a seed s of length $\varepsilon \cdot \log(N)$ and access to a function f over $(1+\varepsilon) \cdot \log(N)$ bits, partitions the truth-table of f into N^ε parts of length N, and outputs the corresponding part f_s that is indexed by s.

Why does this naive construction work? Assume that a distinguisher D rejects all parts f_s of the truth-table of f, and recall that D rejects at most $2^{N^{1-\varepsilon}}$ strings. The intuition is that *we can describe f information-theoretically by $|D| + N^\varepsilon \cdot \log(|D^{-1}(0)|) = O(N) = o(|f|)$ bits*, using the description of D and the N^ε indices of the parts f_s in the set $D^{-1}(0)$. We now show how to leverage this information-theoretic argument to obtain an *efficient nondeterministic circuit that computes f* (*i.e.*, gets input $x \in \{0,1\}^{(1+\varepsilon)\cdot\log(N)}$ and outputs $f(x)$), which would contradict the hypothesized hardness of f.

Since $|D^{-1}(0)| \leq 2^{N^{1-\varepsilon}}$, there exists a quasilinear-time computable hash function $h\colon \{0,1\}^N \to \{0,1\}^{N^{1-\varepsilon/2}}$ that maps every distinct $y, y' \in D^{-1}(0)$ to different images.[29] Fixing such a function h, we construct a circuit C_f that has "hard-coded" the N^ε values $\{z_s = h(f_s)\}_s$.

Given an input x, the circuit C_f *nondeterministically constructs* the relevant part f_s of f that contains the location indexed by x (see below) and outputs the corresponding bit of f_s. Specifically, denoting by s the seed such that f_s contains the location indexed by x, the circuit C_f nondeterministically guesses a string f'_s, verifies that $h(f'_s)$ equals the hard-coded value z_s and that $D(f'_s) = 0$, and if these two conditions hold then it outputs the relevant location in f'_s; otherwise, it outputs $\perp$. (Indeed, this nondeterministic circuit never outputs a wrong answer.) This circuit is of size $|C_f| = O(N^{1+\varepsilon/2})$, and correctly computes f (since the only string $f'_s \in \{0,1\}^N$ such that $D(f'_s) = 0$ and $h(f'_s) = z_s$ is $f'_s = f_s$). Thus, if f (whose truth-table is of size $N^{1+\varepsilon}$) is hard for such circuits, then the HSG "hits" D.

Extending the HSG to a PRG. Let us now show how to construct a PRG for all distinguishers D that that evaluate to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\varepsilon}}$ of their inputs. Note that the required pseudorandomness property is that any such D will evaluate to σ on almost all of the outputs of the PRG; thus, if D violates this property (*i.e.*, D is indeed a distinguisher), then we know that D evaluates to $\neg\sigma$ on a noticeable fraction (say, .01) of the output-set of the PRG. This is a weaker property than in the HSG setting, in which we could assume that D evaluates to 0 on *all* of the pseudorandom strings.

[29] To be more accurate, the function h will be a standard quasilinear-time computable pairwise-independent hash function (see Theorem 11.3.12), and will satisfy a weaker property that suffices for our purposes: For every $s \in [N^\varepsilon]$, there does not exist $y \in D^{-1}(0) \setminus \{f_s\}$ such that $h(y) = h(f_s)$.

Recall that in our reconstruction algorithm for the HSG, after guessing f'_s we check that $D(f'_s) = \neg\sigma$, and otherwise output $\perp$. Applying the same approach to the current setting, we will only be able to compute f on a .01-fraction of the inputs, rather than on all inputs; that is, we reconstruct a "corrupted" version of f, denoted $\tilde{f}$. To solve this problem we simply add an initial step of encoding f by an arbitrary locally list-decodable error-correcting code that is computable in near-linear time (*e.g.*, the Reed-Muller code, as in [STV01]; see Theorem 11.3.19). Then, our actual reconstruction algorithm runs the local list-decoding procedure of this code, while answering its queries using the original reconstruction algorithm to simulate access to the "corrupted" version $\tilde{f}$.

The construction itself

Recall that standard PRGs are pseudorandom for *distinguishers*, which in our setting are modeled as non-uniform circuits. We now wish to construct a PRG for *quantified derandomization*, or in other words for the special case of distinguishers that are extremely biased. To do so we first define such distinguishers, as follows:

Definition 11.5.1 (quantified distinguisher)**.** *Let* $\mathbf{w}$ *be a distribution over* $\{0,1\}^N$ *and let* $B = B(N)$*. We say that a circuit* $D\colon \{0,1\}^N \to \{0,1\}$ *is a* B-quantified ρ-distinguisher for $\mathbf{w}$ *if for some* $\sigma \in \{0,1\}$ *it holds that* $\left|\{x : D(x) = \sigma\}\right| \leq B$ *and* $\Pr[D(\mathbf{w}) = \sigma] \geq \rho$*.*

Our PRG will be reconstructive. Its reconstruction procedure will be modeled as a nondeterministic oracle machine R that gets oracle access to a quantified distinguisher D as well as a bounded number of non-uniform advice bits, and is able to compute the function nondeterministically and unambiguously (see Definition 11.3.3). In more detail:

Proposition 11.5.2 (a reconstructive PRG for quantified derandomization)**.** *For every* $\alpha > 0$ *there exists* $\mu > 0$ *such that for every two constants* $\beta, \gamma > 0$ *the following holds. There exists an oracle machine* G *that, when given input* 1^N *and a random seed of length* $\ell = (\alpha + \beta) \cdot n$ *where* $n = \log(N)$ *and oracle access to* $f\colon \{0,1\}^{(1+\beta)\cdot n} \to \{0,1\}$*, satisfies the following:*

1. *The machine* G *runs in time* $\tilde{O}(N^{1+\alpha+\beta})$ *and outputs* N *bits.*
2. *There exists an oracle machine* R *that, when given oracle access to a* $2^{N^{1-\gamma}}$*-biased* $(N^{-\mu})$*-distinguisher* D *for* $G^f(1^N, \mathbf{u}_\ell)$*, computes* f *nondeterministically and unambiguously in time* $N^{1+\alpha}$*, using* N^α *oracle queries to* D *and* $O(N + N^{1+\beta+(2\alpha-\gamma)})$ *bits of advice.*

Proof. We identify f with its truth-table $f \in \{0,1\}^{N^{1+\beta}}$.

The generator G. For sufficiently small constants $\eta > \mu > 0$, let $\bar{f} = Enc(f)$ where Enc is the code from Corollary 11.3.20, instantiated with $m = |f|$ and $\rho = N^{-\mu}$ and the constant $\eta > 0$; note that $\bar{f} \in \{0,1\}^{\bar{N}\cdot\log(|\Sigma|)}$, where $\bar{N} = O\left(|f| \cdot N^{2\mu/\eta'}\right) < N^{1+\beta+\alpha}$ (relying on a sufficiently small choice of μ) and $|\Sigma| = O\left(|f|^{\eta'} \cdot N^{2\mu}\right) < N^{(1+\beta)\cdot\eta}$.

For every $s \in \left[|\bar{f}|/N\right]$, let $\bar{f}_s \in \{0,1\}^N$ be the s^{th} consecutive substring of length N of $\bar{f}$ (*i.e.*, for $i \in [n]$ it holds that the i^{th} bit of $\bar{f}_s$ equals $\bar{f}_{(s-1)\cdot N+i}$). The machine G gets $s \in [|\bar{f}|/N]$ as a seed and outputs $\bar{f}_s$. Note that $|\bar{f}|/N < N^{\beta+\alpha}$, and therefore $|s| < (\beta+\alpha)\cdot\log(N)$. Also note that the running-time of G is dominated by the time it takes to compute $\bar{f}$, which is $\tilde{O}(N^{1+\beta+\alpha})$.

Reconstructing a corrupt version of $\bar{f}$. For $\rho = N^{-\mu}$, let $D\colon \{0,1\}^N \to \{0,1\}$ be a $2^{N^{1-\gamma}}$-quantified ρ-distinguisher for G^f, let $\sigma \in \{0,1\}$ be the rare output of D, and denote $S_\sigma = D^{-1}(\sigma)$. We first describe an algorithm $\tilde{R}^D$ that computes a function $\tilde{f} \in \Sigma^{\bar{N}}$ that agrees with $\bar{f} \in \Sigma^{\bar{N}}$ on at least a ρ-fraction of their inputs. To do so, let $\mathcal{H} \subseteq \left\{\{0,1\}^N \to \{0,1\}^{N^{1-\gamma+\alpha}}\right\}$ be the family of quasilinear-time computable functions from Theorem 11.3.12 (instantiated with parameters $m = N$ and $m' = m^{\gamma-\alpha}$). For every fixed $s \in [N^{2\beta}]$ we have that

$$\Pr_{h\in\mathcal{H}}\left[\exists g \in S_\sigma : h(\bar{f}_s) = h(g)\right] \leq |S_\sigma| \cdot 2^{-N^{1-\gamma+\alpha}} \leq 2^{-N^{1-\gamma+\alpha}+N^{1-\gamma}},$$

where the last inequality relied on the fact that $|S_\sigma| \leq 2^{N^{1-\gamma}}$. By a union-bound, there exists some $h \in \mathcal{H}$ such that for every $s \in [N^{\beta+\alpha}]$ there does not exist $g \in S_\sigma \setminus \{\bar{f}_s\}$ for which $h(\bar{f}_s) = h(g)$. Let us now fix such an h.

The algorithm $\tilde{R}$ gets as advice the bit σ, the description of h, and all the $N^{\beta+\alpha}$ strings $\{h(\bar{f}_s) : s \in [N^{\beta+\alpha}]\}$, which constitute $O\left(N + N^{1+\beta+2\alpha-\gamma}\right)$ bits of advice. Given input $x \in [\bar{N}]$, the algorithm:

1. Nondeterministically guesses $g \in \{0,1\}^N$.
2. Queries D on input g, and outputs $\perp$ unless $D(g) = \sigma$.
3. For the appropriate s (such that x indexes a location in $\bar{f}_s$), the algorithm verifies that $h(g) = h(\bar{f}_s)$, and otherwise outputs $\perp$.
4. Outputs the symbol of g that appears in the location indexed by x.

Note that the running-time of $\tilde{R}$ is dominated by the computation of h, and is thus bounded by $\tilde{O}(N)$. We now claim that there exists a set $T \subseteq [\bar{N}]$ of density at least ρ such that for every $x \in T$ it holds that $\tilde{R}(x)$ nondeterministically computes $\bar{f}(x)$.[30]

[30] By "nondeterministically computes $f(x)$" we mean that for some nondeterministic choices $\tilde{R}(x) = f(x)$, whereas for all nondeterministic choices $\tilde{R}(x) \in \{\bar{f}(x), \perp\}$.

To see this, let $S = \{s \in [N^{\beta+\alpha}] : D(\bar{f}_s) \neq \sigma\}$, and recall that (by the properties of D) we have that $|S|/N^{\beta+\alpha} \geq \rho$. By the properties of h, for every $s \in S$ there does not exist $g \in D^{-1}(\neg\sigma) \setminus \{\bar{f}_s\}$ such that $h(g) = h(\bar{f}_s)$. Hence, by the construction of $\tilde{R}$ above, for every x that indexes a location in $\bar{f}_s$ for some $s \in S$ we have that $\tilde{R}(x)$ nondeterministically computes $\bar{f}(x)$.[31] We define the set $T \subseteq [\bar{N}]$ to consist of all x that index locations in $\bar{f}_s$ for some $s \in S$, and indeed we have that $|T|/\bar{N} \geq \rho$.

Reconstructing f. We now run the local list-decoding algorithm for f from Corollary 11.3.20, denoted *Dec*, while giving it oracle access to $\tilde{R}$. The underlying oracle machine runs in time $N^{\eta} \cdot (1/\rho)^{1/\eta'} < N^{\alpha/2}$ (relying on a sufficiently small choice of $\eta, \mu > 0$) and uses $\log(O(1/\rho)) < \log(N)$ bits of non-uniform advice.

Answering the queries to $\tilde{R}$ in the latter oracle machine with the actual oracle machine for $\tilde{R}$, we obtain a randomized procedure that nondeterministically computes f in time $N^{\alpha/2} \cdot \tilde{O}(N) = \tilde{O}(N^{1+\alpha/2})$ that makes at most N^{α} oracle queries to D and uses

$$O(N + N^{1+\beta+2\alpha-\gamma} + \log(N)) = O(N + N^{1+\beta+2\alpha-\gamma})$$

bits of non-uniform advice (that depends on D and on f). Using naive error-reduction and fixing an appropriate random string, we obtain an algorithm R that computes f nondeterministically, unambiguously and without randomness. The running-time of R is $\tilde{O}(N^{1+\alpha/2}) < N^{1+\alpha}$ and the number of advice bits is still $O(N + N^{1+\beta+2\alpha-\gamma})$. ■

Remark: Comparison to Kolmogorov-based derandomization. The proof of Proposition 11.5.2 can be viewed as a "constructive" (*i.e.*, efficient) variant of the classical technique of derandomizing probabilistic algorithms using *strings with high Kolmogorov complexity* (see, *e.g.*, [LV08, Thm 7.3.5]). To see this, recall that for any $\bar{D}\colon \{0,1\}^{\bar{N}} \to$ of size $\bar{N}^{.99}$ that accepts all but $2^{\bar{N}^{.99}}$ of its inputs, and any string $f \in \{0,1\}^{\bar{N}}$ with maximal Kolmogorov complexity $\bar{N}$, we have that $\bar{D}(f) = 1$. This is the case since otherwise the string f would have a description of length $O(\bar{N}^{.99})$, consisting of the circuit $\bar{D}$ along with the index $i \in \{0,1\}^{.99 \cdot \bar{N}}$ of f inside the set $\bar{D}^{-1}(0)$. In fact, the argument works even when f only has high *time-bounded Kolmogorov complexity*.[32]

[31] For simplicity, we ignore rounding issues at this point, and assume that blocks of size N in $\bar{f}$ do not truncate blocks of size $\log(|\Sigma|) < (1+\beta) \cdot \eta \cdot \log(N)$.

[32] Recall that the time-bounded Kolmogorov complexity of $f \in \{0,1\}^*$, defined by Levin, is the minimum over $\langle M \rangle + \log(t)$ such that $\langle M \rangle$ is the description of a machine that prints f in time t.

In the reduction in our argument, instead of only showing that f has small time-bounded Kolmogorov complexity, we show that f has small circuit complexity; in other words, we show that f can be computed in a "strongly-explicit" manner by an SVN circuit that gets as input an index $x \in [\log(|f|)]$ and outputs f_x. We note that while the analysis of this argument relies only on classical notions such as hash functions and error-reduction, the instantiation above uses technical constructions that are more recent than in Sipser's time, such as very efficient samplers and hash functions.

11.5.2 High-level Description of the Proofs

Let us now describe the proofs of our results in the current section, in an informal and high-level way. Our goal is to derandomize algorithms that run in time $N = T(n)$, and to do so we will use a PRG that "fools" circuits $D\colon \{0,1\}^N \to \{0,1\}$ of size N. Similarly to Section 11.2, we denote by $\varepsilon > 0$ a very small constant, and construct ε-PRGs (the extension to $n^{-.01}$-PRGs is straightforward).

The proof of our near-optimal quantified derandomization follows easily from Proposition 11.5.2. Specifically, for this proof we only need to consider distinguishers that evaluate to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\varepsilon}}$ of their inputs, and we instantiate the PRG from Proposition 11.5.2 with a function f of truth-table size $|f| = N^{1+\varepsilon}$ that is hard for SVN circuits of size $O(N^{1+\varepsilon/2})$. Our algorithm for quantified derandomization first computes the truth-table of f, which can be done in time $N^{1+O(\varepsilon)}$ by our hypothesis, and then encodes it using a locally-list-decodable code that is computable in near-linear time $\tilde{O}(N^{1+\varepsilon})$; then it enumerates over the N-bit consecutive parts of the encoding of f (which constitute the output-set of the PRG), and outputs the majority of the evaluations of D on these parts. The running-time of this algorithm is at most $N^{1+O(\varepsilon)}$.

To prove Theorem 11.1.8 we need to consider *all* distinguishers $D\colon \{0,1\}^N \to \{0,1\}$ of size N, rather than only very biased distinguishers. Let $\mathsf{Samp}\colon \{0,1\}^{\bar{N}} \times \{0,1\}^{(1+O(\varepsilon))\cdot\log(\bar{N})} \to \{0,1\}^N$ be a linear-time-computable averaging sampler with accuracy $1/10$ and confidence $2^{\bar{N}^{1-\varepsilon}-\bar{N}}$, where $\bar{N} = N^{1+O(\varepsilon)}$.[33] For any potential distinguisher D, denote the (actual) acceptance probability of D by $\mu = \Pr_r[D(r) = 1]$, and define $\bar{D}\colon \{0,1\}^{\bar{N}} \to \{0,1\}$ such that $\bar{D}(z) = 1$ if and only if $\Pr_r\left[\mathsf{Samp}(z,r) \in D^{-1}(1)\right] \in \mu \pm (1/10)$ (where the $1/10$ term corresponds to the error of the sampler). We stress that *our algorithm does not actually construct* $\bar{D}$ (*i.e.*, $\bar{D}$ is a mental experiment for the analysis), and therefore there is no problem to "hard-wire" μ into $\bar{D}$. Note that $\bar{D}$ is of size

[33] As noted in [DMOZ20], such a construction follows by re-analyzing a well-known construction of [TSZS06, Thm 5] for the min-entropy value $\bar{N}^{1-\varepsilon}$; see Theorem 11.3.17 for a formal statement.

$N^{2+O(\varepsilon)}$ and accepts all but $2^{\bar{N}^{1-\varepsilon}}$ of its $\bar{N}$-bit inputs. By a standard analysis, any .01-PRG G_0 for $\bar{D}$ yields a $(1/9)$-PRG $G(s,r) = \mathsf{Samp}(G_0(s),r)$ for D. [34] Therefore, we just need a .01-PRG for $\bar{D}$.

We then use the PRG from Proposition 11.5.2 with parameters as follows. We fix a function f of truth-table size $N^{2+O(\varepsilon)}$, and assume that it is hard for SVN circuits of size $|f|^{1-\varepsilon}$. The derandomization algorithm computes the truth-table of f, encodes it with the error-correcting code to obtain a codeword $\bar{f}$, and outputs the majority of the evaluations of D on the set $\left\{\mathsf{Samp}(\bar{f}_s,r)\right\}_{s,r}$. This set is of size $N^{2+O(\varepsilon)}$ (since each of the sets of values for s and for r is of size $N^{1+O(\varepsilon)}$), evaluating D on the set takes time $N^{3+O(\varepsilon)}$. Thus, we obtain derandomization in either cubic time or quartic time, depending on the time that it takes to compute the truth-table of f (this corresponds to the two different result statements in Theorem 11.1.8).

Finally, to prove Theorem 11.1.1, we want to reduce the running time in the proof of Theorem 11.1.8 from cubic (or quartic) to quadratic. The main bottleneck is that the circuit $\bar{D}$ is of size more than N^2. [35] To decrease the size of $\bar{D}$, recall that in the context of Theorem 11.1.1 we assume hardness for *randomized* nondeterministic circuits, and therefore we are allowed to use randomness in the definition of $\bar{D}$ (*i.e.*, the reconstruction procedure from Proposition 11.5.2 will transform a randomized circuit $\bar{D}$ into a randomized SVN circuit that computes the hard function).

Thus, we define $\bar{D}$ in a way that utilizes this randomness in order to perform error-reduction more efficiently. Specifically, instead of defining $\bar{D}$ such that it computes $\nu(z) = \Pr_r[D(\mathsf{Samp}(z,r)) = 1]$ exactly, the circuit $\bar{D}$ estimates $\nu(z)$ up to error .01 using random sampling of r's, and accepts if and only if its estimate $\tilde{\nu}(z)$ is in the interval $\mu \pm .01$. This handles the main bottleneck in the proof, since the resulting circuit $\bar{D}$ is now of size $N^{1+O(\varepsilon)}$, and therefore we can "fool" it using a hard function f whose truth-table is only of size $N^{1+O(\varepsilon)}$. Overall, since f is computable in exponential time (in its input length), we can compute its truth-table in time $N^{2+O(\varepsilon)}$, encode it using the code (in time $N^{1+O(\varepsilon)}$), and output the majority of evaluations of D on the set $\left\{\mathsf{Samp}(\bar{f}_s,r)\right\}_{r,s}$, which is now of size $N^{1+O(\varepsilon)}$ (since $\bar{f}$ is of size $N^{1+O(\varepsilon)}$ and hence there are at most $N^{O(\varepsilon)}$ values for s). This yields derandomization with quadratic overhead.

[34] To see this, call a string z good if $\Pr_r\left[\mathsf{Samp}(z,r) \in D^{-1}(1)\right] \in \mu \pm (1/10)$. Then, for any $\sigma \in \{0,1\}$ it holds that $\Pr_{s,r}[\mathsf{Samp}(G(s),r) \in D^{-1}(\sigma)] \leq \Pr[G(s) \text{ is not good}] + \mu + .01 < \mu + 1/9$.

[35] To see why this is a problem, observe that "fooling" $\bar{D}$ using our approach requires a function whose truth-table is of size more than $|\bar{D}| \geq N^2$. Thus, the PRG G_0 will have seed length at least $\log(N)$, and the final PRG (*i.e.*, G_0 composed with Samp) will have seed length at least $2 \cdot \log(N)$. Evaluating D at each of the N^2 outputs will take time at least N^3.

11.5.3 Near-optimal Quantified Derandomization

Relying on Proposition 11.5.2, we now present two very efficient solutions for the quantified derandomization problem: The first is an unconditional construction of a *non-uniform* circuit family (*i.e.*, a non-explicit PRG), and the second is a conditional construction of an algorithm that solves the problem, where the hypothesis refers to the existence of a sufficiently hard "batch-computable" function (see below). Specifically, we first show that, unconditionally, there exist a non-uniform PRG for quantified derandomization of circuits with at most $2^{N^{1-\gamma}}$ exceptional inputs that has seed length $\gamma \cdot \log(N)$.

Theorem 11.5.3 (non-explicit PRG with short seed)**.** *For every $\gamma > 0$ there exist $\mu > 0$ such that the following holds. For every $N \in \mathbb{N}$ there exist N^γ strings $w_1, ..., w_{N^\gamma} \in \{0,1\}^N$ such that for every circuit $D\colon \{0,1\}^N \to \{0,1\}$ of size N that evaluates to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\gamma}}$ it holds that $\Pr_{i \in [N^\gamma]}[D(w_i) = \sigma] \geq 1 - N^{-\mu}$.*

Proof. Let $\alpha = \gamma/4$ and $\beta = 3\gamma/4$. By a standard counting argument, there exists a function f whose truth-table is of size $N^{1+\beta}$ that cannot be computed by SVN circuits of size $N^{1+\beta-\gamma/2}$. The strings $w_1, ..., w_{N^\gamma}$ will be the output-set of the machine G from Proposition 11.5.2, when the latter is instantiated with f as the hard function and with parameters $\alpha = \gamma/4$ and β as above. Note that the output-set of G is indeed of size $N^{(\alpha+\beta)} = N^\gamma$. The claim follows by noting that there does not exist a $2^{N^{1-\gamma}}$-biased $(N^{-\mu})$-distinguisher for G, otherwise f could be computed by an SVN circuit of size $O\left(N^{1+\beta+(2\alpha-\gamma)} + N^{1+\alpha}\right) = O\left(N^{1+\beta-\gamma/2}\right)$. ■

Our second result asserts that if there exists a function whose entire truth-table on n-bit inputs can be printed in time $2^{(1.01)\cdot n}$, but that cannot be computed (on an input-by-input basis) by SVN circuits of size $2^{.99 \cdot n}$, then we can solve the quantified derandomization problem with near-linear time overhead.

Theorem 11.5.4 (near-optimal quantified derandomization)**.** *For every $\gamma > 0$ there exists $\mu > 0$ such that for every $\delta > 0$ the following holds. Assume that there exists $L \in \mathsf{DTIME}[2^n]$ such that for every $n \in \mathbb{N}$ it holds that the truth-table of L can be printed in time $2^{(1+\delta)\cdot n}$, but L cannot be computed by SVN circuits of size $2^{(1-\gamma/4)\cdot n}$. Then, there exists an $(N^{-\mu})$-PRG with seed length $\gamma \cdot \log(N)$ for the class of circuits of linear size that evaluate to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\gamma}}$ of their inputs such that the entire output-set of the PRG can be printed in time $N^{(1+\gamma)\cdot(1+\delta)}$.*

The conclusion of Theorem 11.5.4 implies that randomized time-T algorithms that err on at most $2^{T(n)^{1-\gamma}}$ of their random choices can be deterministically simulated in $O\left(T(n)^{(1+\gamma)\cdot(1+\delta)} + T(n)^{1+\gamma}\right) < T(n)^{1+2\gamma+\delta}$ time. (This follows by the standard reduction of prBPP to the circuit acceptance probability problem; see, *e.g.*, [Vad12, Corollary 2.31] or [Gol08, Exercise 6.14].)

Proof of Theorem 11.5.4. Given $D\colon \{0,1\}^N \to \{0,1\}$, let $\alpha = \gamma/4$, let $\beta = 3\gamma/4$, and let f be a function as in our hypothesis whose truth-table is of size $N^{1+\beta}$. We instantiate the PRG from Proposition 11.5.2 with these parameters and with f as the hard function, and claim that D evaluates to σ over $1 - N^{-\mu}$ of the outputs of the PRG (where μ is as in Proposition 11.5.2). This holds since otherwise D is a $2^{N^{1-\gamma}}$-biased $(N^{-\mu})$-distinguisher for the PRG, and hence f can be computed by an SVN circuit of size

$$O\left(N^{1+\alpha} + N^{1+\beta+(2\alpha-\gamma)}\right) = O\left(N^{1+\beta-\gamma/2}\right) < N^{(1+\beta)\cdot(1-\gamma/4)},$$

which is a contradiction.

To print the output-set of the PRG, we first construct the truth-table of f, which can be done in time $N^{(1+\beta)\cdot(1+\delta)} < N^{(1+\gamma)\cdot(1+\delta)}$. Then, for each of the N^γ seeds we can compute the corresponding N-bit output in time $\tilde{O}(N^{1+\alpha+\beta}) = \tilde{O}(N^{1+\gamma})$. ■

11.5.4 Standard Derandomization: Proofs of Theorem 11.1.1 and Theorem 11.1.8

We now prove Theorem 11.1.1 and Theorem 11.1.8. Both results will first reduce the standard derandomization problem to a quantified derandomization problem, and then use the reconstructive PRG from Proposition 11.5.2 to solve the latter. The main difference between the proofs is a different reduction to quantified derandomization.

In the following result statements, the conclusions will be that there exists a PRG with seed length $c \cdot \log(N)$ whose entire output-set is computable in time $c' \cdot \log(N)$ for some small $c, c' \in \mathbb{N}$. To see how these imply the result statements in Section 11.1, recall that this allows us to solve CAPP in time $N^{c'} + N^{c+1}$ (by evaluating a given N-bit circuit of size N over the output-set of the PRG), which in turn implies that randomized algorithms that run in time $N = T(|x|)$ can be deterministically simulated in time $O\left(T(|x|)^{\max\{c',c+1\}}\right)$; that is, $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[T^{\max\{c',c+1\}}]$.

Theorem 11.5.5 (Theorem 11.1.8, restated)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \in \mathsf{DTIME}[2^n]$ such that for every $n \in \mathbb{N}$ it holds that L cannot be*

computed by SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, there exists an $(N^{-\delta})$-PRG with seed length $(2+\varepsilon)\cdot\log(N)$ whose entire output-set can be printed in time $N^{4+\varepsilon}$. Moreover, if for every $n\in\mathbb{N}$ the entire truth-table of L on n-bit inputs can be printed in time $2^{(3/2)\cdot n}$, then the output-set of the PRG can be printed in time $N^{3+\varepsilon}$.

Proof. We first specify the parameters that we will use in the proof, and instantiate the hard function, the reconstructive PRG from Proposition 11.5.2 and the extractor from Theorem 11.3.17 with these parameters.

- For sufficiently small constants $\gamma,\delta>0$ that depends on ε, let $\alpha=\gamma/4$ and let $\beta=1+3c\cdot\gamma$.
- Let $\mathsf{Ext}\colon\{0,1\}^{\bar{N}}\times\{0,1\}^{(1+2c\cdot\gamma)\cdot\log(\bar{N})}\to\{0,1\}^{N}$ be the extractor from Theorem 11.3.17, instantiated with error $\frac{1}{2}\cdot N^{-\delta}$ for a sufficiently small $\delta>0$, where $\bar{N}=N^{1+3\gamma}>(c\cdot N)^{1/(1-2\gamma)}$.
- Let $f\in\{0,1\}^{\bar{N}^{1+\beta}}$ be a function that can be computed in time $\bar{N}^{1+\beta}$ but cannot be computed by SVN circuits of size $\bar{N}^{1+\beta-\gamma/4}<\bar{N}^{1-\delta}$.
- Let G_0 be the reconstructive PRG from Proposition 11.5.2, instantiated for output length $\bar{N}$ with f as the hard function and with the parameters α,β,γ.

Our PRG is defined by $G(1^N,(s_0,s_1))=\mathsf{Ext}(G_0(1^{\bar{N}},s_0),s_1)$. Note that the seed length of G is $(1+2c\cdot\gamma+\alpha+\beta)\cdot\log(\bar{N})<(2+\varepsilon)\cdot\log(N)$, and that G can be computed (on a single seed) in time $\tilde{O}(N^{1+\alpha+\beta})<N^{1+\varepsilon}$. Moreover, we can print the output-set of G by first computing the truth-table of f in time $\bar{N}^{2\cdot(1+\beta)}=\bar{N}^{4+6c\cdot\gamma}<N^{4+\varepsilon}$, and then for each of the $N^{2+\varepsilon}$ seeds computing the corresponding output of G in time less than $N^{1+\varepsilon}$. Thus, we can print the entire output-set of G in time $N^{4+\varepsilon}$.

Fixing any circuit $D\colon\{0,1\}^{N}\to\{0,1\}$ of size N, we now want to show that G "fools" D with error $N^{-\delta}$. To do so we define $\bar{D}\colon\{0,1\}^{\bar{N}}\to\{0,1\}$ such that $\bar{D}(z)=1$ if and only if $\left|\Pr_{s_1}\left[D(\mathsf{Ext}(z,s_1)=1)\right]-\Pr_r[D(r)=1]\right|\leq N^{-\delta}$. Note that $\bar{D}$ can be computed by a circuit of size $O\left(2^{(1+2c\cdot\gamma)\cdot\log(\bar{N})}\cdot\bar{N}\right)=O(\bar{N}^{2+2c\cdot\gamma})$, and that $\Pr_z[\bar{D}(z)=0]\leq 2^{\bar{N}^{1-\gamma}-\bar{N}}$. It follows that $\Pr_{s_0}[G_0(1^{\bar{N}},s_0)\in\bar{D}^{-1}(0)]\leq\frac{1}{2}\cdot N^{-\delta}$ (assuming $\delta>0$ is sufficiently small), otherwise the combination of the oracle machine R and of the distinguisher $\bar{D}$ yields an SVN circuit that computes f of size

$$O\left(\bar{N}^{1+\alpha}+\bar{N}^{\alpha}\cdot\bar{N}^{2+2c\cdot\gamma}+\bar{N}^{1+\beta+(2\alpha-\gamma)}\right)<\bar{N}^{1+\beta-\gamma/4}\,. \tag{11.5.1}$$

Therefore, for every $\sigma \in \{0,1\}$ we have that

$$\begin{aligned}
&\Pr_{s_0,s_1}\left[D(G(1^N,(s_0,s_1))) = \sigma\right] \\
&\quad \leq \Pr_{s_0}\left[G_0(1^{\bar{N}},s_0) \notin \bar{D}^{-1}(1)\right] + \Pr_{s_0,s_1}\left[\mathsf{Ext}(G_0(1^{\bar{N}},s_0),s_1) \in D^{-1}(\sigma) | G_0(1^{\bar{N}},s_0) \in \bar{D}^{-1}(1)\right] \\
&\quad \leq \Pr_r[D(r) = \sigma] + N^{-\delta}\,,
\end{aligned}$$

which means that G "fools" D with error $N^{-\delta}$.

The "moreover" part by noting that under the stronger hypothesis, the first step of computing the truth-table of f takes time $\bar{N}^{(3/2)\cdot(1+\beta)} < N^{3+\varepsilon}$, and it still dominates the running-time of the entire algorithm that prints the output-set of the PRG. ■

Next, we show our alternative and simple proof of Theorem 11.1.1. The high-level outline of this proof is similar to the proof of Theorem 11.5.5, but we will carry out the error-reduction (manifested in the circuit $\bar{D}$) in a more efficient way. In more detail, since we assume that the "hard" function cannot be computed by *randomized* SVN circuits, the existence of a randomized distinguisher would still contradict our hypothesis (*i.e.*, we can use the reconstruction procedure with a randomized distinguisher); accordingly, we will use randomness to reduce the error of the distinguisher more efficiently (*i.e.*, by a smaller circuit). Details follow.

Theorem 11.5.6 (Theorem 11.1.1, restated)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \in \mathsf{DTIME}[2^n]$ such that L cannot be computed by randomized SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, there exists an $(N^{-\delta})$-PRG with seed length $(1+\varepsilon)\cdot\log(N)$ whose entire output-set can be printed in time $N^{2+\varepsilon}$.*

Proof. As in the proof of Theorem 11.5.5, we first specify the parameters and instantiate the hard function, the extractor, and the reconstructive PRG:

- For sufficiently small $\gamma, \delta > 0$, let $\alpha = \gamma/5$, let $\beta = 3\gamma$, and let $\gamma' = \gamma/2$.
- Let $\mathsf{Ext}\colon \{0,1\}^{\bar{N}} \times \{0,1\}^{(1+2c\cdot\gamma)\cdot\log(\bar{N})} \to \{0,1\}^N$ be the extractor from Theorem 11.3.17, instantiated with error $\frac{1}{6}\cdot N^{-\delta}$ and with $\bar{N} = N^{1+3\gamma}$.
- Let $f \in \{0,1\}^{\bar{N}^{1+\beta}}$ be a function that can be computed in time $\bar{N}^{1+\beta}$ but cannot be computed by randomized SVN circuits of size $\bar{N}^{1+\beta-\gamma/4}$.
- Let G_0 be the PRG from Proposition 11.5.2, instantiated for output length $\bar{N}$ with the function f and with parameters α, β, γ'.

Our PRG is defined by $G(s_0, s_1) = \mathsf{Ext}(G(s_0), s_1)$. This PRG has seed length $(1 + 2c \cdot \gamma + (\alpha + \beta)) \cdot \log(\bar{N}) < (1 + \varepsilon) \cdot \log(N)$ and it can be computed in time less than $N^{1+\varepsilon}$ given access to f. Printing the entire output-set of G can be done in time less than $N^{2+\varepsilon}$, by first computing the truth-table of f in time $\bar{N}^{2\cdot(1+\beta)} < N^{2+\varepsilon}$ and then evaluating G on each of the $N^{1+\varepsilon}$ seeds.

Now fix a circuit $D\colon \{0,1\}^N \to \{0,1\}$ of size N. For any $z \in \{0,1\}^{\bar{N}}$, we denote by $\mu(z) = \Pr_{s_1}[D(\mathsf{Ext}(z, s_1)) = 1]$ the average value of D over the sample $\mathsf{Ext}(z, \cdot)$, and we also denote by $\nu = \Pr_r[D(r) = 1]$ the true average value of D. We will now define a circuit $\bar{D}\colon \{0,1\}^{\bar{N}} \to \{0,1\}$ of size $\tilde{O}(N^{1+2\delta})$ such that:

1. The circuit $\bar{D}$ accepts all but at most $2^{\bar{N}^{1-\gamma}}$ of its inputs.
2. For every $z \in \bar{D}^{-1}(1)$ it holds that $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$.
3. The top gate of $\bar{D}$ is a CAPP gate: That is, a gate that takes as input a description of a circuit C', outputs 1 if $\Pr[C'(x) = 1] \geq 2/3$, outputs 0 if $\Pr[C'(x) = 1] \leq 1/3$, and otherwise outputs *some* bit (*i.e.*, we do not care how it behaves on inputs that violate the promise).

To define $\bar{D}$, for $z \in \{0,1\}^{\bar{N}}$ and $t = (1 + 2c \cdot \gamma) \cdot \log(\bar{N})$, let T_z be a circuit that gets as input $O\left(t \cdot N^{2\delta}\right)$ bits, uses these bits to obtain an estimate of $\mu(z)$, and outputs 1 if and only if its estimate is in the interval $\nu \pm \frac{1}{3} \cdot N^{-\delta}$. Note that T_z is of size $\tilde{O}(N^{1+2\delta})$, and that for at least $2/3$ of the inputs of T_z it holds that the estimate is correct up to accuracy $\frac{1}{6} \cdot N^{-\delta}$. In particular, if $\Pr_r[T_z(r) = 0] < 2/3$ then $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$. The circuit $\bar{D}$ gets input z, constructs the circuit T_z, and feeds T_z into the top CAPP gate. Note that the size of $\bar{D}$ is $\tilde{O}(N^{1+2\delta})$, for every $z \in \bar{D}^{-1}(1)$ it holds that $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$, and for all but at most $2^{\bar{N}^{1-\gamma}}$ of the inputs z it holds that $\bar{D}(z) = 1$. [36]

We claim that $\Pr_{s_0}[G_0(1^{\bar{N}}, s_0) \in \bar{D}^{-1}(0)] \leq \frac{1}{2} \cdot N^{-\delta}$ (assuming $\delta > 0$ is sufficiently small). To see this, note that otherwise, the combination of the oracle machine R and of the distinguisher $\bar{D}$ yields an SVN circuit with CAPP gates that computes f in size

$$\tilde{O}\left(\bar{N}^{1+\alpha} + \bar{N}^{\alpha} \cdot \bar{N}^{1+2\delta} + \bar{N}^{1+\beta+(2\alpha-\gamma)}\right) < \bar{N}^{1+\beta-\gamma/4}\,.$$

We now want to transform the foregoing SVN circuit with CAPP gates, denoted C_f, into a randomized SVN circuit. To do so, for each CAPP gate g, the randomized SVN circuit samples $O(\log(\bar{N}))$ inputs for the circuit C' that feeds into g, estimates the acceptance probability of C' up to accuracy .01 and with error $1/N^2$, and sets g's output to 1 iff the estimate is above $1/2$. Note that

[36] This holds because for all but at most $2^{\bar{N}^{1-\gamma}}$ inputs z it holds that $\mu(z) \in \nu \pm \frac{1}{6} \cdot N^{-\delta}$, and for every such input the circuit T_z accepts with probability at least $2/3$.

for every input i to C_f, every nondeterministic choices w by C_f, and every gate g that gets circuit $C' = C'(i, w)$,

- If C' satisfies the promise of CAPP (*i.e.*, it has at most $1/3$ exceptional inputs) then with probability at least $1 - 1/N^2$ we compute g correctly.
- Otherwise, the circuit C_f correctly computes f *regardless of the output* of g. (This is because in the definition of a CAPP gate for $\bar{D}$ we did not enforce its behavior on inputs that violate the promise, and thus the proof holds regardless of the outputs of the original CAPP gate on circuits that violate the promise.)

Recalling that there are at most $\bar{N}^\alpha < \frac{1}{3} \cdot N^2$ such CAPP gates in C_f (since the reconstruction procedure makes at most $\bar{N}^\alpha$ oracle calls), by a union-bound, for every input and nondeterministic choices, with high probability the randomized SVN circuit has the same output as C_f. This yields a contradiction to our hardness hypothesis for f.

Finally, since for every $z \in \bar{D}^{-1}(1)$ it holds that $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$ and since all but at most $\frac{1}{2} \cdot N^{-\delta}$ outputs of G_0 are in $\bar{D}^{-1}(1)$, using the same calculation as in the proof of Theorem 11.5.5 we deduce that G "fools" D with error $N^{-\delta}$. ■

11.6 The $O(n)$ Overhead is Optimal Under #NSETH

In this section we show that under a counting version of the Non-Deterministic Strong Exponential Time Hypothesis (NSETH), which is denoted #NSETH and is weaker than NSETH, for any polynomial function $T(n) = n^k$ (where $k \geq 1$) and $\varepsilon > 0$ it holds that $\mathsf{BPTIME}[T(n)] \not\subset \mathsf{DTIME}[T(n) \cdot n^{1-\varepsilon}]$. Hence, the derandomization in Theorem 11.1.2 is essentially optimal under #NSETH. (Needless to say, the derandomization is optimal under the stronger assumption NSETH.)

The same result has been noted in [Wil16b, Section 3.2] for the special case of $T(n) = n$, and here we adapt the proofs in [Wil16b] for the case of larger $T(n)$. Roughly speaking, we consider a problem called k-OV from fine-grained complexity. Under #NSETH, there is no nondeterministic machine that counts the number of satisfying assignments for a k-OV instance in time $n^{k-\varepsilon}$, for any $\varepsilon > 0$. [37] On the other hand, following [Wil16b], we show a Merlin-Arthuer (MA) algorithm that counts the number of satisfying assignments to a k-OV instance in time n^{k-1}. If $\mathsf{BPTIME}[n^{k-1}] \subseteq \mathsf{DTIME}[n^{k-\varepsilon}]$ for some $\varepsilon > 0$ then the above Merlin-Arther algorithm can be

[37] By "nondeterministically counting" we mean that for all nondeterministic guesses the algorithm either output $\perp$ or the correct number of satisfying assignments, and it must output the correct number for some nondeterministic guess.

derandomized to a nondeterministic machine that counts satisfying assignments for k-OV in time $n^{k-\varepsilon}$, which contradicts #NSETH. [38]

11.6.1 #NSETH and k-OV

The Nondeterministic Strong Exponential Time Hypothesis (NSETH) was first introduced by Carmosino, Gao, Impagliazzo, Mihajlin, Paturi, and Schneider [CGI+16]. We now define a natural counting version of NSETH, denoted #NSETH. Denote by k-TAUT the language of all k-DNFs that are tautologies; then, #NSETH is the following:

Assumption 11.6.1 (#NSETH)**.** *For every $\varepsilon > 0$ there exists a k such that there does not exist a nondeterministic machine M that gets as input a k-SAT formula Φ over n variables, runs in time $2^{(1-\varepsilon)\cdot n}$ and satisfies the following:*

1. *There exists nondeterministic choices such that M outputs the number of satisfying assignments for Φ.*
2. *For all nondeterministic choices, M either outputs the number of satisfying assignments for Φ or outputs $\perp$.*

Note that #NSETH is weaker than NSETH (since a nondeterministic machine that refutes #NSETH also refutes NSETH). We now introduce the problem k-Orthogonal Vectors (k-OV), and recall the well-known fact that under NSETH it holds that $k\text{-OV} \notin \mathsf{coNTIME}[n^{k-\varepsilon}]$, for every $k \in \mathbb{N}$ and $\varepsilon > 0$ (*i.e.*, there does not exist a nondeterministic machine that rejects every no instance for some nondeterministic guess, and accepts every yes instance for all nondeterministic guesses).

Definition 11.6.2 (k-OV)**.** *We define $k\text{-OV}_{n,d}$ to be the following promise problem. An input to the problem consists of k sets $A_1, A_2, \ldots, A_k$, where each A_i is a set of n vectors from $\{0,1\}^d$. We say that a tuple $(a_1, a_2, \ldots, a_k) \in A_1 \times A_2 \times \cdots \times A_k$ is a* satisfying assignment *if*

$$\sum_{j=1}^{d} \prod_{i=1}^{k} (a_i)_j = 0 \,. \tag{11.2}$$

An input Φ to $k\text{-OV}_{n,d}$ is a yes instance if there exists a satisfying assignment for Φ.

[38] This derandomization would be immediate under a hypothesis that refers to promise-problems (*i.e.*, under the hypothesis $\mathsf{prBPTIME}[n^{k-1}] \subseteq \mathsf{prDTIME}[n^{k-\varepsilon}]$), and we show that it can also be done under a weaker hypothesis that refers only to languages.

Lemma 11.6.3 (fine-grained hardness of k-OV under #NSETH)**.** *For any integer $k \geq 2$, assuming* #NSETH*, there is no nondeterministic machine that counts the number of satisfying assignment for a given k-OV$_{n,d}$ instance in $n^{k-\varepsilon}$ time, for any $\varepsilon > 0$ and $d = \log^2 n$.*

For proof of Lemma 11.6.3, see the standard reduction from k-SAT to k-OV [Wil05]. (The standard reduction actually yields dimension $d = O_k(\log(n))$, but for simplicity we bounded the dimension by $d \leq \log^2(n)$.)

11.6.2 Proof of Theorem 11.6.4

Now we ready to prove the following main theorem of this section.

Theorem 11.6.4 (derandomization has a multiplicative overhead of n, under #NSETH)**.** *Assume that* #NSETH *holds. Then, for every $k \geq 1$ and $\varepsilon > 0$ it holds that* $\mathsf{BPTIME}[n^k] \not\subseteq \mathsf{DTIME}[n^{k+1-\varepsilon}]$.

Proof. For $k \geq 1$ and $\varepsilon > 0$, assume towards a contradiction that $\mathsf{BPTIME}[n^k] \subseteq \mathsf{DTIME}[n^{k+1-\varepsilon}]$. We show that in this case, for some $\varepsilon' > 0$ and for $d(n) = \log^2(n)$, there exists a machine that nondeterministically computes the number of satisfying assignments of a given $(k+1)$-OV$_{n,d}$ instance in time $n^{k+1-\varepsilon'}$. Relying on Lemma 11.6.3, this contradicts #NSETH.

Given a $(k+1)$-OV$_{n,d}$ instance $A_1, A_2, \ldots, A_{k+1} \in (\{0,1\}^d)^n$, the machine guesses and verifies a prime p in $(n^{2(k+1)}, 2 \cdot n^{2(k+1)}] \cap \mathbb{N}$. Then, the machine constructs $(k+1)$ degree-n polynomial mappings $P^i : \mathbb{F}_p \to \mathbb{F}_p^d$ (*i.e.*, for each $\ell \in [d]$, the ℓ-th coordinate of P^i is a degree-n polynomial) such that for every $i, \ell \in [k+1] \times [d]$ we have that $P^i(j)_\ell = (A_{i,j})_\ell$ for all $j \in [n]$ (where $A_{i,j}$ is the j^{th} vector in A_i). Each of the $(k+1) \cdot d$ corresponding polynomials can be constructed in time $\tilde{O}(n)$, using interpolation via FFT, and therefore the $(k+1)$ polynomial mappings can be constructed in time $\tilde{O}(n)$.

Now, let $F : \{0,1\}^{(k+1)\cdot d} \to \{0,1\}$ be the function that treats its input as $(k+1)$ vectors from $\{0,1\}^d$, and outputs 1 if and noly if the vectors are a satisfying assignment for our given k-OV instance. We define a polynomial $X : \mathbb{F}_p^{(k+1)\cdot d} \to \mathbb{F}_p$ that is a low-degree extension of F (*i.e.*, F and X agree on all inputs in $\{0,1\}^{(k+1)\cdot d} \to \{0,1\}$). To do so, fix a degree-d polynomial $\Phi : \mathbb{F}_p \to \mathbb{F}_p$ such that $\Phi(0) = 1$ and $\Phi(i) = 0$ for all $i \in \{1, 2 \ldots, d\}$; note that a suitable Φ can be found in time $\mathrm{poly}(d, \log(p))$ by straightforward interpolation. Then, for $x_1, x_2, \ldots, x_{k+1} \in \mathbb{F}_p^d$, we define X as

$$X(x_1, x_2, \ldots, x_{k+1}) := \Phi\left(\sum_{\ell=1}^{d} \prod_{i=1}^{k+1} (x_i)_\ell\right).$$

Note that X can be computed in polynomial time (in its input length), since we can construct Φ in time $\mathrm{poly}(d, \log(p))$. Also, note that X has degree $d \cdot (k+1)$ and is an extension of F. Finally, we construct a single-variate polynomial $Q : \mathbb{F}_p \to \mathbb{F}_p$ as follows:

$$Q(j_{k+1}) = \sum_{(j_1, j_2, \ldots, j_k) \in [n]^k} X\left(P^1(j_1), P^2(j_2), \ldots, P^k(j_k), P^{k+1}(j_{k+1})\right).$$

Note that Q is an $O(n \log^2 n)$-degree polynomial that can be described by $O(n \cdot \mathrm{polylog}(n))$ bits and evaluated in time $n^k \cdot \mathrm{poly}(d, \log p)$.

Consider the following problem:

- Given a prime p in $(n^{2(k+1)}, 2 \cdot n^{2(k+1)}] \cap \mathbb{N}$, and an $O(n \log^2 n)$-degree single-variate polynomial H over $\mathbb{F}_p$, together with an k-$\mathsf{OV}_{n,\log^2 n}$ instance Φ, determine whether H is the same as the polynomial Q above constructed from Φ.

Clearly, the problem above is in $\mathsf{BPTIME}[n^k \cdot \mathrm{polylog}(n)]$[39], as one can sample a random element x from $\mathbb{F}_p$ and check whether $Q(x) = H(x)$ (we rely on the fact that the field size is larger than $> n^{2k}$, and that the degree of the polynomial $O(n \log^2 n)$). By our assumption (and a padding argument), it is also in $\mathsf{DTIME}[n^{k+1-\varepsilon/2}]$.

From that one can get an algorithm computing the number of satisfying assignments to $(k+1)$-$OV_{n,\log^2 n}$ nondeterministically as follows:

1. Guess and verify a prime p in $(n^{2(k+1)}, 2 \cdot n^{2(k+1)}] \cap \mathbb{N}$.
2. One guesses a polynomial H of $O(n \log^2 n)$ degree over $\mathbb{F}_p$.
3. In $n^{k+1-\varepsilon/2}$ time, one reject H if H is not the same polynomial as Q.
4. Otherwise, output $\sum_{i \in [n]} H(i)$.

Note that the number of satisfying assignment is at most n^k, and $p > n^{2(k+1)}$. Hence, assuming the polynomial H passed the test, we have $\sum_{i \in [n]} H(i) = \sum_{i \in [n]} Q(i)$ is the number of satisfying assignments to the given $(k+1)$-$\mathsf{OV}_{n,d}$ instance. Moreover, the above algorithm runs in nondeterministic $n^{k+1-\varepsilon'}$ time for some $\varepsilon' > 0$, so we obtain a contradiction to #NSETH. ■

[39] The problem comes with a promise that p is a prime. But such promise can be easily checked in $\mathrm{polylog}(n)$ time deterministically [AKS04]. Therefore, we can make it as a total function by always outputting 0 when the given p is not a prime.

11.7 The Nisan-Wigderson PRG with Small Output Length

In this section we prove Theorem 11.4.1; that is, we instantiate the NW PRG for a small output length (*i.e.*, when the hard function has truth-table of size N and the output length is N^{ε}), using appropriate modifications a-la [RRV02] to reduce the seed length.

11.7.1 Preliminaries

First we need the notion of *weak combinatorial designs*, which was introduced by Raz, Reingold, and Vadhan [RRV02].

Definition 11.7.1 (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 $\sum_{j<i} 2^{|S_i \cap S_j|} \leq (m-1) \cdot \rho$.*

We also need the following efficient algorithm for constructing weak designs with parameters that are suitable for us (*i.e.*, large intersections between sets and small universe size t), which is from [Tel18].

Lemma 11.7.2 (constructing weak designs; see [Tel18, Lemma 6.2])**.** *There exists an algorithm that gets as input $m \in \mathbb{N}$ and $\ell, \rho \in \mathbb{N}$ such that $\log(\rho) = (1-\alpha) \cdot \ell$, where $\alpha \in (0, 1/4)$, and satisfies the following. The algorithm runs in time $\mathrm{poly}(m) \cdot 2^{\ell}$ and outputs an (m, ℓ, t, ρ) weak design, where $t = \lceil (1+4\alpha) \cdot \ell \rceil$.*[40]

We also need the following standard construction of error correcting codes, which is a concatenation of the Reed-Muller codes from Corollary 11.3.20 and Hadamard codes.

Lemma 11.7.3 (concatenating the RM code with the Hadamard code)**.** *There exists a universal constant $c \geq 1$ such that for every constant $\varepsilon \in (0,1)$ there exists a code $Enc : \{0,1\}^m \to \{0,1\}^{m^{1+c\sqrt{\varepsilon}}}$ satisfying:*

- *The code is computable in time $O(m^{1+c\sqrt{\varepsilon}})$.*
- *The code is locally list-decodable from agreement $1/2 + m^{-\varepsilon}$ with decoding time $m^{c \cdot \sqrt{\varepsilon}}$ and with list size $2^{m^{o(1)}}$.*

Proof. Let $\tau = \Theta(\varepsilon)$ and $\eta = \eta(\tau)$ be two constants to be specified later, and let $\rho = m^{-\tau}$. We use the t-variate Reed-Muller code of degree $d = m^{\eta}$ over $\mathbb{F}_q$, where $t = 1/\eta$ and $q = (8/\rho^2) \cdot d$.

[40] The running time is stated as $\mathrm{poly}(m, 2^{\ell})$ in [Tel18], but it is easy to see that the algorithm runs in $\mathrm{poly}(m) \cdot 2^{\ell}$ time.

Note that the input length to the code (measured in bits) is $\binom{t+d}{d} \cdot \log(q) \geq (d/t)^t \cdot \log(q) \geq m$, and the output length (measured in elements in $\mathbb{F}_q$) is $\bar{m} = q^t = O_\eta(m/\rho^{2/\eta})$. Also, by Theorem 11.3.19, this code is locally list-decodable from agreement ρ with decoding circuit size $\mathrm{poly}(t, \log(q), d, 1/\rho) = m^{O(\eta+\tau)}$ and output list size $O(1/\rho)$.

Now, let $Had : \mathbb{F}_q \to \{0,1\}^q$ be the Hadamard code. Recall that for every $\mu > 0$, Hadamard code is $(\frac{1}{2} + \mu, \frac{4}{\mu^2})$-list-docodable in $\mathrm{poly}(\mu^{-1}, \log q)$ time [GL89]. Concatenating the aforementioned Reed-Muller code with the Hadamard code, we obtain a Boolean code Enc with output length $O_\eta(m/\rho^{2/\eta} \cdot q) = O_\eta(m^{1+2\tau/\eta+\eta+2\tau})$.

Our goal now is to prove that Enc is locally list-decodable as in the statement; the analysis is standard, and we include it for completeness. We set μ so that $\rho = \frac{\mu^3}{4}$. The local decoding algorithm for Enc takes two indexes $i_1 \in [O(1/\rho)], i_2 \in [4/\mu^2]$, and simulates the RM local decoding algorithm with index i_1; whenever the latter algorithm needs to query one coordinate, it decodes that coordinate using the local decoding algorithm for the Hadamard code with index i_2. Note that the decoding time is $\mathrm{poly}(\mu^{-1}, \log q) \cdot m^{O(\eta)}$ and the output list size is $O(\rho^{-1} \cdot \mu^{-2})$.

Now we show that given a string y that is $1/2 + 2\mu$ correlated with $Enc(x)$, there exist some indexes $i_1 \in [O(1/\rho)], i_2 \in [4/\mu^2]$ such that the above algorithm successfully decodes x. Note that there are at least μ fraction of blocks such that within each of these blocks it holds that y is at least $1/2 + \mu$ correlated with $Enc(x)$ (a block is a consecutive segment in the output of Enc which corresponds to a single coordinate of the codeword of the RM code); this holds because otherwise we have that y is at most $\mu \cdot 1 + (1-\mu) \cdot (1/2 + \mu) < 1/2 + 2\mu$ correlated with $Enc(x)$, violating our assumption. Since for each block there exists one element from $[4/\mu^2]$ that causes the Hadamard decoder to correctly decode the block, by an averaging argument there exists an $i_2 \in [4/\mu^2]$ such that the Hadamard decoder correctly decodes at least a $\mu^3/4 = \rho$ fraction of the blocks. Hence, by the property of the RM decoder, there exists some index i_1 such that the overall decoder for Enc is successful.

Finally, let us analyze the parameters of Enc. We want to have $\mu = \frac{1}{2} \cdot m^{-\varepsilon}$, and so we set τ such that $\rho = \Theta(m^{-3\varepsilon})$ (recall that $\rho = \mu^3/4$). The decoding circuit size is thus $m^{O(\eta+\varepsilon)}$, the output list size is $m^{O(\varepsilon)}$, and the output length of the code is $O_\eta(m^{1+6\varepsilon/\eta+\eta+6\varepsilon})$. Then, the statement follows by setting $\eta = \sqrt{\varepsilon}$. ■

11.7.2 Proof of Theorem 11.4.1

Now we are ready to prove Theorem 11.4.1 (restated below for convenience).

Reminder of Theorem 11.4.1. *There exists a universal constant c such that for all sufficiently small ε, there exists an oracle machine G satisfies the following:*

- *When given input 1^{N^ε} and oracle access to a function $f\colon \{0,1\}^{\log(N)} \to \{0,1\}$, the machine G runs in time $N^{1+c\cdot\sqrt{\varepsilon}}$ and outputs $2^{\ell(N)}$ strings in $\{0,1\}^{N^\varepsilon}$, where $\ell_G(N) = (1+c\sqrt{\varepsilon})\cdot\log N$.*
- *There exists an oracle machine R that, when given input $x \in \{0,1\}^{\log(N)}$ and oracle access to an $(N^{-\varepsilon})$-distinguisher for $G(1^N, \mathbf{u}_{\ell(N)})^f$ and $N^{1-\sqrt{\varepsilon}/c}$ bits of advice, runs in time $N^{c\cdot\sqrt{\varepsilon}}$ and outputs $f(x)$.*

Proof. Let c_1 be the constant c from Lemma 11.7.3.

The oracle machine G. We first describe the construction of the oracle machine G. We instantiate Lemma 11.7.3 with parameter $\varepsilon_{RM} \in (2\varepsilon, 3\varepsilon)$ such that $N^{-\varepsilon_{RM}} = N^{-2\varepsilon}/10$. Let $Enc : \{0,1\}^N \to \{0,1\}^{N^{1+\beta}}$ be the corresponding encoder, where $\beta = c_1 \cdot \sqrt{\varepsilon_{RM}}$. We identify $f\colon \{0,1\}^{\log(N)} \to \{0,1\}$ with its truth-table of length N in the natural way, and let $\bar{f}$ be the function whose truth-table is $Enc(f)$. Note that $|\bar{f}| = N^{1+\beta}$.

We also apply Lemma 11.7.2 with $m = N^\varepsilon$ and $\ell = \log|\bar{f}| = (1+\beta)\cdot\log N$ and ρ such that $\log(\rho) = (1-3\beta)\cdot\ell$ (the hypothesis in Lemma 11.7.2 is that $3\beta < 1/4$, which holds since ε is sufficiently small). This yields an (m,ℓ,t,ρ) weak design $\{S_i\}_{i\in[m]}$, where $t = (1+12\beta)\cdot\ell$, that can be computed in time $\mathrm{poly}(m)\cdot 2^\ell = N^{1+O(\beta)}$.

Now, let $\ell_G = \ell_G(N) = t = (1+12\beta)\cdot(1+\beta)\cdot\log N = (1+O(\beta))\cdot\log N$. The machine G computes the truth-table of f, computes the encoding $\bar{f} = Enc(f)$, computes the design, and for every $w \in \{0,1\}^{\ell_G}$ outputs the N^ε-bit string such that $G(1^N, w)_i^f = \bar{f}(w_{|S_i})$ for $i \in [N^\varepsilon]$.[41] Note that given oracle access to f, the time that it takes to compute G for all w is bounded by the time that it takes to compute $\bar{f}$, plus the time it takes to compute the design, plus the time that it takes to print each output (given the design and access to f); this is at most $O\left(N^{1+O(\beta)} + 2^\ell \cdot N^\varepsilon\right) = N^{1+O(\beta)}$.

Construction of the oracle machine R. We introduce some useful notation. For two strings $\alpha, \beta \in \{0,1\}^*$, we denote by $\alpha \circ \beta$ the concatenation of α and β. For an integer $\ell \leq |\alpha|$, we denote by $\alpha_{\leq\ell}$ the length-ℓ prefix of α. For $x \in \{0,1\}^\ell$ and $w \in \{0,1\}^{t-\ell}$ and $S \subseteq [t]$ of size $|S| = \ell$, we denote by $x \circ_S w$ the string that is obtained by fixing the bits in the set S_i to x, and the bits in the set $[t] \setminus S_i$ to w.

[41] For a string $w \in \{0,1\}^t$ and a set $S \subseteq [t]$, we use $w_{|S} \in \{0,1\}^{|S|}$ to denote the projection of w onto the coordinates in S. That is, for all $i \in [|S|]$, $(w_{|S})_i = w_{S_i}$, where S_i is the i-th smallest element in S.

A standard hybrid argument (see, *e.g.*, [RRV02], following [NW94]) shows that there exists an oracle machine P that computes $\bar{f}$ with advantage $N^{-2\varepsilon}/10$ over a random guess when given access to an $(N^{-\varepsilon})$-distinguisher D for $G(1^N, \mathbf{u}_{\ell_G})^f$. In more detail, for any $N^{-\varepsilon}$-distinguisher D there exists an index $i \in [m]$, a suffix $z \in \{0,1\}^{m-i+1}$ and a string $w \in \{0,1\}^{t-\ell}$ such that

$$P(x) \stackrel{\text{def}}{=\!=} D(G(1^N, x \circ_{S_i} w)_{\leq i-1} \circ z) \oplus z_1$$

correctly computes $\bar{f}$ on at least a $1/2 + N^{-2\varepsilon}/10$ fraction of the ℓ-bit inputs.

Claim 11.7.4. *The function P can be computed in time $O(N^{2\varepsilon})$ with a single oracle to D and with $O(N^{1-\Omega(\beta)})$ bits of non-uniform advice.*

Proof. We give (i, z, w) as advice to P. For each $j < i$, note that $G(1^N, x \circ_{S_i} w)_j$ only depends on $|S_i \cap S_j|$ bits of x, hence the its whole truth-table can be stored with $2^{|S_i \cap S_j|}$ bits. By the definition of weak-designs, we have $\sum_{j<i} 2^{|S_i \cap S_j|} < N^{\varepsilon} \cdot \rho = O(N^{1-\Omega(\beta)})$, so we can store the truth-tables of $G(1^N, x \circ_{S_i} w)_j$ for all $j \in [i-1]$. We also give the sets $\{S_i\}_{i \in [m]}$ as advice to P, which takes $O(m \cdot \ell) = O(N^{\varepsilon} \cdot \log N)$ bits. So the overall number of advice bits is bounded by $O(N^{1-\Omega(\beta)})$.

Given $x \in \{0,1\}^{\ell}$, using the sets $\{S_i\}$ and the truth-tables for $\{G(1^N, x \circ_{S_i} w)_j\}_{j<i}$ that are all given as advice, we can compute $\alpha = G(1^N, x \circ_{S_i} w)_{\leq i-1}$ in time $O(N^{2\varepsilon})$. Then we can output $P(x) = D(\alpha \circ z) \oplus z_1$ by a single oracle call to D and linear-time processing involving the advice z. □

Note that P computes a "corrupt" version of $\bar{f}$ (*i.e.*, computes $\bar{f}$ correctly on $1/2 + N^{-2\varepsilon}/10$ of inputs). To compute f we run the local decoder for $\bar{f}$ with oracle access to P. By Lemma 11.7.3, the running time is bounded by $N^{O(\beta)}$ and the number of advice bits is bounded by $N^{1-\Omega(\beta)}$.

Finally, note that the local decoding circuits of Lemma 11.7.3 is randomized (see Definition 11.3.18) and succeeds with probability at least $2/3$ for each input to f. We first repeat the decoder $O(\log N)$ times to amplify the success probability to at least $1 - 1/N^2$. It then follows by a union bound that there is a fixed choice of randomness that makes the amplified decoder successfully compute f on all inputs. Adding such a fixed choice of randomness to the advice, we obtain an $N^{O(\beta)}$ time algorithm computing f exactly on all inputs, with $N^{1-\Omega(\beta)} + O(\log N \cdot N^{O(\beta)}) = N^{1-\Omega(\beta)}$ bits of advice and access to the distinguisher D, which completes the proof. ■

Chapter 12

Non-black-box and Superfast Derandomization from Uniform Hardness Assumptions

Contents

12.1 Introduction

One of the major achievements in complexity theory is the connection between pseudorandomness and lower bounds, which is known as the hardness-to-randomness framework. In a sequence of seminal works following [Yao82, BM84, Nis91, NW94, IW97], derandomization algorithms for *promise*-BPP (denoted prBPP, in short) were constructed under the assumption that there is a

function in $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ that is hard for *non-uniform circuits*. These derandomization algorithms work in a "black-box" fashion, by enumerating the seeds of a PRG and evaluating the relevant probabilistic algorithm on the resulting set. Since efficiently-computable PRGs (*i.e.*, that run in time exponential in the seed length) are well-known to imply circuit lower bounds for E, the bottom line of this line-of-works is that efficient black-box derandomization of prBPP is essentially equivalent to circuit lower bounds for E.

It is a classical open question whether or not the foregoing black-box approach is *necessary* for derandomization of prBPP. A natural suspicion is that this approach might be an "overkill", since the derandomization of a probabilistic machine M on input x does not depend on the way M operates on x (other than its number of steps), and just evaluates M on x using the output-set of a PRG as randomness. In the case of nondeterministic derandomization such as $\mathsf{prBPP} \subseteq \mathsf{prNP}$, we do know that this black-box approach is necessary (see, *e.g.*, [IKW02, Wil13a, MW18, Che19, CW19, CR20, CLW20]). However, for deterministic derandomization such as $\mathsf{prBPP} = \mathsf{prP}$ there has been essentially no unconditional progress on the question of whether black-box derandomization is necessary in the last three decades (see, *e.g.*, [CRTY20, Tel19] for conditional results and further discussion).

In this work we show that *the hardness-to-randomness framework can be revised* into a form that completely avoids the foregoing question. We prove a new and general hardness-to-randomness tradeoff that closely relates *uniform hardness assumptions*, of a particular type that we introduce, to *non-black-box derandomization of* prBPP. That is, under the foregoing uniform hardness assumptions we show how to deduce strong derandomization conclusions such as $\mathsf{prBPP} = \mathsf{prP}$, and we complement this result by showing that similar hardness assumptions are *necessary* for the derandomization conclusion. Thus, our approach suggests an appealing path towards proving an *equivalence* between *any derandomization* of prBPP (*i.e.*, not necessarily a black-box derandomization) and corresponding *uniform lower bounds* of the foregoing type.

In addition, mirroring classical hardness-to-randomness results, the new approach is general enough to allow trading off the hypothesis for the conclusion both in terms of running time and in terms of other structural restrictions on the probabilistic algorithms (*e.g.*, we show hardness-to-randomness tradeoffs for algorithms that work in parallel). In several settings that are obtained by such "scaling", our results already come close to proving a full equivalence between the hypothesized uniform lower bound and the derandomization conclusion.

The general form of our hardness hypotheses. Generally speaking, the uniform hardness hypotheses that we will use towards derandomization of prBPP are of the following form:

> There exists a multi-output function $f\colon \{0,1\}^n \to \{0,1\}^{k(n)}$ that can be computed by a deterministic algorithm that runs in time $T(n)^{O(1)}$ and satisfies an additional efficiency requirement, but cannot be computed by probabilistic algorithms in time $T(n)$.

The precise technical meanings of "additional efficiency requirement" and of "cannot be computed" vary across our particular results below. However, we stress that this form of hypothesis *does not assume* that the function f is hard for non-uniform circuits, or that the string $f(x)$ (for some input $x \in \{0,1\}^n$) is a truth-table of a function that is hard for non-uniform circuits. What we assume is simply that *given x, it is hard to print the string $f(x)$* in probabilistic time $T(|x|)$. This hardness assumption does not refer to circuit complexity at all.[1]

Our work in context: Uniform hardness-to-randomness. Numerous previous works deduced *some* derandomization conclusion from assumptions asserting that an efficiently-computable function is hard for uniform probabilistic algorithms.[2] However, *the conclusions that we deduce from the assumptions in this work are considerably stronger*. In more detail, the line-of-works following Impagliazzo and Wigderson [IW98] uses standard uniform hardness assumptions (such as $\mathsf{BPP} \neq \mathsf{EXP}$) to construct PRGs that derandomize BPP in the *average-case*. (In fact, these works typically show that their assumptions are equivalent to average-case derandomization.) Besides working only "on average", the derandomization typically also works only on infinitely-many input lengths, and is relatively slow, even when the hypothesized lower bound is strong (see [CIS18, CRTY20] for recent results and for discussions of previous results).

We use our new uniform hardness assumptions to construct derandomization algorithms that work in the *worst-case, on almost all input lengths, and can work in polynomial time* (or, in another result, to construct an extremely fast average-case derandomization that was not previously known to follow from uniform assumptions). Thus, our results strike a better balance than previous hardness-to-randomness tradeoffs: In contrast to the average-case results above, we deduce strong conclusions such as $\mathsf{prBPP} = \mathsf{prP}$, which demonstrates that *our assumptions are not "too weak"*; and

[1] Our notion of hardness is reminiscent of a lower bound on the conditional randomized time-bounded Kolmogorov complexity $\mathtt{rK}^T(f(x)|x)$ of $f(x)$; see [Oli19, LOS21, OL21, LP22] for recent studies of closely related notions.

[2] We note that hardness for *nondeterministic* probabilistic algorithms (*i.e.*, for MA) is a significantly stronger assumption, which is in fact sufficiently strong to imply the non-uniform hardness assumptions needed for worst-case derandomization using PRGs (this follows from known Karp-Lipton-style theorems such as [BFNW93]). We thus limit our comparison to results that use hardness assumptions for standard probabilistic algorithms.

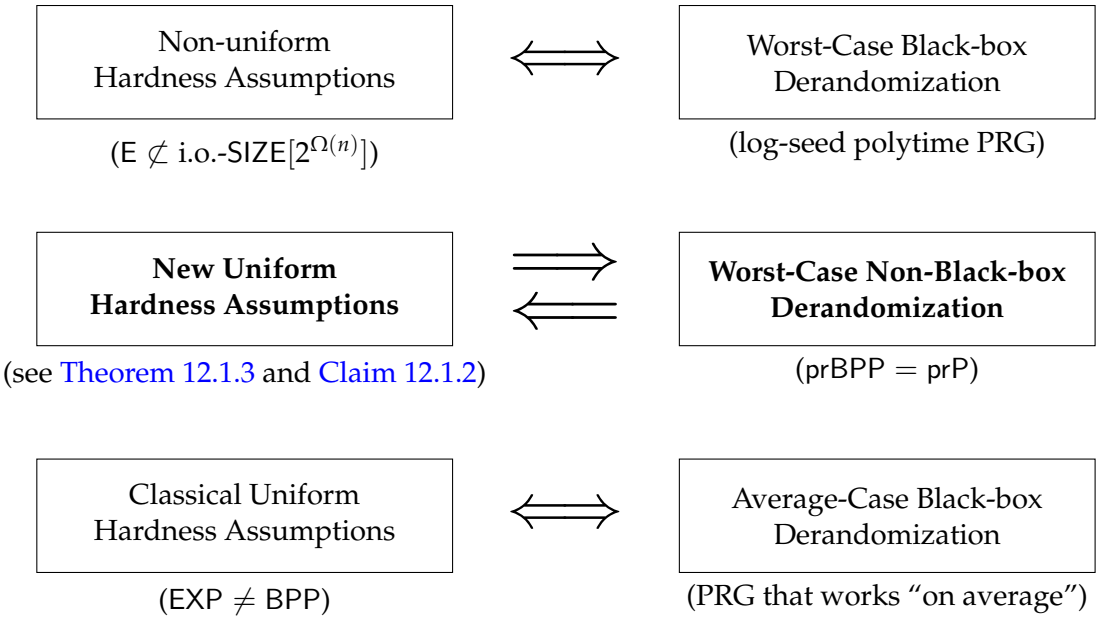


Figure 12-1: Classical hardness-to-randomness results compared with our new results for worst-case derandomization. For concreteness, below each box we mention in parentheses a specific parameter setting of the general result. The top row was proved by [IW97] (a smooth parametric scaling was subsequently shown in [SU05, Uma03]), the bottom row was proved by [IW98] (non-smooth parametric scalings were shown in subsequent works such as [TV07, CRTY20]), and the middle row corresponds to the results in Section 12.1.1.

in contrast to results that rely on non-uniform hardness assumptions, we prove that uniform hardness assumptions similar to the ones that we use are *necessary for derandomization*, suggesting that our hardness assumptions might not be "too strong". Indeed, the reason that we are able to strike this better balance is since our derandomization algorithms work in a non-black-box fashion. See Section 12.1 for a visual comparison with known results.

12.1.1 Derandomization from Hardness on Almost All Inputs

Our first main result is motivated by the following observation. Recall that if $\mathsf{prBPP} = \mathsf{prP}$, then for every $c > 1$ there exists $f \in \mathsf{P}$ that is hard for $\mathsf{BPTIME}[n^c]$ on almost all input lengths. (This is because we can derandomize $\mathsf{BPTIME}[n^c] \subseteq \mathsf{DTIME}[n^{O(c)}]$ and then appeal to a standard time-hierarchy theorem). The observation is that if we take f to have multiple outputs, then we can deduce that every probabilistic n^c-time algorithm fails to compute f *on almost all inputs*, rather than only on some input for every large enough input length; that is, for every probabilistic n^c-time algorithm there exists $n_0 \in \mathbb{N}$ such that the probabilistic algorithm fails to compute f on *each and every input* $x \in \{0,1\}^*$ *of length* $|x| \geq n_0$.

Definition 12.1.1 (probabilistically computing a function with multiple output bits)**.** *We say that a probabilistic algorithm A* computes a function *$f\colon \{0,1\}^n \to \{0,1\}^n$* at input *$x \in \{0,1\}^n$ if $\Pr[A(x) = f(x)] \geq 2/3$, where the probability is taken over the random coins of A.*

Claim 12.1.2 (almost-all-inputs hardness is necessary for derandomization)**.** *If* $\mathsf{prBPP} = \mathsf{prP}$*, then for every $c \in \mathbb{N}$ there exists $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in deterministic polynomial time such that f cannot be computed in probabilistic time n^c on* almost all inputs*.*

The proof of Claim 12.1.2 amounts to diagonalizing against every probabilistic machine that runs in time n^c on almost every input, by using each output bit of f to diagonalize against a different machine (and relying on the derandomization hypothesis); see Claim 12.5.2 for details.[3]

Our main result is that the existence of f as in conclusion of Claim 12.1.2 in fact *suffices* to deduce that $\mathsf{prBPP} = \mathsf{prP}$, assuming one additional structural restriction. Specifically, to deduce that $\mathsf{prBPP} = \mathsf{prP}$ we need the "almost-all-inputs" hard function f to be computable not just in P, but also by uniform circuits of depth that is noticeably smaller than their size. For example, it suffices to assume that f is computable by circuits of depth n^2:

Theorem 12.1.3 (polynomial-time non-black-box derandomization of prBPP)**.** *Assume that there exists $f\colon \{0,1\}^n \to \{0,1\}^n$ computable by deterministic logspace-uniform circuits of polynomial size and depth n^2 such that f cannot be computed in probabilistic time n^c on* almost all inputs*, where $c \in \mathbb{N}$ is a sufficiently large universal constant. Then* $\mathsf{prBPP} = \mathsf{prP}$*.*

We can replace the depth n^2 in the hypothesis of Theorem 12.1.3 by an arbitrary fixed depth n^k, at the expense of increasing the constant $c = c_k$ (setting $c > k + O(1)$ suffices; see Corollary 12.5.7).

We stress that the notion of uniformity in Theorem 12.1.3 (*i.e.*, logspace-uniformity) is meaningful only due to the depth constraint,[4] and is considerably weaker than standard notions of uniformity for circuits (such as DLOGTIME-uniformity). Thus, the main gap between the hypothesis that suffices to deduce that $\mathsf{prBPP} = \mathsf{prP}$ and the hypothesis that is necessary for this conclusion is that in the former the "almost-all-inputs" hard function can be computed in parallel (*e.g.*, by $\mathrm{poly}(n)$ processors running in parallel in time n^2).

[3] Indeed, it is not necessary for f to have precisely n output bits, and our main result also does not depend on f having precisely n output bits. We chose this output length for simplicity of presentation.

[4] This is since the classical transformations of uniform Turing machines to uniform circuits yield circuits that are logspace-uniform (and in fact even "more uniform", *e.g.* DPOLYLOGTIME-uniform).

Extensions: Scaling the new hardness-to-randomness result

Recall that classical hardness-to-randomness results extend both to derandomization of restricted complexity classes (such as $\mathsf{prBP} \cdot \mathsf{NC}$), and to "low-end" results, which are tradeoffs between weaker lower bounds and slower derandomization. We now state several extensions of Theorem 12.1.3, demonstrating that:

1. The hardness-to-randomness result in Theorem 12.1.3 *scales quite well*, both to restricted circuit classes and to weaker parameter settings.
2. When scaling the result in *either* of the foregoing two directions, the gap between the required lower bound and the necessary one *considerably narrows*.

As a first extension, we show that quantitatively weaker "almost-all-inputs" lower bounds imply slower derandomization. The tradeoff we obtain is very smooth when deducing derandomization of prRP, and less smooth when deducing derandomization of prBPP; specifically, the following result is a more general version of Theorem 12.1.3:

Theorem 12.1.4 (non-black-box derandomization of prBPP; a general version of Theorem 12.1.3)**.** *There exists a universal constant $c > 1$ such that the following holds. Assume that there exists a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable by deterministic logspace-uniform circuits of size $T(n)$ and depth $d(n)$ that cannot be computed in probabilistic time $d(n) \cdot n^c$ on* almost all inputs*. Then, we have that*

$$\mathsf{prRP} \subseteq \bigcup_{a \geq 1} \mathsf{prDTIME}[\bar{T}(n^a)]$$

$$\mathsf{prBPP} \subseteq \bigcup_{a \geq 1} \mathsf{prDTIME}[\bar{T}(\bar{T}(n^{c \cdot a})^a)] \,,$$

where $\bar{T}(m) = 2^{c \cdot \log^2(T(m))/\log(m)}$.

We stress that the meaning of logspace-uniform circuits of size T in Theorem 12.1.4 is that there exists uniform algorithm that prints a description of circuit using space $O(\log(T))$ (see Definition 12.3.5 and the subsequent comment). To see that Theorem 12.1.4 is a more general version of Theorem 12.1.3, note that when T is a polynomial then $\bar{T}(n^a)$ and $\bar{T}(\bar{T}(n^{c \cdot a})^a)$ are also polynomials. Similarly, as another example, note that when T is quasi-polynomial then $\bar{T}(n^a)$ and $\bar{T}(\bar{T}(n^{c \cdot a})^a)$ are also quasi-polynomials. A natural instantiation of Theorem 12.1.4 is with $d(n) = \mathrm{poly}(n)$, in which case the hypothesized hardness is for probabilistic polynomial-time algorithms, but other instantiations are also useful; see Section 12.1.1 for details.

When scaling our tradeoff even further to an extreme "low-end" setting, we are able to *completely eliminate the structural restrictions* on f. Specifically, in order to derandomize prRP infinitely-often in fixed exponential-time 2^{n^c} for some $c \geq 1$ (indeed a very weak derandomization), we do not need to assume that the hard function is computable by uniform circuits of bounded depth.

Theorem 12.1.5 (non-black-box fixed-exponential-time derandomization of RP)**.** *Assume that there exists a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in time 2^{n^c}, where $c \geq 1$ is a constant, such that f cannot be computed by probabilistic polynomial-time algorithms infinitely-often on almost all inputs.*[5] *Then, for some constant $c' \geq 1$ it holds that* $\mathsf{prRP} \subseteq$ *i.o.*-$\mathsf{DTIME}[2^{n^{c'}}]$.

Indeed, Theorem 12.1.5 already comes close to a full equivalence between the hardness hypothesis and the derandomization conclusion; essentially, the only remaining gap is that we derandomize prRP rather than prBPP. See Theorem 12.5.17 for details.

We suspect that it is possible to improve the derandomization overhead in Theorem 12.1.4 (see Remark 12.5.5), but we did not optimize our construction towards this purpose. Instead, we optimized our construction to scale down to *weak circuit classes*, in which case both the hardness hypothesis and the derandomization conclusion scale down to the relevant circuit class. The weakest natural class for which the result holds is that of logspace-uniform NC circuits; that is, logspace-uniform circuits of polynomial size and polylogarithmic depth. Towards stating the results, recall that NC^i is the class of polynomial-sized circuits of depth $\log^i(n)$; then:

Theorem 12.1.6 (non-black-box derandomization of NC)**.** *There exists a universal constant $c > 1$ such that the following holds. Assume that for every sufficiently large $i \in \mathbb{N}$ there exists $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in logspace-uniform deterministic $\mathsf{NC}^{(.99 \cdot i)}$ that cannot be computed by logspace-uniform probabilistic $\mathsf{NC}^i[n^c]$ on* almost all inputs. *Then,* $\mathsf{logspace\text{-}uniform\text{-}prBP} \cdot \mathsf{NC} \subseteq \mathsf{logspace\text{-}uniform\text{-}prNC}$.

Note that, similarly to the "low-end" extensions above, the structural requirements on the hard function in Theorem 12.1.6 are relaxed. First, the logspace-uniformity of circuits for f is now called for, since we want the derandomization algorithm to be computable by such circuits. And secondly, the gap in depth (between the deterministic circuits for f and the probabilistic circuits for which f is hard) is now only polylogarithmic. See Corollary 12.5.15 for further details.

[5] The meaning of "infinitely-often hard on almost all inputs" is that there is an infinite set $S \subseteq \mathbb{N}$ of input lengths such that for every probabilistic algorithm and every sufficiently large $n \in S$ and every $x \in \{0,1\}^n$, the algorithm fails to compute f on x.

Robustness of our results: Relaxing the hypotheses

One may ask what happens if the hardness in (say) Theorem 12.1.4 holds not with respect to almost all inputs, but "only" with respect to some very dense set of inputs. In all of the results in this section, when one assumes hardness with respect to an (arbitrary) distribution $\mathbf{x}$ and some succeess bound $\mu > 0$, we can deduce derandomization of probabilistic algorithms that have one-sided error with respect to the precise same distribution $\mathbf{x}$ and with μ as the error parameter. (This is because our hardness-to-randomness tradeoffs are "instance-wise", and hold with respect to any fixed input x; we explain this in Section 12.1.3.)

Indeed, the tradeoff emphasized above between almost-all-inputs hardness and worst-case derandomization simply represents the extreme setting when $\mathbf{x}$ is supported on all inputs and $\mu = 0$ (in which case we reduce derandomization of probabilistic algorithms that have two-sided error to derandomization of probabilistic algorithms that have one-sided error).

Another type of tradeoff, which can be obtained when considering derandomization in super-polynomial time, allows to almost completely eliminate the depth constraint on f. Specifically, to deduce worst-case derandomization of prBPP in (say) quasipolynomial time, it suffices to assume that for some constant $\varepsilon > 0$, the function f is computable by deterministic logspace-uniform circuits of quasipolynomial size T and of depth $T(n)^{1-\varepsilon}$, but cannot be computed in probabilistic time $T(n)^{1-\varepsilon/4}$ on almost all inputs (see Corollary 12.5.8 for precise details). In comparison to Theorem 12.1.3, this lower bound is stricter in terms of time (intuitively, we need a function in time T that is hard for time $T^{.99}$), but poses almost no depth restriction on f.

Discussion: Derandomization vs "almost-all-inputs" lower bounds

The main implication of the results in this section is a new and tight connection between **worst-case derandomization** and **almost-all-inputs lower bounds** for probabilistic algorithms, which is the connection hinted at in Section 12.1. Our results demonstrate that this is a *general* connection, which manifests itself in different parametric settings as well as for classes of restricted algorithms, such as parallel algorithms. (Some technical elaborations on the two latter connections, beyond what was already stated above, appear in Theorem 12.5.17 and Corollary 12.5.15.)

We stress that this new connection is *interesting per-se*, and not only since it improves on classical hardness-to-randomness results (*i.e.*, not only since it manages to get worst-case derandomization without relying on PRGs and on the corresponding non-uniform lower bounds). First, this

connection highlights the importance of a natural problem in theoretical computer science that, as far as we are aware, received little attention so far (*i.e.*, almost-all-inputs lower bounds). Secondly, the connection sheds additional light on the central role of derandomization in complexity theory, by significantly refining its connections to questions of lower bounds.

And thirdly, this connection is a very natural one. Loosely speaking, it relates the statement "deterministic machines can efficiently compute a *hard function* for probabilistic machines" to the statement "deterministic machines can efficiently *simulate* probabilistic machines"; that is, we relate simulation to lower bounds for these classes of machines. This can be viewed as demonstrating a setting with a partial affirmative answer to a classical question asked by Kozen [Koz78]: Informally, he asked for which pairs of complexity classes is simulation equivalent to lower bounds (pointing out the implication that in this case diagonalization is a complete method for lower bounds).[6] The interesting and non-standard part in the answer suggested by our results is that the hardness refers to multi-output functions and to almost-all-inputs hardness.

12.1.2 Superfast Derandomization from Non-batch-computable Functions

Our second main result refers to *superfast derandomization* of BPP, meaning derandomization that has almost no time overhead. The question of the best possible polynomial time overhead was recently raised by Doron *et al.* [DMOZ20], who showed a conditional derandomization with near-quadratic overhead. In a follow-up work, Chen and Tell showed [CT21b] (among other results) a conditional average-case derandomization with *almost no overhead*: For every $\varepsilon > 0$, probabilistic algorithms that run in time T were simulated by deterministic algorithms that run in time $T \cdot n^{\varepsilon}$, on average-case over any T-time samplable distribution.

The hardness hypotheses in both previous works were very strong non-uniform lower bounds.[7] Our second main result is a new hardness-to-randomness approach for the setting of superfast derandomization that relies only on *appealing uniform hardness hypotheses*. As in Section 12.1.1, we will consider a multi-output function $f\colon \{0,1\}^n \to \{0,1\}^{k(n)}$ that is computable in deterministic time T but not in smaller probabilistic time T'; and since we now consider superfast derandomization, the gap between T and T' will now be very small. However, in the current setting our structural

[6] Needless to say, this is a very informal phrasing of his broad question, which can be interpreted in various different ways to begin with. For further study and some formalizations see [Koz78, NIR03, NIR06].

[7] Specifically, the assumption in [DMOZ20] was that there exists a problem in $\mathsf{DTIME}[2^n]$ that is hard for randomized nondeterministic circuits of size $\approx 2^{.99n}$. The assumptions in [CT21b] were of that non-uniformly secure one-way functions exist, and that there exists a problem in time $2^{k \cdot n}$ that cannot be solved in time $2^{.99k \cdot n}$ even with $2^{.99n}$ bits of non-uniform advice (where $k = k_{T,\varepsilon}$ is a suitable large constant).

requirement on f will be different: We will assume that *each individual output bit of* $f(x)$ can be computed faster than the time-complexity of *printing the entire string* $f(x)$.

Intuitively, we refer to such hard functions as non-batch-computable: This is since the hardness assumption implies that when printing all the bits of the string $f(x)$ in a batch, one has to invest noticeably larger running time than when printing just a single bit of $f(x)$.

The most appealing special case: A direct-product hypothesis

We first describe what we consider to be the most appealing special case of our result. Recall that standard *direct-product results* start with a function f_0 that is hard to compute in time T on (say) $1/3$ of the inputs, and prove that its k-wise direct product $f = f_0^{\times k}$ is hard to compute in time $T' \approx T$ on $1 - 2^{-\Omega(k)}$ of the inputs (see, *e.g.*, [IJKW10]).[8] Our starting point is that these results are also known to extend (unconditionally) to "approximate-direct-product" versions, showing that $f(z)$ cannot even be *approximately-printed* for the vast majority of z's: That is, every time-T' algorithm fails, for .99 of the tuples $z = (x_1, ..., x_k)$, to even output a string $\widetilde{f}(z)$ such that $\Pr_{i \in [k]}\left[\widetilde{f}(z)_i = f(z)_i\right] \geq .99$. See Proposition 12.6.11 for a precise statement, following [IJKW10].

A well-known conjecture is that in some cases f is harder also for algorithms with larger running time; taken to the extreme, the "strong direct-product" conjecture asserts that in some cases one cannot improve on the naive algorithm that runs in time $k \cdot T$ (see, *e.g.*, [Sha03]). Our hypothesis will be that a mildly-strong approximate-direct-product result holds for *some function* $f_0 \in \mathsf{DTIME}[T]$ that is hard for time $T \cdot n^{-\Omega(1)}$. Specifically, given a parameter $\delta > 0$, we assume that for every $T(n) = \mathrm{poly}(n)$ and $k(n) = n^{\Omega(1)}$ there exists $f_0 \in \mathsf{DTIME}[T]$ such that $f = f_0^{\times k}$ satisfies the following: Every probabilistic algorithm that gets input $z \in \{0,1\}^{n \cdot k(n)}$ and runs in time $T(n) \cdot k(n)^{\alpha}$ does not succeed in printing a string $\widetilde{f}(z)$ satisfying $\Pr_{i \in [k(n)]}\left[\widetilde{f}(z)_i = f(z)_i\right] \geq 1 - \alpha$ on more than a δ-fraction of z's, where $\alpha > 0$ can be an arbitrarily small constant (see Assumption 12.6.12 for a formal statement).

As in our previous work, to deduce derandomization with essentially no time overhead we will need another assumption, albeit a completely standard one – the existence of one-way functions. Assuming that the mildly-strong approximate-direct-product hypothesis holds, and that one-way functions secure against uniform polynomial-time algorithms exist, we show that BPP can be simulated, on average, with almost no time overhead:

[8] Recall that $f = f_0^k$ is defined by $f(x_1, ..., x_k) = (f_0(x_1), ..., f_0(x_k))$.

Theorem 12.1.7 (superfast non-black-box derandomization from mildly-strong approximate-direct-product)**.** *Assume that one-way functions exist, and that for some $\delta > 0$ the mildly-strong approximate-direct-product hypothesis holds. Then, for every polynomial T we have that*[9]

$$\mathsf{BPTIME}[T] \subseteq \bigcap_{\varepsilon>0} \mathsf{avg}_{1-2\delta}\text{-}\mathsf{DTIME}[n^{\varepsilon} \cdot T] \ .$$

We view the hypothesis in Theorem 12.1.7 as quite mild considering the strong conclusion. First, as mentioned above, the existence of $f_0 \in \mathsf{DTIME}[T] \setminus \mathsf{BPTIME}[T \cdot n^{-\Omega(1)}]$ is necessary for superfast derandomization. Secondly, the mildly-strong approximate-direct-product hypothesis is a relaxed one: We only assume the existence of *one* suitable f_0 (rather than a direct-product result for a class of functions), and only assume that the hardness grows to $T \cdot k^{.0001}$ (rather than to $T \cdot k$). And thirdly, in contrast to [CT21b], the hypothesis in Theorem 12.1.7 is completely uniform, since even the one-way function only needs to be uniformly-secure (in [CT21b] we needed a one-way function secure against non-uniform circuits).

Randomness might be "indistinguishable from useless" for decision problems

The drawback of the concluded derandomization in Theorem 12.1.7 is that it succeeds only over the uniform distribution. Recall that in Chapter 11, assuming strong non-uniform lower bounds, the derandomization succeeded over all T-time samplable distributions.

In the following result we show that a natural strengthening of the uniform hypothesis in Theorem 12.1.7 allows to derandomize BPP with the same tiny time overhead over *all polynomial-time-samplable distributions* and with a *negligible average-case error*. Informally, an appealing interpretation of the latter conclusion is that randomness is "indistinguishable from being essentially useless" for solving decision problems in polynomial time (where "essentially useless" here means that it can save a factor of at most n^{ε} in the running time).

Towards stating the result, let us say that a probabilistic algorithm A approximately prints a function $f\colon \{0,1\}^n \to \{0,1\}^{k(n)}$ on input x if $\Pr[A(x)_i = f(x)_i] \geq .99$, where the probability is over $i \in [k(n)]$ and the internal randomness of A (note that the input x is fixed). Then:

Theorem 12.1.8 (superfast non-black-box derandomization over all polynomial-time-samplable distributions; see Theorem 12.6.6)**.** *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a polynomial. Assume that one-way functions*

[9] The notation $\mathsf{avg}_{1-2\delta}$- means that for every L there exists a deterministic algorithm that errs with probability at most 2δ over a uniformly-chosen input.

exist, and that for every $\varepsilon > 0$ there exist $\delta > 0$ and a function $f\colon \{0,1\}^n \to \{0,1\}^{n^\varepsilon}$ such that:

1. *There exists an algorithm that gets input $(x,i) \in \{0,1\}^n \times [n^\varepsilon]$ and outputs the i^{th} bit of $f(x)$ in time $T'(n)$, where $T'(n) = T(n) \cdot n^\varepsilon$.*
2. *For every probabilistic algorithm A that runs in time $T'(n) \cdot n^\delta$ and every polynomial-time samplable distribution $\mathbf{x}$, the probability over $x \sim \mathbf{x}$ that A approximately prints f on input x is negligible.*

Then $\mathsf{BPTIME}[T] \subseteq \bigcap_{\varepsilon>0} \mathsf{heur}_{1-\mathsf{negl}}\text{-}\mathsf{DTIME}[n^\varepsilon \cdot T]$, *where* negl *is a negligible function.*[10]

We note that the conclusion of Theorem 12.1.8 can *only be obtained using a non-black-box derandomization algorithm* (such as the one that we use); that is, any PRG-based approach cannot yield such a conclusion. See Remark 12.6.7 for an explanation.

As we explain below (see Section 12.1.3), all of our derandomization algorithms not only solve the decision problem, but also *find* random strings that lead the relevant probabilistic machine to a correct decision. This has the following implication to the question of *explicit constructions*.

We say that an algorithm A is an $\mathsf{explicit\ construction}$ of objects from a set $\Pi \subseteq \{0,1\}^*$ if $A(1^n)$ prints an n-bit string in Π. We are particularly interested in deterministic explicit constructions of random-looking objects, which corresponds to the case where Π is dense (*e.g.*, $|\Pi \cap \{0,1\}^n| \geq 2^n/2$). In this case, we show that under the same assumption as in Theorem 12.1.8, *the complexity of explicit constructions is nearly identical to that of verifying that the object has the specified property*; that is, loosely speaking, deterministically constructing good random-looking objects is never significantly more expensive than deciding if an object is good:

Corollary 12.1.9 (superfast explicit constructions)**.** *Suppose that the assumption of Theorem 12.1.8 holds for every polynomial T. Then, for every $\Pi \in \mathsf{BPTIME}[n^k]$ such that $|\Pi \cap \{0,1\}^n| \geq 2^n/n^{o(1)}$ and every $\varepsilon > 0$, there exists an explicit construction of objects from Π in deterministic time $n^{k+\varepsilon}$.*

Lastly, our results also allow to interpolate between the derandomization over the uniform distribution as in Theorem 12.1.7 and the derandomization over all polynomial-time-samplable distributions as in Theorem 12.1.8. Specifically, if a hypothesis similar to the one in Theorem 12.1.8 holds with respect to some fixed polynomial-time-samplable distribution $\mathbf{x}$, then the derandomization conclusion holds over this distribution $\mathbf{x}$ (see Theorem 12.6.4 for details).

[10] The notation $\mathsf{heur}_{1-\mathsf{negl}}$- means that for every L there exists a deterministic algorithm A_L such that for every polynomial-time-samplable distribution $\mathbf{x}$, the probability over $x \sim \mathbf{x}$ that A_L errs on x is negligible.

Necessity of non-batch-computable functions for superfast derandomization

We do not know if "non-batch-computable" functions as in the hypotheses of Theorem 12.1.7 and Theorem 12.1.8 are necessary for average-case superfast derandomization *in general*. However, we prove that a non-batch-computable function is necessary in one *natural special case*.

Specifically, consider balanced formulas of polynomial size that are DLOGTIME-uniform, which we refer to as well-structured formulas. Loosely speaking, we show that if probabilistic well-structured formulas can be derandomized with overhead $T \mapsto T \cdot n^{\varepsilon}$ on average over the uniform distribution, then the following holds: There exists a function $g\colon \{0,1\}^n \to \{0,1\}^{n^{\delta}}$ such that the mapping $(x,i) \mapsto g(x)_i$ is computable in time $\tilde{O}(T)$, but every probabilistic well-structured formula of size $o(T \cdot n^{\delta-\varepsilon})$ fails to print $g(x)$, with high probability over uniform choice of x. For precise details (and a slightly stronger statement) see Proposition 12.6.16.

12.1.3 Non-black-box Derandomization and Instance-wise Hardness

As mentioned above, our derandomization algorithms for all results above will not rely on the standard approach of enumerating the seeds of a PRG. Instead, our algorithms rely on what was defined by Goldreich [Gol11a] as targeted PRGs: These are PRGs that do not "fool" all circuits, but rather get an input x and fool all efficient uniform machines that also have access to x.[11] In other words, the potential distinguisher for the PRG is not modeled as a non-uniform circuit (as in the classical "textbook" approach, where the non-uniformity represents any possible fixed input), but rather by a uniform Turing machine that only gets access to the *particular fixed input* x that is also given to the PRG. Indeed, such targeted PRGs suffice for derandomization, and Goldreich in fact showed that their existence is also necessary for derandomization.

Recall that a complete problem for prBPP is the Circuit Acceptance Probability Problem (CAPP): In this problem we are given a circuit and want to approximate its acceptance probability, up to an additive error of (say) $1/10$. The classical PRG approach for solving this problem treats the given circuit as a black-box, instantiating the PRG using only the *circuit size* as information. In contrast, our targeted PRGs *treat the given circuit in a non-black-box way*, using the description of the circuit as information in order to generate pseudorandom strings that are good for this particular circuit; see further details in Section 12.2.

[11] In [Gol11a] and in a subsequent work [Gol11b] Goldreich proposed two definitions for targeted PRGs, which he called targeted canonical derandomizers (and targeted canonical hitters for the HSG version). We use the first of those definitions, which appears in [Gol11a, Definition 4.1].

Following classical techniques, our technical approach is *reconstructive*: We design a "reconstruction" algorithm R that transforms every distinguisher for the targeted PRG to an efficient procedure that computes the hard function. The innovative aspect in our reconstruction arguments is that they work instance-wise, for *each fixed input* x: That is, our targeted PRG maps every x to a collection S_x of pseudorandom strings, and our reconstruction algorithm R gets input x and access to a distinguisher for S_x, and prints the output of the hard function f at x (*i.e.*, it prints $f(x)$). Thus, the hardness guarantee for a particular fixed x implies pseudorandomness of S_x for distinguishers that get access to that particular fixed x; in other words, *our hardness-to-randomness tradeoff holds with respect to each particular input* x. Accordingly, when we assume almost-all-inputs hardness of f we get worst-case derandomization as in Section 12.1.1, and when we assume average-case hardness of f we get average-case derandomization as in Section 12.1.2.

12.1.4 Organization

In Section 12.2 we describe our proof techniques, in high level. Section 12.3 includes preliminary definitions and statements of well-known results. In Section 12.4 we present the technical results underlying the proofs of the results that were described in Section 12.1.1, and we prove the foregoing results in Section 12.5. Finally, in Section 12.6 we prove the results that were described in Section 12.1.2. The appendices include proofs of several technical results that are stated and used in this chapter.

12.2 Technical Overview

We first explain the ideas behind our new hardness-to-randomness approach, in very high-level, and relate them to previous works and known approaches. Then, in Section 12.2.1 and Section 12.2.2 we describe the proofs of our main results in more detail.

As mentioned in Section 12.1.3, the underlying setting is the following: We are given an input $x \in \{0,1\}^n$ and want to produce a collection of strings S_x that appear random to any distinguisher that is also given the same input x. At a bird's eye, our proof techniques merge the techniques from the two main lines-of-work concerning uniform hardness-to-randomness, which were separate so far: The line-of-works following Impagliazzo and Wigderson [IW98] (see, *e.g.*, [CNS99, Kab01, Lu01, GSTS03, TV07, SU07, GV08, CIS18, CRTY20]) and the line-of-works following Goldreich and Wigderson [GW02] (see, *e.g.*, [vMS05, Zim08, Sha10, Sha11, KvMS12, SW13, Alm19, Hoz19]).

The first main idea (extending [GW02]): Applying a hard function to the input. As a starting point, let us return to the original idea of Goldreich and Wigderson [GW02] – using the *information in the input* to produce good pseudorandom strings. In their original work they applied a randomness extractor $\mathtt{Ext}$ to the input x to obtain a list $\{\mathtt{Ext}(x,s)\}_s$ of pseudorandom strings. A later modification of their approach by Kinne, van Melkebeek and Shaltiel [KvMS12] applied a seed-extending pseudorandom generator G to obtain a pseudorandom string $G(x)$, using the input as a seed (see [KvMS12] for the definition of a seed-extending PRG).

These two approaches naturally give rise to the following general question: Which function f can we apply to the input x in order to obtain good pseudorandom strings? The answer presented in this chapter seems, in retrospect, to be the most straightforward one: *We apply a hard function f to the input x*, as the first step in our constructions of S_x.

As observed in [GW02], the main danger in using the input x as a source of randomness is the correlation between x and the set of random strings that lead the relevant probabilistic algorithm to a wrong decision on x. A key insight implicit in [Sha11, KvMS12] is that this correlation is computational (*i.e.*, can be recognized by an efficient algorithm), and thus we can avoid this danger by enforcing a suitable hardness requirement on S_x. Our use of a computationally-hard function $x \mapsto f(x)$ as the first step in constructing S_x can be viewed as following this idea.

The second main idea (extending [IW98]): A non-black-box uniform reconstruction argument. As mentioned in Section 12.1.3, we do not just output $f(x)$, but show efficient transformations of $f(x)$ to a set $S_x = S_{f,x}$ of pseudorandom strings such that any distinguisher D for S_x can be efficiently transformed to an algorithm F satisfying $F(x) = f(x)$. The mapping of x to $f(x)$ and then to the set S_x is the core of our targeted PRGs, and the transformation of a distinguisher for S_x to an algorithm F is called an *instance-wise reconstruction procedure*.

The main previously-known approach for designing uniform reconstruction procedures, introduced by [IW98, TV07], requires that the hard function will be downward self-reducible and randomly self-reducible, and only allows to deduce that the PRG works on average-case and infinitely-often (we explain this in Section 12.2.2). We show a "non-black-box" version of this argument that crucially relies on access to the input x, and that works whenever the hard function is computable by *logspace-uniform circuits of bounded depth* (*e.g.*, of size $T(n)$ and depth $\sqrt{T(n)}$). Indeed, this new version allows to deduce that the derandomization works in the worst-case and on almost all input lengths. Our reconstruction procedure is based on the ideas in the doubly-efficient

proof system of Goldwasser, Kalai, and Rothblum [GKR15].

12.2.1 Warm-up: Proofs of Theorem 12.1.7 and Theorem 12.1.8

Let us begin by proving Theorem 12.1.7 and Theorem 12.1.8. The proofs of these results are simpler, and showcase our ideas of applying a hard function to the input and of analyzing a targeted PRG by "instance-wise" reconstruction. Our goal now will be to derandomize probabilistic algorithms running in time $T(n) = \text{poly}(n)$ in deterministic time $n^{\varepsilon} \cdot T(n)$ on average, and for simplicity we consider derandomization over the uniform distribution.

The main idea. Let us first explain our idea while ignoring the precise parameters. We think of $f(x)$ as a hard truth-table that is produced from the input x, and use this truth-table to instantiate a suitable version of the reconstructive PRG of [NW94, IW97] (see below). As observed in [IW98], the reconstruction procedure for this PRG is a *uniform learning algorithm* that works when given access to a distinguisher for the PRG; that is, the reconstruction procedure gets access to a distinguisher D, issues a small number of queries to the function described by $f(x)$,[12] and outputs a small oracle circuit C such that C^D computes the function described by $f(x)$.

The main pressing question is how our reconstruction procedure can answer the queries of this learning algorithm without using any non-uniformity. This is where our assumption that the function $(x, i) \mapsto f(x)_i$ is efficiently-computable comes in. Specifically, assume that we can compute $(x, i) \mapsto f(x)_i$ in time $T'(n) = \text{poly}(n)$, but that printing all of $f(x)$ cannot be done in time much smaller than $|f(x)| \cdot T'(n)$. Then, our reconstruction argument relies on two parts:

1. **Non-black-box reconstruction:** The reconstruction procedure answers queries of the learning algorithm to $f(x)$ by computing the mapping $(x, i) \mapsto f(x)_i$ in time T', relying on the fact that *it has explicit access to the input* x. With an appropriate setting of parameters, we can ensure that the reconstruction procedure runs in time noticeably less than $|f(x)| \cdot T'(n)$. (Specifically, we set $|f(x)| = n^{\Omega(\varepsilon)}$ and have the PRG of [NW94] output only $|f(x)|^{\Omega(1)} = n^{\Omega(\varepsilon)}$ pseudorandom bits; see details below.)
2. **Learning a small circuit $\Rightarrow$ batch-computing the truth-table:** Given the small oracle circuit C that the learning algorithm outputs, our reconstruction procedure can quickly print the entire string $f(x)$, by evaluating C^D on all inputs. Specifically, it can do so in time $|f(x)| \cdot$

[12] These queries are used for the reconstruction procedure of the PRG of [NW94] as well as to eliminate the required advice in the list-decoding algorithms of [GL89, IW97], which are both used as part of the reconstruction procedure.

$\tilde{O}(|C|) \cdot T(n)$, which we can ensure is noticeably less than $|f(x)| \cdot T'(n)$, by defining T' to be slightly larger than T (see below). This contradicts the hardness of f on input x.

Assuming that $f(x)$ is hard to print in time $|f(x)| \cdot T'(n)$ on 0.99 of the inputs x, our derandomization succeeds on 0.99 of the inputs. In fact, for every polynomial-time samplable distribution $\mathbf{x}$, if $f(x)$ is hard to print with high probability over $x \sim \mathbf{x}$, then our derandomization succeeds with high probability over $x \sim \mathbf{x}$ (the requirement that $\mathbf{x}$ will be polynomial-time samplable is due to our use of one-way functions, which was not described above; we explain this in a moment). By tweaking the parameters, we can even relax the required lower bound to be of the form $|f(x)|^{\alpha} \cdot T'(n)$ for an arbitrarily small constant $\alpha > 0$, rather than $|f(x)| \cdot T'(n)$.

Unfortunately, the idea above does not work as-is. First, as mentioned above, to make this idea work we need the number of output bits of the PRG of [NW94, IW97] to be only $n^{\Omega(\varepsilon)}$ rather than $T(n)$; we will use our assumption that one-way functions exist to bridge this gap (this assumption implies the existence of a very fast PRG with an arbitrarily large polynomial stretch). Secondly, the PRG of [NW94, IW97] actually encodes the truth-table $f(x)$ by error-correcting codes and the reconstruction procedure needs query access to the encoded string rather than to $f(x)$; indeed, these encodings do not preserve the time-complexity of computing individual bits of the string. We explain below how to overcome this difficulty, using the stronger assumption that $f(x)$ is hard to "approximately-print".

Implementation: Some technical details. Recall that our goal is to derandomize a probabilistic algorithm that runs in time $T(n)$. Fix any small constant $\alpha > 0$ (which will characterize the hardness of f), let $\varepsilon' > 0$ be a sufficiently small constant, and let $k = n^{O(\varepsilon'/\alpha)} \ll n^{\alpha}$. For $T' = T \cdot k$, we fix a function $f\colon \{0,1\}^n \to \{0,1\}^k$ such that $(x,i) \mapsto f(x)_i$ is computable in time T', but every probabilistic algorithm that runs in time $T' \cdot k^{\alpha}$ fails, on all but a small fraction of inputs x, to print whp a string $\widetilde{f}(x)$ that agrees with $f(x)$ on more than 0.99 of the bits. Recall that such a function f is obtained naturally as the k-wise direct-product of a function $f_0 \in \mathsf{DTIME}[T']$ that is hard for probabilistic time slightly smaller than T' (see Section 12.6.2 for details).

As a first step, relying on the assumption that one-way functions exist, we use a PRG that runs in near-linear time to reduce the number of random coins used by the probabilistic algorithm from $T(n)$ to $n^{\varepsilon'}$, without significantly increasing its running time and while maintaining correctness on all but a negligible fraction of inputs (see Theorem 12.3.4, Theorem 12.6.4, and Claim 12.6.5 for details). For simplicity, let us assume that we reduced the number of random coins to $n^{\varepsilon'}$ without

affecting the running time or correctness at all.

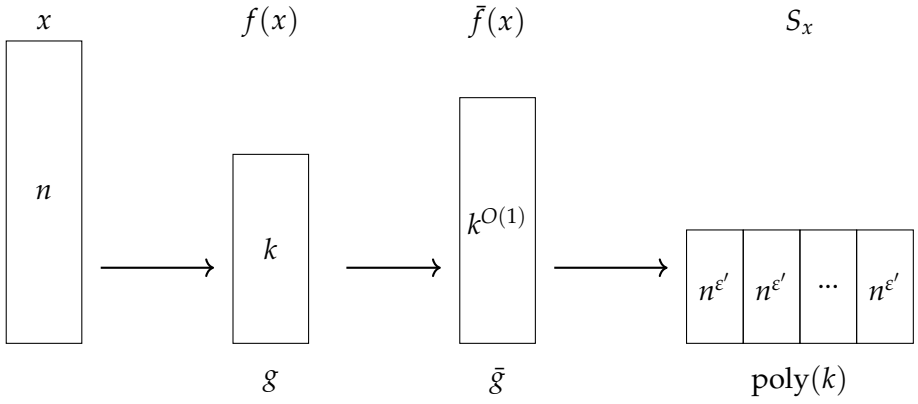
The targeted PRG. On input x we first compute the string $f(x)$, which we think of as the truth-table of a function $g\colon [k] \to \{0,1\}$. We then apply the "mild average-case" to "extreme average-case" hardness amplification step of [IW97] to the function g, and then apply the Hadamard encoding to the resulting non-Boolean function, to obtain the truth-table $\bar{f}(x)$ of a function $\bar{g}$. We instantiate the NW PRG with $\bar{g}$ as the hard function and with output length $n^{\varepsilon'}$, and this yields a collection $S_x \subseteq \{0,1\}^{n^{\varepsilon'}}$ of strings that we hope fools the probabilistic algorithm on input x.

The sequence of transformations $x \mapsto f(x) \mapsto \bar{f}(x) \mapsto S_x$ above is depicted visually in Section 12.2.1. One crucial point for us is that the first two transformations, mapping x to $\bar{f}(x)$, satisfy the following property: Each output bit of $\bar{f}(x)$ can be computed, when given access to x, in time $T' \cdot \log(k)$. The reason is that each output bit of $f(x)$ is computable from x in time T' by our assumption, and the truth-table $\bar{f}(x)$ is obtained from $f(x)$ by applying the efficient derandomized direct product of [IW98] and the Hadamard encoding, which approximately maintain the complexity of the underlying function (see Theorem 12.6.3 and Section 12.7.2 for details).

The resulting derandomization algorithm is extremely fast: We compute the string $f(x)$ in time $T' \cdot k$, transform it into $\bar{f}(x)$ and S_x in time $\mathrm{poly}(k)$, and evaluate the original probabilistic algorithm on each string in S_x (and output the majority value) in time $\mathrm{poly}(k) \cdot T' = \mathrm{poly}(k) \cdot T$. Choosing ε' to be sufficiently small, the running time is less than $n^{\varepsilon} \cdot T$.

Analysis: Non-black-box reconstruction. Our goal is to design a reconstruction procedure that, when given access to x and to a time-T distinguisher D for S_x, runs in time $T' \cdot n^{\alpha}$ and with high probability prints a string $\widetilde{f}(x)$ that agrees with $f(x)$ on more than 0.99 of the bits. Our procedure will use the reconstruction algorithm `Rec` of the NW PRG as well as the list-decoding algorithm `Dec` underlying the transformation of g to $\bar{g}$ (the latter combines the list-decoding algorithms of the derandomized direct-product of [IW97] and of the Hadamard encoding [GL89]).

Since the output length of the targeted PRG is small (*i.e.*, the output length is $n^{\varepsilon'}$), both `Rec` and `Dec` run in time at most $n^{O(\varepsilon')} \leq k^{\alpha/2}$ (see Theorem 12.6.3 for details). Whenever `Rec` queries $\bar{g} = \bar{f}(x)$ at location $i \in [\mathrm{poly}(k)]$, our reconstruction procedure uses its access to x in order to compute the mapping $(x,i) \mapsto \bar{f}(x)_i$ in time $T' \cdot \log(k)$. Also, the list-decoding algorithm `Dec` produces a list of candidate circuits, and we estimate the agreement of each of these circuits with $f(x)$ up to a small constant error, using random sampling and by computing the mapping $(x,i) \mapsto f(x)_i$, which can be done in time T' using our access to x.



$\Rightarrow$ Each entry $i \in [k]$ of $f(x)$ is computable from x in time T'.

$\Rightarrow$ Each entry $i \in [k^{O(1)}]$ of $\bar{f}(x)$ is computable from x in time $T' \cdot \log(k)$.

Figure 12-2: A diagram of the steps in our construction, noting the lengths of the strings in each step. Note that the function g is $\{0,1\}^{\log(k)} \to \{0,1\}$ and that the function $\bar{g}$ is $\{0,1\}^{O(\log(k))} \to \{0,1\}$. Also, the number of strings in S_x is $\text{poly}(k)$, and each of them has $n^{\varepsilon'}$ bits.

After running this procedure, with high probability we obtain an oracle circuit C of size at most $k^{\alpha/2}$ such that the truth-table of C^D agrees with $f(x)$ on more than 0.99 of the inputs. We can then compute this truth-table in time $k \cdot \tilde{O}(|C|) \cdot T = \tilde{O}(|C|) \cdot T' < k^{\alpha} \cdot T'$ and print a string $\widetilde{f}(x)$ that agrees with $f(x)$ on more than 0.99 of the bits.

Two additional comments. The reconstruction procedure above is "instance-wise", in the sense that for every x such that the probabilistic algorithm distinguishes the outputs of the targeted PRG on x from uniform, the reconstruction procedure prints an approximate version of $f(x)$. Thus, when assuming hardness of f over the uniform distribution (as in Theorem 12.1.7) we obtain derandomization over the uniform distribution; and when hardness of f is over *all polynomial-time-samplable distributions* **x** (as in Theorem 12.1.8) we obtain a derandomization that succeeds over all polynomial-time-samplable distributions. Further details appear in Section 12.6.

As mentioned in Section 12.1.2, we also show that the existence of a "non-batch-computable" function is necessary for average-case superfast derandomization in the natural setting of highly-uniform balanced formulas. In high level, assuming that we can derandomize probabilistic formulas with overhead $T \mapsto T \cdot n^{\varepsilon}$, we first diagonalize against all probabilistic formulas of size T using a function that is computable by a formula F of size $T' = T \cdot n^{\varepsilon}$. We define a non-batch-computable function $g\colon \{0,1\}^n \to \{0,1\}^{n^{2\varepsilon}}$ that, on input x, prints the $n^{2\varepsilon}$ gate values in the appropriate in-

termediate level of $F(x)$. Due to our choice of computational model (*i.e.*, balanced formulas) each output bit of g can be computed in time approximately $T'/n^{2\varepsilon} = T \cdot n^{-\varepsilon}$; whereas probabilistically computing the gate-values of the entire layer in F in time $o(T)$ allows to compute F itself in time $o(T)$, a contradiction to its hardness. See Proposition 12.6.16 for further details.

12.2.2 The Proof of Theorem 12.1.3, Theorem 12.1.4, Theorem 12.1.5, and Theorem 12.1.6

Loosely speaking, in Section 12.2.1 we used the input x to produce a hard truth-table $f(x)$, and then instantiated a version of the NW PRG with $f(x)$ as the hard function. The main technical bottleneck was that the reconstruction procedure for the PRG needed oracle access to $f(x)$, and we were able to answer the procedure's oracle queries by our assumption that the individual bits of $f(x)$ are computable quickly. In Theorem 12.1.3 we have no such assumption, and therefore we will need a more complicated construction.

The classical reconstruction approach of [IW98] and its drawbacks. Previous works following [IW98, TV07] handled a similar technical challenge using an influential "bootstrapping" argument. Recall that this argument instantiates a version of the NW PRG using a hard problem $L \in \mathsf{PSPACE}$ that is both downward self-reducible and randomly self-reducible. Assuming that there exists a distinguisher for the PRG on *all input lengths* $n \in \mathbb{N}$, the reconstruction procedure is *iterative*: For $i = 1, ..., n$, the i^{th} reconstruction step constructs a circuit that decides $L_i = L \cap \{0,1\}^i$, using the circuit for L_{i-1} and the distinguisher for NW^{L_i} (*i.e.*, for the NW PRG that uses L_i as the hard function). The base case of this procedure (*i.e.*, computing L_1) is trivial, each iterative step uses the self-reducibility properties of L, and the conclusion of this argument is that L_n can be efficiently decided for every $n \in \mathbb{N}$, a contradiction to the hardness of L.

There are two well-known drawbacks in this argument. First, the entire argument is "black-box", in the sense that it does not involve the input x that is available to the PRG and to the reconstruction procedure. This is a drawback (rather than an advantage) since it causes their derandomization to *succeed only in average-case*, rather than in the worst-case.[13] Secondly, the argument yields derandomization that *succeeds only on infinitely many input lengths*, since the assumption that

[13] Specifically, their reconstruction procedure is a uniform algorithm that samples an input $x \in \{0,1\}^n$, and uses a corresponding distinguisher D_x for the PRG (that is obtained by running the probabilistic algorithm that we are trying to derandomize on input x) to compute L; since L is hard for probabilistic algorithms, it follows that the probability of sampling x on which the derandomization fails is small. When trying to strengthen the conclusion and deduce that there *does not exist* x on which the derandomization fails, we need to allow the reconstruction procedure access to an *arbitrary string* x. In such a case the reconstruction is a non-uniform procedure (the advice modeled by x), and so the argument does not work under the original uniform hardness hypothesis.

we make towards a contradiction is that a distinguisher succeeds (in distinguishing the PRG from uniform) on almost all input lengths.

A non-black-box version of their reconstruction argument. The main building-block in our proof is a non-black-box version of the [IW98] reconstruction argument, which allows us to overcome the two aforementioned drawbacks. Intuitively, for any fixed $x \in \{0,1\}^n$, instead of using the truth-tables of $L_1, ..., L_n$ as hard functions,[14] we use a *sequence of truth-tables that can be efficiently produced from the input* x, while guaranteeing that this sequence of truth-tables has the structural properties needed for the reconstruction argument. The only downside to our argument is that we construct a targeted HSG, rather than a targeted PRG. Details follow.

We first define a property of the hard function $f\colon \{0,1\}^n \to \{0,1\}^n$ that will allow us to construct a targeted reconstructive HSG using f. For parameters $t \ll T$ and $d \ll T$, we say that f has $(d \times T)$-bootstrapping systems with bootstrapping time t if for every $x \in \{0,1\}^n$ there exists a $d \times T$ matrix $B = B_f(x)$ satisfying the following.

1. **(Base case.)** There is an algorithm that gets input $(x, i) \in \{0,1\}^n \times [T]$, runs in time t, and outputs the i^{th} bit in the bottom row of B.
2. **(Final case.)** There is an algorithm that gets input x and oracle access to the top row of B, and outputs $f(x)$ in time $|f(x)| \cdot t$.
3. **(Self-reducibility.)** There is an algorithm that gets input $(i, j) \in [d] \times [T]$ and oracle access to the $(i-1)^{th}$ row of B, runs in time t, and outputs $B_{i,j}$.
4. **(Error-correction.)** Each row of B is a codeword in a sufficiently good code. (The entries in the matrix will be non-Boolean, but for simplicity we ignore this issue in the current overview.) In particular, a sufficient requirement from the code is that every row, considered as a truth-table, is sample-aided worst-case to rare-case reducible in time t.[15]

See Figure 12-3 for a visual depiction of a bootstrapping system, and see Definition 12.4.1 for the formal definition. We stress that the bootstrapping matrix $B = B_f(x)$ *depends on the particular input* x. In fact, if f has $(d \times T)$ bootstrapping systems with bootstrapping time t, then there is an efficient algorithm that maps x to $B_f(x)$ in time $\mathrm{poly}(T, d, t)$ (*i.e.*, the algorithm iteratively

[14] To be accurate, in the arguments of [IW98, TV07] and of subsequent works the PRG uses the truth-tables of $L_1, ..., L_\ell$, where ℓ is proportional to the seed length of the PRG.

[15] This property of a function g means that there is a probabilistic algorithm that gets input z, access to a highly corrupted version of g, and uniform samples $(r, g(r))$, and outputs $g(z)$ with high probability (see Definition 12.3.8, following Goldreich and Rothblum [GR17]). We can also relax this property, and only require that each row will be a codeword in a locally-decodable code; see the discussion after the proof of Proposition 12.4.2.

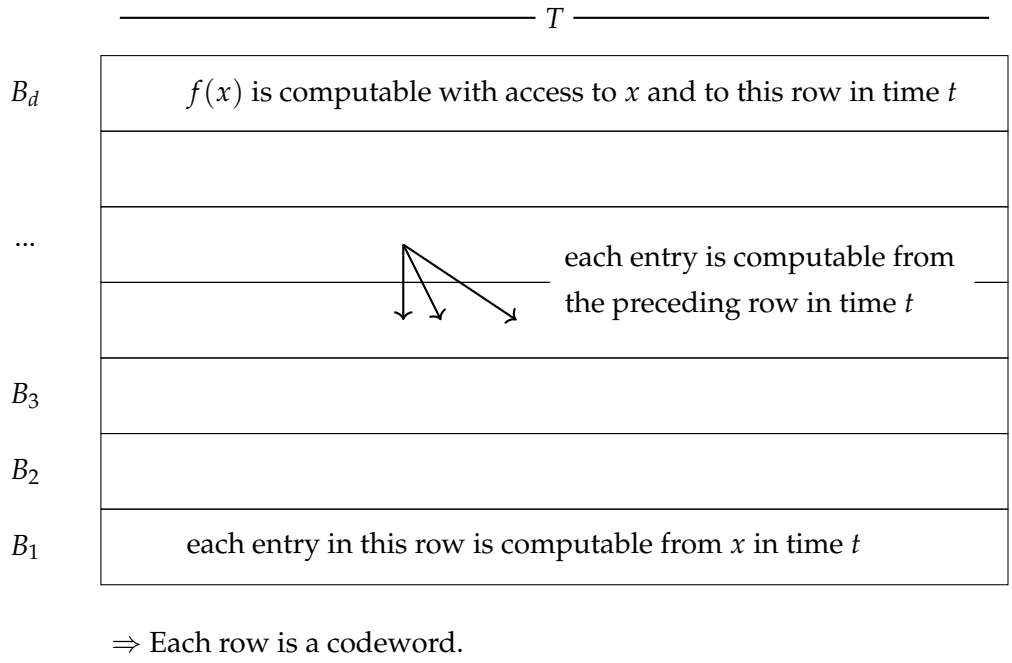


Figure 12-3: Visual depiction of a $(d \times T)$-bootstrapping system $B = B(f, x)$. We denote the i^{th} row of B by B_i.

computes each row in $B_f(x)$ from bottom to top, using the self-reducibility algorithm).

Previous works following [IW98, TV07] can also be viewed as using bootstrapping systems, albeit of a particular and more constrained type than the notion that we leverage (intuitively, the bootstrapping systems implicit in their works are "black-box" since they do not depend on the input; see Section 12.4.1 for an explanation). Using our more general notion, given any function that has $(d \times T)$-bootstrapping systems with bootstrapping time t satisfying $\mathrm{poly}(d, t) \ll T$, we can construct a corresponding reconstructive targeted HSG H_f that has the following parameters:

Proposition 12.2.1 (from bootstrapping systems to a targeted HSG; informal, see Proposition 12.4.4)**.** *Assume that $f\colon \{0,1\}^n \to \{0,1\}^n$ has $(d \times T)$-bootstrapping systems with bootstrapping time t and sufficiently good error-correction properties. Then, for any $\varepsilon > 0$ and $n \leq M(n) \leq \min\left\{T(n)^{\Omega(\varepsilon)}, t(n)^{\Omega(1)}\right\}$ there exist:*

1. **Targeted HSG.** *A deterministic algorithm H_f that gets input $x \in \{0,1\}^n$, runs in time $\mathrm{poly}(T)$, and outputs a set of M-bit strings.*
2. **Reconstruction procedure.** *A probabilistic algorithm R that gets input x and oracle access to a $(1/M)$-distinguisher D (i.e., $\Pr_{r\in\{0,1\}^M}[D(r) = 1] \geq 1/M$ but D rejects all the strings that $H_f(x)$ outputs), runs in time $\mathrm{poly}(t, T^{\varepsilon}, M) \cdot d$, and with probability at least $2/3$ outputs $f(x)$.*

We explain the proof of Proposition 12.2.1, which uses the same ideas as in [IW98] but now applies them to the bootstrapping system $B_f(x)$ that depends on x (rather than to a sequence of the truth-tables of a hard function on input lengths $1, ..., n$). The targeted HSG thinks of each row in $B_f(x)$ as the truth-table of a function, and applies a suitable instantiation of the NW PRG to this truth-table in order to map it to a set of M-bit strings.[16] The union over all $i \in [d]$ of these sets of M-bit strings is the final pseudorandom set of strings that our targeted HSG outputs.

The reconstruction procedure R gets input x and access to a distinguisher D_x for $H_f(x)$, and iteratively constructs small circuits that compute each row of $B_f(x)$, from bottom to top. The first circuit computes the "base case" – the bottom row in the matrix $B_f(x)$. This can be done by a circuit of size $\tilde{O}(t)$ since the reconstruction procedure has access to x. Then, for $i \geq 2$, we mimic the iterative reconstruction argument of [IW98], showing that when given access to a circuit C_{i-1} that computes the truth-table B_{i-1}, and to the distinguisher D_x, we can compute in time $\text{poly}(t, M)$ a circuit C_i that computes the truth-table B_i. (In a gist, this step consists of combining the well-known reconstruction and list-decoding algorithms of [GL89, NW94, IW98] and the sample-aided worst-case to rare-case reducibility algorithm for B_i, which in our construction follows [STV01]; see Section 12.4.4 for details.) Finally, we obtain a circuit C_d that computes B_d, which allows us to compute $f(x)$ by evaluating the circuit $|f(x)| \cdot t$ times. The crucial point is that the reconstruction time is a fixed polynomial in M, d and t, which can be much smaller than T assuming that $t \ll T$ and $d \ll T$.

Indeed, the description above is high-level and omits many technical details, but these generally follow known techniques. The formal connection between bootstrapping systems and HSGs is stated in Proposition 12.4.4 and proved in Section 12.4.4.

Bootstrapping systems and logspace-uniform circuits of bounded depth. So far we proved a generic statement, saying that if f has bootstrapping systems then we can design a targeted reconstructive HSG using f as the hard function. Loosely speaking, the additional result needed to prove Theorem 12.1.3 is that *the class of functions that have bootstrapping systems with $d, t \ll T$ is essentially the class of functions computable by logspace-uniform circuits of bounded depth*.

One direction in this connection is relatively straightforward: If f has $(d \times T)$-bootstrapping systems with bootstrapping time t, then f can be computed by circuits of depth approximately $(d \cdot t)$ and size approximately $(d \cdot t) \cdot T$ (this direction is visually evident from Figure 12-3; see

[16] For simplicity, in this presentation we simply refer "a suitable version" of the NW PRG, ignoring the alphabet from which each entry in the matrix comes from and additional encodings applied to each row.

Proposition 12.4.2 for details). The other direction is the more demanding one: We show that any function computable by logspace-uniform circuits of depth d and size T also has bootstrapping systems with dimensions $d \cdot \log(T) \times \text{poly}(T)$ and bootstrapping time T^ε.

Proposition 12.2.2 (bootstrapping systems for logspace-uniform circuits; informal, see Proposition 12.4.3)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be computable by logspace-uniform circuits of size $T(n)$ and depth $d(n)$ and let $\varepsilon > 0$ be an arbitrarily small constant. Then, for $d' = d \cdot \log(T)$ and $T' = \text{poly}(T)$ there are $(d' \times T')$-bootstrapping systems for f with bootstrapping time $t = T^\varepsilon \cdot n$ and sufficiently good error-correction properties.*

To get initial intuition for the proof, assume for a moment that f from Proposition 12.2.2 is computable by circuits that are not only logspace-uniform, but such that given the index of a gate g in the circuit, we can compute the indices of the two gates feeding into g in time $n^{o(1)}$. Then, it is easy to construct a $(d \times T)$ system $B = B_f(x)$ that satisfies the base case, the final case, and the self-reducibility properties of a bootstrapping system, but does not necessarily satisfy any error-correction property. Specifically, denoting the circuit for f by C_f, and considering the gate-values of C_f when it is given input x, we can define each row in B to be the gate-values in the corresponding row of $C_f(x)$. The self-reducibility property then follows since the value of each gate in the i^{th} row of $C_f(x)$ depends on the values of just two gates in the $(i-1)^{th}$ row of $C_f(x)$, and we can compute the indices of these gates in time $n^{o(1)}$. The challenge in proving Proposition 12.2.2 is to construct a system in which the rows are not only self-reducible, but are simultaneously also error-correctable (*i.e.*, codewords in a good code), and to do so even when the circuit only satisfies the weaker logspace-uniformity property.

To do so we rely on the ideas underlying the doubly-efficient proof system of Goldwasser, Kalai, and Rothblum [GKR15]. Let us recall their construction. For $i = 1, ..., d$, denote by $\alpha_i \in \{0,1\}^T$ the string representing the gate-values in the i^{th} row of $C_f(x)$ (note that we index the bottom row by 1 and the top row by d). Thinking of each α_i as a function $\{0,1\}^{\log(T)} \to \{0,1\}$, let $\hat{\alpha}_i$ be a suitable low-degree extension of α_i over a field of sufficiently large polynomial size (the precise degree of each $\hat{\alpha}_i$ will be T^ε for a small constant $\varepsilon > 0$).

As a first step (which does not yet complete the construction), define a system $\widetilde{B}$ in which the i^{th} row is the truth-table of $\hat{\alpha}_i$ (*i.e.*, the evaluation of the polynomial $\hat{\alpha}_i$ on all inputs). Indeed, low-degree polynomials are codewords in the Reed-Muller code, and this code satisfies the error-correction properties that we need in a bootstrapping system (see Section 12.3.4). However, we

now need to show that the rows in the system are self-reducible.

The key technical idea in [GKR15] is an interactive protocol that reduces computing $\hat{\alpha}_{i+1}$ at any given point $\vec{w}$ to computing $\hat{\alpha}_i$ at a different point (that the verifier chooses). This interactive reduction consists of running an appropriate sumcheck protocol on a low-degree polynomial that expresses the value $\hat{\alpha}_{i+1}(\vec{w})$ as the weighted sum of values of $\hat{\alpha}_i$ on T^2 inputs. Specifically, let $\hat{\Phi}$ be an appropriate arithmetization of the circuit-structure function (*i.e.*, of the function that gets input (w, u, v) and outputs 1 iff gate w is fed by gates u and v), and assume wlog that all gates in C_f compute the `NAND` function. Then, we have that

$$\hat{\alpha}_{i+1}(\vec{w}) = \sum_{\vec{u},\vec{v}} \hat{\Phi}(\vec{w}, \vec{u}, \vec{v}) \cdot (1 - \hat{\alpha}_i(\vec{u}) \cdot \hat{\alpha}_i(\vec{v})) \ , \tag{12.2.1}$$

where the summation ranges over the arithmetizations $\vec{u}, \vec{v}$ of all T^2 pairs of gates u, v in the i^{th} layer. (It is not a-priori clear how to arithmetize the circuit-structure function by a polynomial $\hat{\Phi}$ of a low degree. This problem was solved in [GKR15], relying on the assumption that C_f is logspace-uniform, using another auxiliary interactive protocol; we avoid this auxiliary protocol relying on a simplification suggested by Goldreich [Gol18]. See Section 12.4.3 for details.)

The sumcheck protocol reduces computing the LHS in Eq. (12.2.1) to computing $\hat{\alpha}_i$ at two points, in at most $\log(T)$ rounds.[17] The last observation that we need to complete the proof of Proposition 12.2.2 dates back to the work of Trevisan and Vadhan [TV07]: They observed that the sumcheck protocol can be thought of as yielding a sequence of polynomials that is self-reducible, in the sense that computing each polynomial reduces to computing the subsequent polynomial in a small number of points. (Each polynomial corresponds to a round in the protocol, and the number of points corresponds to the degree of the original polynomial in Eq. (12.2.1).)

We are now ready to define the bootstrapping system. We start from the initial system $\widetilde{B}$ whose layers are the $\hat{\alpha}_i$'s, and in between each pair of layers we add the polynomials that arise from the sumcheck protocol. The self-reducibility property now follows since the polynomials in each pair of layers either both come from the sumcheck protocol (in which case they are self-reducible by the observation of [TV07]), or one of them is the last one in the sequence of polynomials corresponding to $\hat{\alpha}_{i+1}$ and the other one represents $\hat{\alpha}_i$ (in which case the former polynomial just depends on the values of the latter in two points). For the full proof details see Proposition 12.4.3 and

[17] In fact, since we use low-degree extensions $\hat{\alpha}_i$ of degree $T^{\Omega(1)}$, the number of rounds will be constant. However, this is immaterial to the high-level overview and we mention it only to avoid confusion.

Section 12.4.3.

An alternative view of our construction. An equivalent way of viewing our construction is that each row in the bootstrapping system is the truth-table of the prover's strategy function on input x in a corresponding round of the [GKR15] protocol. Specifically, the d^{th} row (*i.e.*, the top row) corresponds to the first round, in which the prover just sends the claimed value $f(x)$; the $(d-i)^{th}$ row corresponds to round $i+1$ in the proof system; and the bottom row corresponds to the last round in the proof system. From this perspective, the reconstruction procedure for B that was described above reconstructs the prover strategy functions in the protocol of [GKR15], round-by-round, starting from the last round in the interaction and working back until the first round.

Wrapping-up: Proof of Theorem 12.1.3 Combining Proposition 12.2.1 and Proposition 12.2.2, if f has logspace-uniform circuits of size $T(n) = \text{poly}(n)$ and depth $d(n) = n^2$, then we have a reconstructive targeted HSG H_f with output length $M = O(n) = T^{\Omega(1)}$ and reconstruction time $\bar{t} = \text{poly}(T^{\varepsilon}, n)$. In particular, if there exists an f as above that cannot be computed by probabilistic algorithms in time $\bar{t}$ on *almost all inputs* x, then for every x we have that H_f "fools" all linear-time machines that also get access to x, and it follows that $\mathsf{prRP} = \mathsf{prP}$. The standard reductions of [Sip83, Lau83] imply that $\mathsf{prBPP} = \mathsf{prP}$, concluding the proof of Theorem 12.1.3.

We wish to highlight where we used the assumption that f is computable in parallel, *i.e.* by uniform circuits of low depth. Denoting the circuit depth by $d = d(n)$, we construct (via Proposition 12.2.2) a bootstrapping system of slightly larger depth $d' > d$, although for now we can pretend that $d' = d$. The reconstruction algorithm from Proposition 12.2.1 then runs in d' iterations, and in each iteration it performs a computation in time $\text{poly}(n)$ (when setting the parameters M, T^{ε} and t as above). Thus, our hardness assumption is for probabilistic algorithms that run in time $d' \cdot \text{poly}(n)$. In particular, this means that the deterministic time bound for computing f must be noticeably larger than the depth d' of the bootstrapping system and of the circuit for f.

We also comment that to deduce the derandomization conclusion, it is not actually necessary to assume that f is hard for *all* probabilistic algorithms that run in the prescribed time; instead, it suffices to assume that f is hard for a *particular* algorithm. See Remark 12.5.6 for details.

Extensions: Proofs of Theorem 12.1.4, Theorem 12.1.5, and Theorem 12.1.6. The generalization of Theorem 12.1.3 to Theorem 12.1.4 (*i.e.*, scaling the time bounds for the hard function and the derandomization algorithm) is mainly based on parametric modifications to the argument, and

does not require additional new ideas. However, one interesting point is that we reduce derandomization of prBPTIME to derandomization of prRTIME with a significant time overhead (using the classical techniques of [Sip83, Lau83], following [BF99]); more efficient reductions are known (see [ACR98, GVW11]), but these work only when the derandomization of prRTIME is black-box (using HSGs), whereas our derandomization is non-black-box.

Scaling Theorem 12.1.3 to smaller circuit classes, and in particular to logspace-uniform NC as in Theorem 12.1.6, requires significantly more work and involves many low-level technical details. Given a function f from the class, we need to ensure that the construction of bootstrapping system, targeted HSG, and reconstruction procedure can all be carried out by circuits that are both of *polylogarithmic depth* and *logspace-uniform*. To do so we prove several technical results that are seemingly new (although they rely on known high-level algorithmic ideas); one result that might be of independent interest is that the list-decoding procedure for the Reed-Muller code can be implemented by such circuits, since the corresponding special case of list-decoding the Reed-Solomon code can be implemented by such circuits. This construction relies on an idea that was communicated to us by Madhu Sudan, and is presented in Section 12.8.

Lastly, let us explain how we prove the "low-end" hardness-to-randomness tradeoff in Theorem 12.1.5, in which there are *no structural restrictions* on the hard function f. First, if $\mathsf{EXP} \not\subset \mathsf{P/poly}$ then $\mathsf{prBPP} \subseteq$ i.o.-$\mathsf{prSUBEXP}$ (using the NW PRG, see [BFNW93]) and we are done. Otherwise, by [BFNW93], we have that $\mathsf{EXP} = \mathsf{MA}$. In particular, the function f that is computable in time 2^{n^c} has MA protocols running in some *fixed* time bound n^k. The crucial observation is that any such MA protocol can by simulated by a uniform circuit that, while having size $2^{O(n^k)}$, is nevertheless of *fixed polynomial depth* $d(n) = n^{O(k)}$; this is by a brute-force circuit that enumerates the witnesses for the MA verifier *in parallel*. Moreover, this circuit is logspace-uniform, since we just use the standard transformation of Turing machines to highly-uniform circuits that compute the tableau of the machine's computation.

We now want to base a reconstructive targeted HSG H_f on this hard function f. To do so we need to tweak the parameters in Proposition 12.2.1 and Proposition 12.2.2 so that the reconstructive overhead and bootstrapping time will both be proportional to $\mathrm{polylog}(T) = \mathrm{poly}(n)$ rather than to $T^\varepsilon = 2^{\mathrm{poly}(n)}$. This modification is indeed possible, where the cost is that the dimensions of the bootstrapping system increase and the resulting targeted HSG is slower, working in time $2^{\mathrm{polylog}(T)} = 2^{n^{O(c)}}$ (see Proposition 12.4.3 and Proposition 12.4.4 for details). This overhead is fortunately good enough for the current setting: Instantiating the targeted HSG H_f with such overhead

and with polynomial output length M, the derandomization time is a fixed exponent $2^{n^{O(c)}}$ and the reconstruction time is $\text{polylog}(T) \cdot \text{poly}(n) \cdot d = \text{poly}(n)$. Thus, if f is hard for probabilistic polynomial-time algorithms then $\mathsf{pr}\mathcal{RP}$ can be simulated in fixed exponential time $2^{n^{O(c)}}$.

12.3 Preliminaries

Throughout this chapter we use the notation $\langle x, y \rangle$ to denote the inner-product over $\mathbb{F}_2$; that is, $\langle x, y \rangle = \oplus_i (x_i \cdot y_i)$. We will typically denote random variables by boldface.

Unless explicitly stated otherwise, we assume that all circuits are comprised of Boolean NAND gates of fan-in two and are layered, where the latter property means that gates at distance i from the inputs only feed to gates at distance $i + 1$. [18] In several places in this chapter we will need the following notion, which strengthens the standard notion of a time-computable function by requiring that the function will be computable in logarithmic space.

Definition 12.3.1 (logspace-computable functions)**.** *We say that a function $T : \mathbb{N} \to \mathbb{N}$ is* logspace-computable *if there exists an algorithm that gets input 1^n, runs in space $O(\log(T(n)))$, and outputs $T(n)$.*

12.3.1 Pseudorandomness and Targeted Pseudorandom Generators

Our non-black-box derandomization algorithms rely on constructions of *targeted pseudorandom generators* and *targeted hitting-set generators*. Targeted PRGs were defined by Goldreich [Gol11a], who showed that the existence of polynomial-time computable targeted PRGs with logarithmic seed length is equivalent to the hypothesis $\mathsf{prBPP} = \mathsf{prP}$. Let us recall his definition of targeted PRGs, and also define a targeted HSGs in the natural way:

Definition 12.3.2 (targeted PRG; see [Gol11a, Definition 4.10])**.** *For $T : \mathbb{N} \to \mathbb{N}$, let G be an algorithm that gets input $x \in \{0,1\}^n$ and a seed of length $\ell(n)$ and outputs a string of length $T(n)$. We say that G is a* targeted pseudorandom generator with error μ for time T *(or μ-targeted-PRG for time T, in short) if for every algorithm A running in time T, every sufficiently large $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$ it holds that*

$$\left| \Pr_{r \in \{0,1\}^{T(n)}} [A(x,r) = 1] - \Pr_{s \in \{0,1\}^{\ell(n)}} [A(x, G(x,s)) = 1] \right| \leq \varepsilon \,.$$

[18] The assumption that circuits are layered is made merely for simplicity. In particular, logspace-uniform circuits (as defined in Definition 12.3.5) that are not layered can be transformed to logspace-uniform circuits that are layered with a linear overhead in size. (By adding dummy gates at each layer, and since the distance of each gate from the inputs can be computed in space that is logarithmic in the size of the circuit.)

Definition 12.3.3 (targeted HSG). *For $T\colon \mathbb{N} \to \mathbb{N}$, let H be an algorithm that gets input $x \in \{0,1\}^n$ and a random seed of length $\ell(n)$ and outputs a string of length $T(n)$. We say that H is a* targeted hitting-set generator with error μ for time T *(or μ-targeted-HSG for time T, in short) if for every algorithm A running in time T, every sufficiently large $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$ such that $\Pr_{r\in\{0,1\}^{T(n)}}[A(x,r) = 1] \geq 1/2$, there exists $s \in \ell(n)$ such that $A(x, H(x,s)) = 1$.*

The foregoing definitions refer to PRGs and HSGs that fool distinguishers in the worst-case (*i.e.*, for every $x \in \{0,1\}^n$). These definitions extend naturally to PRGs and HSGs that only fool distinguishers on average-case (*i.e.*, with high probability over choice of x according to some pre-determined probability distributions).

In Section 12.6 we rely on the following claim, which asserts that if one-way functions exist, then there exist PRGs with seed length n^ε that are computable in time $n^{1+\varepsilon}$, for an arbitrarily small $\varepsilon > 0$. The proof of this claims amounts to using the classical constructions of PRGs from one-way functions [HILL99] and then applying standard techniques to extend the expansion factor of PRGs (see, *e.g.*, [Gol01, Construction 3.3.2]).

Theorem 12.3.4 (OWFs yield PRGs with near-linear running time)**.** *If there exists a polynomial-time computable one-way function secure against polynomial-time algorithms, then for every $\varepsilon > 0$ there exists a PRG that has seed length $\ell(n) = n^\varepsilon$, is computable in time $n^{1+\varepsilon}$, and fools every polynomial-time algorithm with negligible error.*

Proof. By [HILL99], the hypothesis implies that for some negligible function negl there exists $G_1\colon \{0,1\}^m \to \{0,1\}^{2m}$ that is computable in time m^c, and for all $k \in \mathbb{N}$, no probabilistic algorithm running in time $(2m)^k$ can distinguish between $G_1(\mathbf{u}_m)$ and $\mathbf{u}_{2m}$ with advantage at least $\mathsf{negl}(2m)$. Our PRG G gets input 1^n and a seed $x \in \{0,1\}^m$ where $m = n^{\varepsilon/2c} < n^\varepsilon$ and acts as follows.

1. Let $\sigma_1 = x$.
2. For $i \in \{2,3,...,n/m\}$, compute σ_i as the last m bits of $G_1(\sigma_{i-1})$.
3. Output the concatenation of $\sigma_1, \sigma_2, ..., \sigma_{n/m}$, which is an n-bit string

The running time of G is at most $m^c \cdot n + O(n) \leq n^{1+\varepsilon}$, and a standard hybrid argument (as in [Gol01, Theorem 3.3.3]) reduces distinguishing G_1 with noticeable advantage to distinguishing G with noticeable advantage. ■

12.3.2 Logspace-uniform Circuits

Recall that, as mentioned in the introduction, we generalize the definition of logspace-uniform circuits to super-polynomial size bounds, in a straightforward way. The usual definition refers to circuits of size $\text{poly}(n)$ whose adjacency relation (*i.e.*, $\Phi(u,v,w) = 1$ iff gates v, w feed into gate u) can be decided in space $O(\log(n))$; that is, in space that is linear in the input size to the adjacency relation function. We simply scale this definition up, while maintaining the requirement that the adjacency relation can be decided in space logarithmic in the circuit size (*i.e.*, linear in the input size to the adjacency relation); that is:

Definition 12.3.5 (logspace-uniform circuit)**.** *We say that a circuit family $\{C_n\}_{n\in\mathbb{N}}$ of size $T(n)$ is* logspace-uniform *if there exists an algorithm A such that:*

1. *(***Decides the adjacency relation.***) The algorithm gets as input $(u,v,w) \in \{0,1\}^{3\log(T(n))}$ and accepts if and only if the gates in C_n indexed by v and by w feed into the gate in C_n indexed by u.*
2. *(***Runs in linear space.***) On an input of length $\ell = 3\log(T(n))$, the algorithm runs in space $O(\ell)$.*

When we mention logspace-uniform probabilistic circuits *we refer to circuits that are logspace-uniform and also use additional input gates that are assigned random values (i.e., the randomness is used by the circuit rather than by the logspace algorithm that constructs the circuit).*

Note that Definition 12.3.5 is equivalent to a definition asserting that there exists an algorithm that gets input 1^n, runs in space $O(\log(T))$, and prints a description of C_n (where the description is a list of gates, and for each gate we list the indices of gates feeding into it).

12.3.3 Average-case Complexity Classes and Simulations

We now recall standard definitions of average-case simulation of a problem $L \subseteq \{0,1\}^*$. The following definitions refer to average-case simulation over an arbitrary distribution, over the uniform distribution, and over all polynomial-time samplable distributions, respectively.

Definition 12.3.6 (average-case simulation over arbitrary distributions)**.** *Let $L \subseteq \{0,1\}^*$, let $\beta\colon \mathbb{N} \to (0,1)$ let $\mathcal{C}$ be a complexity class, and let $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions (where $\mathbf{x}_n$ is a distribution over n-bit inputs). We say that $L \in \text{heur}_{\mathbf{x},1-\beta}$-$\mathcal{C}$ if there exists $C \in \mathcal{C}$ such that for every sufficiently large $n \in \mathbb{N}$ it holds that $\Pr_{x\sim\mathbf{x}_n}[C(x) = L(x)] \geq 1 - \beta(n)$.*

Definition 12.3.7 (average-case simulation, two special cases)**.** *Let $L \subseteq \{0,1\}^*$, let $\beta\colon \mathbb{N} \to (0,1)$ and let $\mathcal{C}$ be a complexity class. Then:*

1. *We say that* $L \in \mathsf{avg}_{1-\beta}\text{-}\mathcal{C}$ *if there exists* $C \in \mathcal{C}$ *such that for every sufficiently large* $n \in \mathbb{N}$ *it holds that* $\Pr_{x\in\{0,1\}^n}[C(x) = L(x)] \geq 1 - \beta(n)$ *(i.e., the choice of* x *is according to the uniform distribution).*
2. *We say that* $L \in \mathsf{heur}_{1-\beta}\text{-}\mathcal{C}$ *if there exists* $C \in \mathcal{C}$ *such that for every polynomial-time samplable ensemble* $\mathbf{x}$ *of distributions and sufficiently large* $n \in \mathbb{N}$ *it holds that* $\Pr_{x\sim\mathbf{x}_n}[C(x) = L(x)] \geq 1 - \beta(n)$.

We say that a complexity class $\mathcal{C}'$ satisfies $\mathcal{C}' \subseteq \mathsf{avg}_{1-\beta}\text{-}\mathcal{C}$ if for every $L \in \mathcal{C}'$ it holds that $L \in \mathsf{avg}_{1-\beta}\text{-}\mathcal{C}$ (and similarly define $\mathcal{C}' \subseteq \mathsf{heur}_{\mathbf{x},1-\beta}\text{-}\mathcal{C}$ and $\mathcal{C}' \subseteq \mathsf{heur}_{1-\beta}\text{-}\mathcal{C}$).

12.3.4 Sample-aided Worst-case to Rare-case Reductions for Polynomials

Our algorithms will use sample-aided worst-case to rare-case reductions. The foregoing term was coined recently by Goldreich and Rothblum [GR17], and is implicit in many previous works. Intuitively, a sample-aided worst-case to rare-case reduction for a function f is an algorithm that computes f at any point given a "highly corrupted" version $\widetilde{f}$ of f as well as random labeled examples of f. We also specialize this definition to the case where the algorithm can be implemented by logspace-uniform circuits of bounded depth.

Definition 12.3.8 (sample-aided reductions)**.** *For* $s, w\colon \mathbb{N} \to \mathbb{N}$ *and* $\rho, \varepsilon\colon \mathbb{N} \to [0,1]$*:*

1. *Let* $f\colon \{0,1\}^* \to \{0,1\}^*$ *be a function that maps* n *bits to* $w(n)$ *bits.*
2. *Let* M *be a probabilistic procedure 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* $\widetilde{f}_n\colon \{0,1\}^n \to \{0,1\}^{w(n)}$*, and outputs a circuit* $C\colon \{0,1\}^n \to \{0,1\}^{w(n)}$ *with oracle gates.*

We say that M *is a* sample-aided reduction of computing f in the worst-case to computing f on ρ of the inputs using a sample of size s and with error ε *if for every* $\widetilde{f}_n\colon \{0,1\}^n \to \{0,1\}^{w(n)}$ *satisfying* $\Pr_{x\in\{0,1\}^n}\left[\widetilde{f}_n(x) = f_n(x)\right] \geq \rho(n)$ *the following holds: With probability at least* $1 - \varepsilon$ *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* $M^{\widetilde{f}_n}(1^n, (r_i, f_n(r_i))_{i\in[s(n)]})$ *outputs an oracle circuit* C *such that* $Pr\left[C^{\widetilde{f}_n}(x) = f_n(x)\right] \geq 2/3$ *for every* $x \in \{0,1\}^n$.

Definition 12.3.9 (sample-aided worst-case to rare-case reducibility)**.** *For* $\rho, \varepsilon\colon \mathbb{N} \to (0,1)$*, and* $T, D, s\colon \mathbb{N} \to \mathbb{N}$*, we say that a function* $f\colon \{0,1\}^* \to \{0,1\}^*$ *is* sample-aided worst-case to ρ-rare-case reducible by logspace-uniform circuis of size T and depth D with error ε and sample size s *if there exists a sample-aided reduction* M *of computing* f *in worst-case to computing* f *on* ρ *of the inputs such that* M

uses $s(n)$ samples and has error $\varepsilon > 0$, and can be implemented by a logspace-uniform circuits of size $T(n)$ and depth $D(n)$.

The following result asserts that low-degree polynomials (*i.e.*, the Reed-Muller code) are sample-aided worst-case to rare-case reducible. The proof of the foregoing statement is quite standard, but in the following result we will also assert something stronger: We claim that the reduction can be implemented by logspace-uniform circuits of polylogarithmic depth (*i.e.*, in logspace-uniform NC). A reduction meeting this additional efficiency requirement, and in particular the logspace-uniformity of the circuits, seems not to have been known before. The full proof involves many low-level details, and we present it in Section 12.8. A key idea in the proof was suggested to us by Madhu Sudan.

Proposition 12.3.10 (low-degree polynomials are uniformly sample-aided worst-case to average–case reducible)**.** *Let $q\colon \mathbb{N} \to \mathbb{N}$ be a function mapping integers to primes, let $\ell\colon \mathbb{N} \to \mathbb{N}$ such that $n \geq \ell(n) \cdot \log(q(n))$, and let $d\colon \mathbb{N} \to \mathbb{N}$. Let $f = \{f_n\}_{n\in\mathbb{N}}$ 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-rare-case reducible by logspace-uniform oracle circuits of size $\mathrm{poly}(q, \ell)$ and depth $\mathrm{polylog}(q, \ell)$ with error $1 - 2^{-q}$ and $\mathrm{poly}(q)$ samples, where $\rho = 10\sqrt{d(n)/q(n)}$.*

12.4 A Targeted HSG via Bootstrapping Systems

The main technical result underlying Theorem 12.1.3 is a construction of a reconstructive targeted HSG that is based on any function computable by logspace-uniform circuits of bounded depth. In this section we present this construction.

First, in Section 12.4.1 we formally define bootstrapping systems and discuss the complexity of functions with bootstrapping systems. Then, in Section 12.4.2 we state the two technical results that we will need for our HSG construction, and then use them to state the construction and prove it. Finally, in Section 12.4.3 and Section 12.4.4 we prove each of the two technical results, respectively.

Throughout the section, we will frequently refer to integers in $[T]$ as representing gates in a given circuit of size T. This representation refers to the indices of gates according to an ordering in some fixed canonical way of describing the entire circuit as a list of gates.[19] We also consider

[19] The only exception to this rule is in the proof of Claim 12.4.8, where a particular algorithm will need to describe circuits in a specific different way. We will explicitly mention this point in that proof.

integers that represent gates in individual layers of the circuit; this refers to indices of gates between $1, ..., T$ where T is the number of gates in the layer, where the ordering of gates is induced by the same global ordering of all the gates in the circuit.

12.4.1 Bootstrapping Systems

The following definition of bootstrapping systems expands on the one that was presented in Section 12.2.2. The main difference is that in the definition below we distinguish the complexity of the algorithms for each of the different tasks associated with a bootstrapping system, whereas in Section 12.2.2 we just bounded them all by a single parameter (denoted t there). After the definition we demonstrate a typical setting of parameters that we will use.

Definition 12.4.1 (bootstrapping systems)**.** *Let $T, d, A : \mathbb{N} \to \mathbb{N}$ and $\rho : \mathbb{N} \to (0,1)$ be logspace-computable functions. We say that a function $f : \{0,1\}^* \to \{0,1\}^*$ has $(d \times T)$-*bootstrapping systems *with* alphabet size *A if for every $x \in \{0,1\}^n$ there exists a sequence of strings $P_0(x), ..., P_{d(n)}(x) \in [A(n)]^{T(n)}$ with the following properties:*

1. *(***Layers are efficiently printable.***) There exists an efficient algorithm that gets input $(x, i) \in \{0,1\}^n \times [d(n)]$ and prints the string $P_i(x)$ (using the natural encoding of integers in $[A(n)]$ as binary strings).*
2. *(***Base case.***) There exists an efficient algorithm that gets input $(x, j) \in \{0,1\}^n \times [T(n)]$ and outputs $P_0(x)_j$.*
3. *(***Downward self-reducibility.***) There exists an efficient algorithm that gets input $(x, i, j) \in \{0,1\}^n \times [d(n)] \times [T(n)]$ and oracle access to $P_{i-1}(x)$, and outputs $P_i(x)_j$.*
4. *(***Worst-case to rare-case reducibility.***) Each $P_i(x)$, considered as the truth-table of a function, is sample-aided worst-case to $\rho(n)$-rare-case self-reducible.*[20]
5. *(***Final case.***) There exists an efficient algorithm that gets input $x \in \{0,1\}^n$ and oracle access to $P_{d(n)}(x)$ and outputs $f(x)$.*

When claiming that a function f has bootstrapping systems, we will specify the precise complexities *of each of the efficient algorithms mentioned in Items (1), (2), (3), (4) and (5). We also call the parameter A the* alphabet size*, and the parameter ρ the* rare-case agreement*.*

[20] A minor technicality is that we defined sample-aided worst-case to rare-case reductions for Boolean functions $f : \{0,1\}^* \to \{0,1\}^*$, whereas here for each $n \in \mathbb{N}$ we have a collection of functions $\{P_i\}$ and we define the reduction for each function P_i in the collection. Our intention is that there exists a single uniform algorithm that gets input n and i and performs the reduction for P_i.

We will typically use bootstrapping systems in which the algorithm for printing layers runs in time $\text{poly}(T)$, the algorithms in Items (2), (3), (4) and (5) run in time $t \ll T$ (say, $t = T^{\delta}$ for a very small constant $\delta > 0$), and the rare-case agreement is $\rho(n) = t^{-\Omega(1)}$. We note that there are several natural relaxations of our requirements from bootstrapping systems that still allow our main argument to follow through; see details below.

The complexity of functions with bootstrapping systems. Let f be a function that has a $(d \times T)$-bootstrapping with alphabet size $o(2^n)$, and assume that the algorithms for Items (2), (3), (4), and (5) all work in time t. We observe that f can be computable by logspace-uniform circuits of depth approximately $d \cdot t^2$ and size approximately $T \cdot (d \cdot t^2)$. In particular, for our parameter setting $\text{poly}(d, t) \ll T$, these are logspace-uniform circuits of size T and depth $\text{poly}(d, t) \ll T$.

Proposition 12.4.2 (functions with bootstrapping systems are computable by bounded-depth circuits)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a function that has $(d \times T)$-bootstrapping systems with alphabet size $2^{o(n)}$, and assume that the algorithms for Items (2), (3), and (5) all work in time $t(n)$. Then f can be computed by a logspace-uniform circuit of depth $\tilde{O}(t(n)) \cdot d(n)$ and size $\tilde{O}(T(n) \cdot t(n)) \cdot d(n) \cdot n$.*

Proof. We rely on the fact that a time-t Turing machine can be simulated by a logspace-uniform circuit of size $O(t^2)$ (*i.e.*, the standard method of computing the tableau of the machine's computation can be implemented by logspace-uniform circuits).

Given $x \in \{0,1\}^n$, iteratively for $i = 0, ..., d(n)$, the circuit computes a binary representation of $P_i(x)$, which is of length $T(n) \cdot o(n) \leq T(n) \cdot n$. By the base case in Definition 12.4.1, each block of $\log(o(n))$ bits in $P_0(x)$ can be computed in time t, and hence by a logspace-uniform circuit of size $O(t^2)$. Thus, we can compute the binary representation of $P_i(x)$ by a circuit of size $T(n) \cdot \tilde{O}(t(n))$. Similarly for $i \in [d(n)]$, we compute a binary representation of $P_{i+1}(x)$ from the precomputed binary representation of $P_i(x)$ using a logspace-uniform circuit of size $O(T(n) \cdot t(n)^2)$. Oracle gates of the circuit (which issue queries to the preceding layer $P_i(x)$) can be replaced by a gadget that implements the indexing function, which is computable by a logspace-uniform circuit of size $\tilde{O}(T(n))$. The output layer of the circuit uses the final case algorithm in a similar manner. The overall depth of this circuit is $(d(n) + 2) \cdot O(t(n)^2)$, and its size is $O(t(n)^2 \cdot T(n) \cdot d(n) \cdot n)$. ■

Relaxing the requirements. There are two natural relaxations of the requirements in Definition 12.4.1 that still allow our main argument (*i.e.*, the construction of an HSG from bootstrapping

systems in Proposition 12.4.4) to follow through. First, the requirement that each layer is worst-case to rare-case reducible can be relaxed, only requiring that each layer is a codeword in a code that has an efficient local decoder from a constant (say, $1/4$) fraction of errors. This is the case because given a bootstrapping system that only satisfied the relaxed requirement, we can apply the derandomized direct product construction of Impagliazzo and Wigderson [IW97] to each layer: Their construction transforms each such codeword into the truth-table of a function that is sample-aided worst-case to rare-case reducible, where each entry in the new truth-table can be efficiently computed using logarithmically many queries to the previous truth-table (see Theorem 12.7.5 for precise details).

Secondly, when presenting Definition 12.4.1 we implicitly assumed that the algorithms in Items (2), (3) and (5) are deterministic. While this is the case in our particular constructions, the proof of Proposition 12.4.4 follows through even if these algorithms are probaiblistic (this is since the reconstruction argument that uses these algorithms is probabilistic to begin with).

Previous works as using bootstrapping systems. As mentioned in Section 12.2, previous works in uniform hardness-to-randomness can be viewed in retrospect as relying on bootstrapping systems. Specifically, in the works of Impagliazzo and Wigderson [IW98] and of Trevisan and Vadhan [TV07] the hard function was a set $L \subseteq \{0,1\}^*$ that is downward self-reducible and randomly self-reducible (instead of a multi-output function f as in our work), and given $x \in \{0,1\}^n$ considered the following bootstrapping system: For each $i \in [n]$, the layer P_i is the truth-table of L on inputs of length i.

To see that this yields bootstrapping systems as in Definition 12.4.1, note the downward self-reducibility of L yields the algorithm in Item (3), and the random self-reducibility of L ensures that each row in the bootstrapping system is a codeword in a code that has an efficient local decoder (which, as explained above, suffices to meet the requirement in Item (4)). The layers in their constructions are printable in polynomial space, and the algorithms for Items (2) and (5) follow since L_1 is trivially-computable and since L_n at location x is simply the sought value $L(x)$.

An important point to notice, which we already mentioned in Section 12.2, is that the foregoing bootstrapping systems *are identical for all inputs of a given length n* (*i.e.*, the bootstrapping system matrices $B_L(x)$ and $B_L(x')$ are identical for every $x, x' \in \{0,1\}^n$). In contrast, our bootstrapping systems will depend on the particular input.

12.4.2 The Result Statements: A Reconstructive Targeted HSG

We now state the two main technical results that we need for our HSG construction, and then combine them to obtain the HSG construction. The first technical result asserts that any function computable by logspace-uniform circuits of bounded depth has efficient bootstrapping systems. When parsing the parameters below, we encourage the reader to notice that the algorithms for computing P_0, for downward self-reducibility, and for worst-case to rare-case reducibility, all run in time much smaller than T (*i.e.*, in time $t = T^{\mu}$ for a very small μ).

Proposition 12.4.3 (bootstrapping systems based on the proof system of [GKR15])**.** *There exist two universal constants $\alpha \in (0,1)$ and $k > 1$ such that the following holds. Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be length-preserving function computable by a logspace-uniform family $\{C_n\}_{n\in\mathbb{N}}$ of circuits of size at most $T(n)^k$ and depth $d(n)$. Then, there exist $d' = O(d \cdot \log(T))$ and $T' = \mathrm{poly}(T)$ such that for every logspace-computable $\mu\colon \mathbb{N} \to (0,1)$ satisfying $t = T^{\mu} \geq \log^{1/\alpha}(T)$, the function f has $(d' \times T')$-bootstrapping systems with alphabet size $O(\log(T))$ and the following properties:*

1. *The algorithm that prints $P_i(x)$ is a logspace-uniform circuit of size $\mathrm{poly}(T)$ and depth $d' + O(\log^2(T))$.*
2. *The algorithm that computes $P_0(x)$ is a logspace-uniform circuit of size $\max\{n,t\} \cdot t$ and depth $O(\log^2(T))$.*
3. *The downward self-reducibility algorithm is a logspace-uniform circuit of size t and depth $O(\log^2(T))$.*
4. *The sample-aided worst-case to ρ-rare-case algorithm supports agreement $\rho = t^{-\alpha} \cdot \mathrm{polylog}(T)$, has error $2^{-t^{\alpha}}$, and is computable by logspace-uniform circuits of size t and depth $\log^{1/\alpha}(T)$.* [21]

The second technical result below constructs a reconstructive targeted HSG based on any function f that has an efficient bootstrapping system. We stress that the transformation does *not depend on f being logspace-uniform or having bounded-depth circuits*, but holds for any function with bootstrapping systems. (Indeed, only in the "moreover" we assume that f has this particular form, in which case we deduce that the algorithms associated with the HSG are also logspace-uniform circuits of bounded depth.) When parsing the parameters below, we encourage the reader to think of t and t_0 as very small compared to T' (*e.g.*, $t = T^{\gamma}$ for a very small γ).

Proposition 12.4.4 (from bootstrapping systems to a targeted HSG)**.** *There exist universal constants $c', c'' > 1$ such that the following holds.*

Assumption: *For $T', A, \bar{T}, t_0, t, d'\colon \mathbb{N} \to \mathbb{N}$ and $\alpha \in (0,1)$ such that $\max\{A(n), t(n), t_0(n), d'(n)\} \leq$*

[21] The polylogarithmic power in the definition of ρ depends on the family $\{C_n\}$.

$T'(n)$, let f be a length-preserving function that has $(d' \times T')$-bootstrapping systems with alphabet size A satisfying the following:

1. *The algorithm that prints $P_i(x)$ runs in time $\bar{T}$.*
2. *The algorithm that computes $P_0(x)$ runs in time t_0.*
3. *The downward self-reduction runs in time t.*
4. *The sample-aided worst-case to rare-case reduction runs in time t, supports a rare-case agreement of $t^{-\alpha}$, and has error $2^{-t^{\alpha}}$.*

Conclusion: Then, for every time-computable $M\colon \mathbb{N} \to \mathbb{N}$ and $\gamma\colon \mathbb{N} \to (0,1)$ such that $\log(T') \leq M \leq \min\left\{(T')^{\gamma/c''}, t^{\alpha/c''}\right\}$ there exist a deterministic algorithm H_f and a probabilistic algorithm R that for every $x \in \{0,1\}^n$ satisfy the following:

1. **Generator.** *When H_f gets input $x \in \{0,1\}^n$ it runs in time $(\bar{T} + (T')^{1/\gamma})^{c'}$ and outputs a set of M-bit strings.*
2. **Reconstruction.** *When R gets input x and oracle access to a function $D\colon \{0,1\}^M \to \{0,1\}$ such that $\Pr_{r\in\{0,1\}^M}[D(r) = 1] \geq 1/M$ but D rejects all the strings that $H_f(x)$ outputs, it runs in time $(t \cdot (T')^{\gamma} \cdot M)^{c'} \cdot \left(d' + n + t_0^{c'}\right)$ and with probability at least $1 - d' \cdot \left(1/(T')^2 + 3 \cdot 2^{-M}\right)$ outputs $f(x)$.*

Moreover, assume that γ is constant, that all the relevant functions (i.e., $T', A, \bar{T}, t_0, t, d'$, and M) are logspace-computable, that the algorithms in Items (1), (2), (3), and (4) of the hypothesis are logspace-uniform circuits of size identical to the stated time bounds, that the circuit in Item (1) has depth $\widetilde{d}$, and that the circuits in the other three items have depth $\operatorname{polylog}(T')$. Then, generator H_f can be computed by a logspace-uniform circuit of size $\operatorname{poly}(T')$ and depth $\widetilde{d} + \operatorname{polylog}(T')$, and the reconstruction algorithm R can be computed by a logspace-uniform probabilistic circuit of size $(t \cdot (T')^{\gamma} \cdot M)^{c'} \cdot \left(d' + n + t_0^{c'}\right)$ and depth $d' \cdot \operatorname{polylog}(T')$.

As mentioned above, the proofs of Proposition 12.4.3 and Proposition 12.4.4 appear in Section 12.4.3 and Section 12.4.4, respectively. The following result is our main construction of the HSG in this section, and its proof amounts to a straightforward combination of the foregoing two results. We state the result for a relatively high reconstruction overhead (*i.e.*, T^{δ} for a constant $\delta > 0$), in which case the HSG and reconstruction are guaranteed to be a logspace-uniform circuit of bounded depth; a statement allowing lower overheads (*i.e.*, $T^{o(1)}$), but without such guarantee, appears in Section 12.5.

Proposition 12.4.5 (a reconstructive targeted HSG)**.** *There exists a universal constant $c > 1$ such that*

the following holds. Let $f\colon \{0,1\}^n \to \{0,1\}^n$ *be computable by logspace-uniform circuits of size* $T(n)$ *and depth* $d(n)$*, let* $\delta > 0$*, and let* $M\colon \mathbb{N} \to \mathbb{N}$ *such that* $c \cdot \log(T(n)) \leq M(n) \leq T(n)^{\delta/c}$*. Then, there exist a deterministic algorithm* H_f *and a probabilistic algorithm* R *that for every* $x \in \{0,1\}^n$ *satisfy the following:*

1. **Generator.** *The generator* H_f *gets input* x *and outputs a set of* M*-bit strings. It is computable by a logspace-uniform circuit of size* $\mathrm{poly}(T)$ *and depth* $(d + \log(T)) \cdot O(\log(T))$*.*
2. **Reconstruction.** *When* R *gets input* x *and oracle access to a function* $D\colon \{0,1\}^M \to \{0,1\}$ *such that* $\Pr_{r\in\{0,1\}^M}[D(r) = 1] \geq 1/M$ *but* D *rejects all the strings that* $H_f(x)$ *prints, it outputs* $f(x)$ *with probability at least* $1 - 1/M$*. The procedure* R *is computable by a logspace-uniform probabilistic circuit of size* $(d + n^c) \cdot T^\delta \cdot M^c$ *and depth* $d \cdot \log^c(T)$*.*

Proof. Let c' and c'' be the universal constants from Proposition 12.4.4, and let $\alpha \in (0,1)$ and k be the universal constants from Proposition 12.4.3. We instantiate the bootstrapping systems for f from Proposition 12.4.3 with parameter $\mu = \delta/5c'$. This yields a system with dimensions $(d' \times T')$ for $d' = O(d \cdot \log(T))$ and $T' = T^k$, where the algorithm that prints each P_i is a logspace-uniform circuit of size $\bar{T} = T^{O(1/\mu)} = \mathrm{poly}(T)$ and depth $d' + O(\log^2(T))$, the algorithms computing the two reductions (*i.e.*, downward self-reducibility and worst-case to rare-case reducibility) are logspace-uniform circuits of size $t = T^{\delta/5c'}$ and depth $\mathrm{polylog}(T)$, and the algorithm that computes P_0 is a logspace-uniform circuit of size $t_0 = \max\{n, t\} \cdot t$ and depth $\mathrm{polylog}(T)$.

We now plug this system into Proposition 12.4.4, using the parameter $\gamma = \delta/(5k \cdot c')$. The hypothesis in Proposition 12.4.4 that $A(n) \leq T(n)$ is satisfied (since the bootstrapping system has alphabet size $A(n) = O(\log(T(n)))$), and the constraint in its conclusion that $k \cdot \log(T) \leq M \leq \min\left\{t^{\alpha/c''}, (T')^{\gamma/c''}\right\} = \min\left\{T^{(\alpha\cdot\delta)/(5c'\cdot c'')}, T^{\delta/(5c'\cdot c'')}\right\} = T^{(\alpha\cdot\delta)/(5c'\cdot c'')}$ is satisfied by our hypothesis that $c \cdot \log(T) \leq M \leq T^{\delta/c}$ for a sufficiently large universal constant $c \geq \max\{k, (5c' \cdot c'')/\alpha\}$.

Note that the generator H_f is indeed computable by a logspace-uniform circuit of size $\mathrm{poly}(\bar{T} + (T')^{1/\gamma}) = \mathrm{poly}(T)$ and depth

$$\widetilde{d} + O(\log^2(T)) = d' + O(\log^2(T)) = O(d \cdot \log(T)) + O(\log^2(T)) \,.$$

The reconstruction algorithm R can be computed by a logspace-uniform probabilistic circuit of size

$$\left(t \cdot T^{k\gamma} \cdot M\right)^{c'} \cdot \left(d' + n + (\max\{n, t\} \cdot t)^{c'}\right) < T^\delta \cdot M^c \cdot (n^c + d)$$

and of depth $d' \cdot \text{polylog}(T') = d \cdot \log^c(T)$, assuming that the universal constant c is sufficiently large. The probability that R errs is at most

$$O(d \cdot \log(T)) \cdot \left(\frac{1}{(T')^2} + 3 \cdot 2^{-M} \right) < 1/M \,. \quad \blacksquare$$

12.4.3 Bootstrapping Systems for Logspace-uniform Bounded-depth Circuits

In this section we prove Proposition 12.4.3. As explained in Section 12.2, our construction mimics the interactive proof system of Goldwasser, Kalai, and Rothblum [GKR15]. Specifically, we arithmetize the circuit layer-by-layer over an appropriate arithmetic setting, and "in between layers" we add additional polynomials that represent a suitable sumcheck protocol, which allows reducing claims about each layer to claims about the preceding layer. Our construction is actually simpler, and does not refer to two auxiliary interactive protocols from their original proof system; this simplification is possible since we only need a bootstrapping system rather than a proof system, and using an additional idea of Goldreich [Gol18].[22]

We first define the notion of a polynomial decomposition of a circuit, then show that any function computable by logspace-uniform bounded-depth circuits has a polynomial decomposition with good parameters, and finally show how a function with such a polynomial decomposition has bootstrapping systems with good parameters.

Definition 12.4.6 (polynomial decomposition of a circuit)**.** *Let C be a circuit that has n input bits, fan-in two, size T, and depth d. For every $x \in \{0,1\}^n$, we call a collection of polynomials a* polynomial decomposition of $C(x)$ *if it meets the following specifications:*

1. **(Arithmetic setting.)** *For some prime $p \leq T$, the polynomials are defined over the prime field $\mathbb{F} = \mathbb{F}_p$. For some integer $h \leq p$, let $H = [h] \subseteq \mathbb{F}$, let m be the minimal integer such that $h^m \geq T$, and let $m' \leq m$ be the minimal integer such that $h^{m'} \geq n$.*
2. **(Circuit-structure polynomial.)** *For each $i \in [d]$, let $\Phi_i \colon H^{3m} \to \{0,1\}$ be the function such that $\Phi_i(\vec{w}, \vec{u}, \vec{v}) = 1$ if and only if the gate in layer i indexed by $\vec{w}$ is fed by the gates in layer $i-1$ indexed by $\vec{u}$ and $\vec{v}$. (If one of the elements $\vec{w}, \vec{u}, \vec{v}$ does not index a valid gate in the corresponding layer, then*

[22] Let us spell out the differences, for readers who are familiar with [GKR15]. First, following the simplification idea of Goldreich [Gol18], we avoid an auxiliary protocol from their original system intended to compute the circuit-structure polynomial. Secondly, we avoid an auxiliary protocol that reduces verification of a pair of points at each step to verification of a single point at each step; this is because in bootstrapping systems we are fine with verifying a set of points at each step (in contrast to proof systems, where this yields an exponential blow-up in the verification time). Lastly, we warn readers that our indexing of the layers below is reverse order compared to the indexing in their paper (*i.e.*, we index from bottom to top rather than from top to bottom).

Φ_i outputs zero.) The polynomial $\hat{\Phi}_i \colon \mathbb{F}^{3m} \to \mathbb{F}$ can be any extension of Φ_i.

3. **(Input polynomial.)** *Let $\alpha_0 \colon H^m \to \{0,1\}$ represent the string $x0^{h^m-n}$, and let $\hat{\alpha}_0 \colon \mathbb{F}^m \to \mathbb{F}$ be defined by*

$$\hat{\alpha}_0(\vec{w}) = \sum_{\vec{z} \in H^{m'} \times \{0\}^{m-m'}} \delta_{\vec{z}}(\vec{w}) \cdot \alpha_0(\vec{z}) \ ,$$

where $\delta_{\vec{z}}$ is Kronecker's delta function (i.e., $\delta_{\vec{z}}(\vec{w}) = \prod_{j \in [m]} \prod_{a \in H \setminus \{z_j\}} \frac{w_j - a}{z_j - a}$).

4. **(Layer polynomials.)** *For each $i \in [d]$, let $\alpha_i \colon H^m \to \{0,1\}$ represent the values of the gates at the i^{th} layer of C in the computation of $C(x)$ (with zeroes in locations that do not index valid gates), and let $\hat{\alpha}_i \colon \mathbb{F}^m \to \mathbb{F}$ be defined by*

$$\hat{\alpha}_i(\vec{w}) = \sum_{\vec{u},\vec{v} \in H^m} \hat{\Phi}_i(\vec{w},\vec{u},\vec{v}) \cdot (1 - \hat{\alpha}_{i-1}(\vec{u}) \cdot \hat{\alpha}_{i-1}(\vec{v})) \ .$$

5. **(Sumcheck polynomials.)** *For each $i \in [d]$, let $\hat{\alpha}_{i,0} \colon \mathbb{F}^{3m} \to \mathbb{F}$ be the polynomial*

$$\hat{\alpha}_{i,0}(\vec{w},\sigma_1,\ldots,\sigma_{2m}) = \hat{\Phi}_i(\vec{w},\sigma_{1,\ldots,m},\sigma_{m+1,\ldots,2m}) \cdot (1 - \hat{\alpha}_{i-1}(\sigma_{1,\ldots,m}) \cdot \hat{\alpha}_{i-1}(\sigma_{m+1,\ldots,2m})) \ ,$$

and for every $j \in [2m-1]$, let $\hat{\alpha}_{i,j} \colon \mathbb{F}^{3m-j} \to \mathbb{F}$ be the polynomial

$$\hat{\alpha}_{i,j}(\vec{w},\sigma_1,\ldots,\sigma_{2m-j}) = \sum_{\sigma_{2m-j+1},\ldots,\sigma_{2m} \in H} \hat{\Phi}_i(\vec{w},\sigma_{1,\ldots,m},\sigma_{m+1,\ldots,2m}) \cdot (1 - \hat{\alpha}_{i-1}(\sigma_{1,\ldots,m}) \cdot \hat{\alpha}_{i-1}(\sigma_{m+1,\ldots,2m})) \ ,$$

where $\sigma_{k,\ldots,k+r} = \sigma_k,\sigma_{k+1},\ldots,\sigma_{k+r}$. Lastly, let $\hat{\alpha}_{i,2m}(\vec{w}) = \hat{\alpha}_i(\vec{w})$.

We now show that any function computable by logspace-uniform circuits of bounded depth has a polynomial decomposition whose reducibility algorithms are very efficient.

Proposition 12.4.7 (polynomial decompositions of logspace-uniform circuits using universal circuits)**.** *There exist two universal constants $c, c' \in \mathbb{N}$ such that the following holds. Let $\{C_n\}_{n \in \mathbb{N}}$ be a logspace-uniform family of circuits of size $T(n)$ and depth $d(n)$, and let $\gamma \colon \mathbb{N} \to (0,1)$ be a logspace-computable function such that $T(n)^{\gamma(n)} \geq \log(T(n))$. Then, there exists a logspace-uniform family of circuits $\{C'_n\}_{n \in \mathbb{N}}$ of size $T'(n) = O(T(n)^c)$ and depth $d'(n) = O(d(n) \cdot \log(T(n)))$ that computes the same Boolean function as $\{C_n\}$ such that for every $x \in \{0,1\}^n$ there exists a polynomial decomposition of $C'_n(x)$ satisfying:*

1. **(Arithmetic setting.)** *The polynomials are defined over $\mathbb{F}_p$, where p is the smallest prime in the interval $[T^{\gamma \cdot c}, 2T^{\gamma \cdot c}]$. Let $H = [h] \subseteq \mathbb{F}$, where h is the smallest power of two of magnitude at least $T^{\gamma/6}$, and let m be the minimal integer such that $h^m \geq 2T^c$.*
2. **(Faithful representation.)** *For every $i \in [d'(n)]$ and $\vec{w} \in H^m$ representing a gate in the i^{th} layer it holds that $\hat{\alpha}_i(\vec{w})$ is the value of the gate $\vec{w}$ in $C'_n(x)$.*
3. **(Layer downward self-reducibility.)** *There is a logspace-uniform oracle circuit of size $\max\{n, h\} \cdot h^{c'}$ and depth $O(\log^2(T))$ that, when given oracle access to $x \in \{0,1\}^n$, computes the function $\hat{\alpha}_0$. Also, there is a logspace-uniform oracle circuit of size $h^{c'}$ and depth $O(\log^2(T))$ that computes $\hat{\alpha}_{i,0}$ while querying $\hat{\alpha}_{i-1}$ twice on inputs in H^m. Lastly, $\hat{\alpha}_{i,2m} \equiv \hat{\alpha}_i$.*
4. **(Sumcheck downward self-reducibility.)** *There is a logspace-uniform oracle circuit of size $h^{c'}$ and depth $O(\log(T))$ that gets input $\vec{w} \in \mathbb{F}^m$ and $(\sigma_1, ..., \sigma_{2m}) \in \mathbb{F}^{2m}$ and $j \in [2m]$ and oracle access to $\hat{\alpha}_{i,j-1}$ and outputs $\hat{\alpha}_{i,j}(\vec{w}, \sigma_1, ..., \sigma_{2m-j})$.*
5. **(Sample-aided worst-case to rare-case reducibility.)** *For each $i \in [d'(n)]$ and $j \in [2m]$, the Boolean function representing $\hat{\alpha}_{i,j}$ is sample-aided worst-case to ρ-rare-case reducible with error 2^{-h} by logspace-uniform circuits of size $h^{c'}$ and depth $\mathrm{polylog}(T)$, where $\rho = h^{-c} \cdot \mathrm{polylog}(T)$. The same claim holds for $\hat{\alpha}_0$.*

Proof. Following [Gol18], we consider a circuit C'_n that first computes a description of C_n (represented as a $T(n) \times T(n) \times T(n)$ tensor) and then computes the `Eval` function $(\langle C_n \rangle, x) \mapsto C_n(x)$. The construction of C'_n in [Gol18] essentially suffices for our purposes, but we will need to use a property of this construction that was not explicitly stated in [Gol18] (specifically, the fact that the adjacency relation of C'_n is computable in small space). That is:

Claim 12.4.8. *There exists a circuit C'_n as above of depth $d'(n) = O(\log^2(T(n)) + d(n) \cdot \log(T(n)))$ and size $T' = 2^{c \cdot \lceil \log(T(n)) \rceil}$ (for some universal integer $c > 1$) such that the layered adjacency relation function $\Phi' \colon [d'] \times \{0,1\}^{3\log(T')} \to \{0,1\}$ of C'_n can be decided by a formula that can be constructed in time $\mathrm{polylog}(T(n))$ and space $O(\log(T(n)))$.*

Proof. We follow the construction of C'_n and its analysis in [Gol18, Sections 3.3 and 3.4.2], while tracking the relevant changes. The original construction is stated with respect to a parameter n such that the input length is n and C_n is a circuit of size $\mathrm{poly}(n)$ and depth $d(n)$ whose adjacency relation can be decided in time $O(\log(n))$. However, this construction scales smoothly to the case where C_n is of size $T(n)$ and depth $d(n)$ and the adjacency relation can be decided in time $O(\log(T(n)))$, by considering C_n as a circuit over $T(n)$ bits that ignores all but the first n bits. Also,

without loss of generality, we can assume that $T'(n)$ is a power of two.

Turning to the adjacency relation Φ', in [Gol18] it is only stated that Φ' can be computed by a formula that can be constructed in time $\text{polylog}(T(n))$. To see that the algorithm constructing the formula only uses space $O(\log(T(n)))$, note that each of the three stages of the construction of C'_n yields a very simple description of the adjacency relations in the corresponding part of C'_n: The first step consists of constructing the matrix of transitions between instantaneous configurations of the $O(\log(T))$-space machine that prints a description of C_n (the gates in this step have no incoming wires); the second steps consists of squaring the foregoing matrix for $O(\log(T(n)))$ times; and the third step consists of computing the `Eval` function with input $(\langle C_n \rangle, x)$. (See [Gol18, Section 3.4.2] for explicit descriptions of the adjacency relations in the two latter parts.) □

Since $d' \leq T'$, we can extend Φ' to a function $\{0,1\}^{4\log(T')} \to \{0,1\}$ without noticeably affecting its complexity. We denote the size of the formula for Φ by $s' = \text{polylog}(T)$, and for every $i \in [d']$, we denote by $\Phi_i(\cdot) = \Phi'(i,\cdot)$ the i^{th} "slice" of Φ' (that is, $\Phi_i(\vec{w},\vec{u},\vec{v}) = \Phi'(i,\vec{w},\vec{u},\vec{v})$.

Recall that p is the smallest prime in the interval $[T^{\gamma\cdot c}, 2T^{\gamma\cdot c}]$, and that h is the smallest power of two of magnitude at least $T^{\gamma/6}$. Note that given input 1^n, both p and h can be found in space $O(\log(T))$. For every $x \in \{0,1\}^n$, we consider the polynomial decomposition of $C'_n(x)$ with $h \leq p \leq T$. The claim about faithful representation holds for any valid arithmetic extension of the Φ_i's. Thus, to complete the description of the decomposition and conclude the proof, we now specify the arithmetization $\hat{\Phi}_i$ of each Φ_i, and then prove our claims regarding downward self-reducibility and worst-case to rare-case reducibility.

Arithmetization of the Φ_i's. Our goal now is to arithmetize each Φ_i as a polynomial $\mathbb{F}^{3m} \to \mathbb{F}$ that has low degree and is efficiently computable, as follows:

Claim 12.4.9. *For $i \in [d']$ there exists $\hat{\Phi}_i \colon \mathbb{F}^{3m} \to \mathbb{F}$ that satisfies the following:*

1. *For every $(\vec{w},\vec{u},\vec{v}) = z_1,...,z_{3m} \in H^{3m}$ we have that $\hat{\Phi}_i(\vec{w},\vec{u},\vec{v}) = 1$ if gate $\vec{w}$ in the i^{th} layer of C'_n is fed by gates $\vec{u}$ and $\vec{v}$ in the $(i-1)^{th}$ layer of C'_n, and $\hat{\Phi}_i(\vec{w},\vec{u},\vec{v}) = 0$ otherwise.*
2. *The degree of $\hat{\Phi}_i$ is at most $h \cdot \text{polylog}(T)$.*
3. *For a universal constant $c_1 > 1$, there exists a logspace-uniform circuit of size h^{c_1} and depth $O(\log^2(T))$ that computes $\hat{\Phi}_i$.*

Proof. We think of Φ_i as a function $\mathbb{F}_2^{3\log(T')} \to \mathbb{F}_2$, and note that it is computable by an *arithmetic formula* (over $\mathbb{F}_2$), whose structure mimics the one of the original Boolean formula (*i.e.*, each gate

w in the original formula computes the NAND of two sub-formulas u and v, and thus w can be replaced by the expression $\hat{w} = 1 - \hat{u} \cdot \hat{v}$). The same arithmetic formula can be used to compute a polynomial $\Phi'_i \colon \mathbb{F}^{3\log(T')} \to \mathbb{F}$ of degree $\mathrm{poly}(s')$ that agrees with Φ_i on $\mathbb{F}_2^{3\log(T')}$. [23]

Next, let $\ell = \log(h)$, and for every $j \in [\ell]$ consider the function $\pi_j \colon H \to \{0,1\}$ such that $\pi_j(a)$ is the j^{th} bit in the binary representation of the integer a. Note that there is a polynomial $\hat{\pi}_j \colon \mathbb{F} \to \mathbb{F}$ of degree at most h that agrees with π_j on H. Finally, let $\hat{\Phi}_i \colon \mathbb{F}^{3m} \to \mathbb{F}$ such that

$$\hat{\Phi}_i(z_1, ..., z_{3m}) = \Phi'_i(\hat{\pi}_1(z_1), ..., \hat{\pi}_\ell(z_1), ..., \hat{\pi}_1(z_{3m}), ..., \hat{\pi}_\ell(z_{3m})) \ . \tag{12.4.1}$$

By definition, when $\hat{\Phi}_i$ is given input $(\vec{w}, \vec{u}, \vec{v}) \in H^{3m}$, the π_j's project each of the three inputs to a bit-string that is the index of a gate, and the arithmetic formula Φ'_i computes Φ_i on the corresponding gates. Also, the degree $\Delta \stackrel{\text{def}}{=\!=} \deg(\hat{\Phi}_i)$ satisfies $\Delta \leq h \cdot \mathrm{poly}(s') = h \cdot \mathrm{polylog}(T)$.

Lastly, the algorithm for $\hat{\Phi}_i$ gets input $z_1, ..., z_{3m}$, computes $\hat{\pi}_j(w_k)$ for each $k \in [3m]$ and $j \in [\ell]$, and evaluates the arithmetic formula for Φ'_i on the resulting sequence of $3m \cdot \ell$ elements from $\mathbb{F}$. Now, recall that addition of h elements in $\mathbb{F}_p$ and iterated multiplication of h elements in $\mathbb{F}_p$ are computable by logspace-uniform circuits of depth $O(\log(p))$ and size $\mathrm{poly}(h \cdot \log(p)) = \mathrm{poly}(h)$ (see Lemma 12.8.1)). Thus, using the formula $\hat{\pi}_j(u) = \sum_{a \in H} \pi_j(a) \cdot \prod_{a' \in H \setminus \{a\}} \frac{u - a'}{a - a'}$, the polynomial $\hat{\pi}_j$ can be computed by a logspace-uniform circuit of depth $O(\log(p)) = O(\log(T))$ and size $\mathrm{poly}(h)$. Also, since the formula Φ_i is logspace-uniform and is of size $\mathrm{polylog}(T)$ and depth $O(\log(T))$, we can replace each gate in Φ_i by a logspace-uniform circuit of depth $O(\log(p)) = O(\log(T))$ and size $\mathrm{polylog}(p) = \mathrm{polylog}(T)$ for the corresponding arithmetic operation and obtain a logspace-uniform circuit of size $\mathrm{polylog}(T)$ and depth $O(\log^2(T))$ for the arithmetic version of Φ_i. The final algorithm for $\hat{\Phi}_i$ is thus a logspace-uniform circuit of size

$$\ell \cdot m \cdot \mathrm{poly}(h) + \mathrm{polylog}(T) \leq h^{c_1} \ ,$$

and depth $O(\log^2(T))$, where we relied on the fact that $m \leq \log(T) \leq \mathrm{poly}(h)$ and that $c_1 > 1$ is a universal constant (which depends on the size of the logspace-uniform circuits for iterated multiplication). □

[23] Note that the domain size of Φ'_i is superpolynomial in T, but none of our algorithms will explicitly construct the entire truth-table of Φ'_i at any point.

Layer downward self-reducibility. Recall that, by Definition 12.4.6, m' is the minimal integer such that $h^{m'} \geq n$, and that $\delta_{\vec{z}}$ is Kronecker's delta function. By the definition of $\hat{\alpha}_0$, to compute it we can first enumerate (in parallel) over all elements in $H^{m'}$, then enumerate over $j \in [m]$ in parallell; for each element and j perform $(h-1)$ multiplication operations on our input, and then multiply the m results; and finally sum up the $h^{m'}$ results. Relying on Lemma 12.8.1, this can be done by logspace-uniform circuits of size

$$h^{m'} \cdot m \cdot \mathrm{poly}(h) + O(h^{m'} \cdot \log(p \cdot h^{m'})) \leq \max\{n, h\} \cdot h^{c_2}$$

and of depth $O(\log(p)) + O(\log(p) \cdot \log(n)) = O(\log^2(T))$, for some universal constant $c_2 > 1$. (We again relied on the fact that $m \leq \log(T) \leq \mathrm{poly}(h)$.)

To compute $\hat{\alpha}_{i,0}$ with an oracle to $\hat{\alpha}_{i-1}$, we get inputs $(\vec{w}, \vec{u}, \vec{v})$ and we need to compute $\hat{\Phi}_i(\vec{w}, \vec{u}, \vec{v})$ and perform two oracle calls and arithmetic operations; by Claim 12.4.9, we can do so with a logspace-uniform circuit of size $h^{c_1} + O(\log(p))$ and depth $O(\log^2(T))$. The identity $\hat{\alpha}_{i,2m} \equiv \hat{\alpha}_i$ is by definition.

Sumcheck downward self-reducibility. The algorithm for $\hat{\alpha}_{i,j}$ follows from the fact that

$$\hat{\alpha}_{i,j}(\vec{w}, \sigma_1, ..., \sigma_{2m-j}) = \sum_{\sigma_{2m-j+1} \in H} \hat{\alpha}_{i,j-1}(\vec{w}, \sigma_1, ..., \sigma_{2m-j+1}) \ ,$$

where the RHS can be computed by adding the answers to h oracle queries to $\hat{\alpha}_{i,j-1}$. We construct a uniform circuit that takes input j and $\vec{w}, \sigma_1, ..., \sigma_{2m}$, enumerates (in parallel) all $\sigma \in H$, uses each fixed σ along with the first $2m-j$ elements $\sigma_1, ..., \sigma_{2m-j}$ to issue an oracle query $\sigma_1, ..., \sigma_{2m}, \sigma$ to $\hat{\alpha}_{i,j-1}$, and adds the h results. This can be done by logspace-uniform circuits of size h^{c_3} and depth $O(\log(p)) = O(\log(T))$, where $c_3 > 1$ is a universal constant.

Worst-case to rare-case reducibility. We claim that for every $i \in [d']$, the degree of $\hat{\alpha}_i$ and of $\hat{\alpha}_{i,j}$, is at most $\Delta \stackrel{\text{def}}{=\!=} h \cdot \mathrm{polylog}(T)$. To see this, for $i \in [d']$, note that $\hat{\alpha}_i$ feeds its input $\vec{w}$ only to the function $\hat{\Phi}_i$, and recall that $\deg(\hat{\Phi}_i) = h \cdot \mathrm{polylog}(T) \leq \Delta$. An identical argument shows that for every $i \in [d']$ and $j \in [2m]$, the degree of $\hat{\alpha}_{i,j}$ is at most Δ. Now, note that

$$\Delta / |\mathbb{F}| \leq h \cdot \mathrm{polylog}(T) / (T')^{\gamma} \leq h \cdot T^{-\gamma \cdot c} \cdot \mathrm{polylog}(T) \ .$$

By Proposition 12.3.10, the Boolean function associated with $\hat{\alpha}_i$ is sample-aided worst-case to ρ-rare-case reducible, for

$$\rho = 10\sqrt{\Delta/|\mathbb{F}|} < T^{-(\gamma\cdot c)/2}\cdot\sqrt{h}\cdot\mathrm{polylog}(T) < T^{-c\gamma/6}\cdot\mathrm{polylog}(T)\ ,$$

which is upper-bounded by $h^{-c}\cdot\mathrm{polylog}(T)$. The reduction can be computed by logspace-uniform circuits of size $\mathrm{poly}(m, T^{\gamma\cdot c}) \leq h^{c_4}$ using at most h^{c_4} samples (for some universal $c_4 \geq 1$) and depth $\mathrm{polylog}(T)$ and has error $2^{-|\mathbb{F}|} < 2^{-h}$.

Wrapping-up. To conclude we define the constant c' to satisfy $c' \geq \max\{c_1+1, c_2, c_3, c_4\}$. ■

We now prove Proposition 12.4.3, which is our construction of bootstrapping systems. The proof amounts to transforming the polynomial decomposition from Proposition 12.4.7 into bootstrapping systems, by ordering the layer polynomials and the sumcheck polynomials in an appropriate ascending order (and carefully verifying the parameters of the resulting construction).

Proof of Proposition 12.4.3. Let C_n be the logspace-uniform circuit for f on inputs of length n, which is of depth $d = d(n)$ and of size $T = T(n)$. Let c and c' be the universal constants from Proposition 12.4.7, and let C'_n be the corresponding logspace-uniform circuit from Proposition 12.4.7, which has depth $d'_0 = O(d\cdot\log(T))$ and size $T' = O(T)^c < T^k$ (for any constant $k > c$). For $\gamma = 5\mu/c'$, consider the corresponding polynomial decomposition of C'_n from Proposition 12.4.7; the requirement that $T^\gamma > \log(T)$ is satisfied by our hypothesis that $T^\mu \geq \log^{1/\alpha}(T)$ and by a choice of sufficiently small $\alpha \leq 1/c'$. Note that $|\mathbb{F}| \leq \mathrm{poly}(T)$ and $h = O(T^{5\mu/6c'})$ and $m = O(1/\mu)$. Denote $t = h^{c'} = O(T^{5\mu/6}) < T^\mu$.

Let $\{\hat{\alpha}_i\colon \mathbb{F}^m \to \mathbb{F}\}_{i\in\{0,\ldots,d'_0\}}$ and $\{\hat{\alpha}_{i,j}\colon \mathbb{F}^{3m-j} \to \mathbb{F}\}_{i\in[d'_0],j\in\{0,\ldots,2m-1\}}$ be the corresponding layer polynomials and sumcheck polynomials in the decomposition, respectively. We now view each $\hat{\alpha}_i$ and $\hat{\alpha}_{i,j}$ as a Boolean function mapping $3m\cdot\log(|\mathbb{F}|) = O(\log(T))$ bits to $|\mathbb{F}| = O(\log(T))$ bits (note that we think of all these functions as having the same domain, meaning that some of the Boolean functions will ignore a suffix of their input).

Defining the bootstrapping system. The bootstrapping system has $d' = d'_0\cdot(2m+1) = O(d'_0/\mu)$ layers, each of length $|\mathbb{F}|^{3m} = \mathrm{poly}(T)$ and over alphabet $\mathbb{F}$, in addition to the base layer P_0. The base layer P_0 is the truth-table of the function $\hat{\alpha}_0$, and for each $(i,j) \in [d'_0]\times\{0,\ldots,2m\}$, the $(i,j)^{th}$ layer is the truth-table of the function $\hat{\alpha}_{i,j}$. The layers are ordered first according to an increasing

value of i, and then according to an increasing order of j; for example, the following three layers are listed in ascending order: $(1, 2m-1)$, then $(1, 2m)$, and then $(2, 0)$.

Printing each layer. Given $x \in \{0,1\}^n$ and an index $(i,j) \in [d_0'] \times \{0, ..., 2m\}$, we print the layer corresponding to (i,j) as follows. We first compute the values of all the gates of C_n', which can be done by a logspace-uniform circuit of size $\text{poly}(T)$ and depth d_0'. This gives us the values of $\hat{\alpha}_{i-1}$ on the set H^m (because these are the values of the gates in the $(i-1)^{th}$ layer of $C_n'(x)$, which in the case of $i = 1$ are just the bits of x). We compute the truth-table of $\hat{\alpha}_{i,0}$, using the layer self-reducibility algorithm and our oracle access to values of $\hat{\alpha}_{i-1}$ on H^m; this can be done by a logspace-uniform circuit of size $t \cdot \text{poly}(T) = \text{poly}(T)$ and depth $O(\log^2(T))$. Then, iteratively for $j' = 1, ..., j$, we compute the truth-table of $\hat{\alpha}_{i,j'}$, using the sumcheck self-reducibility algorithm and oracle access to $\hat{\alpha}_{i,j'-1}$; this can be done by a logspace-uniform circuit of size $O(m) \cdot t \cdot \text{poly}(T) = \text{poly}(T)$ and depth $O(m \cdot \log(T)) = O(\log(T)/\mu)$. Thus, the final algorithm that prints the $(i,j)^{th}$ layer is a logspace-uniform circuit of size $\text{poly}(T)$ and depth $d_0' + O(\log^2(T))$.

Base case. Given $x \in \{0,1\}^n$ and $k \in [\text{poly}(T)]$, we can use the algorithm for $\hat{\alpha}_0$ and our access to x to output $P_0(x)_k$. This can be done by a logspace-uniform circuit of size $\max\{n,h\} \cdot t \leq \max\{n,t\} \cdot t$ and depth $O(\log^2(T))$.

Downward self-reducibility. We are given $x \in \{0,1\}^n$ and $(i,j) \in [d_0'] \times \{0, ..., 2m\}$. The proof differs according to whether $j = 0$ or $j \in [2m]$:

1. If $j = 0$, then we need to compute $\hat{\alpha}_{i,0}$ with oracle access to $\hat{\alpha}_{i-1,2m} = \hat{\alpha}_{i-1}$. By layer self-reducibility, we can do so by a logspace-uniform circuit of size t and depth $O(\log^2(T))$.
2. If $j \in [2m]$, we need to compute $\hat{\alpha}_{i,j}$ with oracle access to $\hat{\alpha}_{i,j-1}$. Using sumcheck self-reducibility, this can be done by a logspace-uniform circuit of size t and depth $O(\log(T)) < O(\log^2(T))$.

Worst-case to rare-case reduction. By Proposition 12.4.7, each of the P_i's is the truth-table of a function that is worst-case to ρ_0-rare-case self-reducible, where $\rho_0 = h^{-c} \cdot \text{polylog}(T)$, with error 2^{-h} and using logspace-uniform circuits of size t and depth $\text{polylog}(T)$. This value of ρ_0 suffices by choosing a sufficiently small $\alpha \leq 1/c'$, which guarantees that $t^\alpha \leq h \leq h^c$. ∎

12.4.4 From Bootstrapping Systems to a Targeted HSG

In this section we prove Proposition 12.4.4. We will directly prove the "moreover" part, under the stronger hypotheses (referring to logspace-uniformity and depth bounds); the proof of the basic claim (without the "moreover" part) is essentially identical, just ignoring depth bounds and logspace-uniformity. For convenience, we will denote the dimensions of the bootstrapping system in our hypothesis by $d \times T$ instead of $d' \times T'$.

For the proof we will need the following standard tools: The Nisan-Wigderson PRG [NW94] and the Goldreich-Levin [GL89] list-decoding algorithm for the Hadamard code. In the result statements below we assert that the reconstruction algorithm for the Nisan-Wigderson PRG is a uniform learning algorithm, following the classical observation of [IW98], and moreover assert that all the associated algorithms can be implemented by logspace-uniform circuits of bounded depth. The only non-standard thing in the latter efficiency requirement is that the circuits are logspace-uniform; we meet this requirement by constructing combinatorial designs in logspace (following [KvM02, HR03] and an idea attributed to Salil Vadhan), so that the logspace algorithm that constructs the uniform circuit can "hardwire" the design into the circuit. Proofs of the two result statements appear in Section 12.7.1 and Section 12.7.4, respectively.

Theorem 12.4.10 (the NW PRG with reconstruction as a learning algorithm)**.** *There exists a universal constant $c > 1$, an oracle machine G, and a probabilistic oracle machine R_0, such that the following holds:*

1. **Generator:** *When given input $(1^\ell, 1^M, \gamma)$ such that $M \leq 2^{(\gamma/c)\cdot\ell}$ oracle access to $h\colon \{0,1\}^\ell \to \{0,1\}$, the machine G runs in time $2^{c\cdot\ell/\gamma}$ and outputs a set of strings in $\{0,1\}^M$. Moreover, if γ is constant and ℓ, M are sufficiently large, then G can be implemented by logspace-uniform oracle circuits of size $2^{c\cdot\ell/\gamma}$ and depth $O(\log(M,\ell))$.*
2. **Reconstruction:** *When given input $(1^\ell, 1^M, \gamma)$ and oracle access to a $(1/M)$-distinguisher D for $G^h(1^\ell, 1^M, \gamma)$ and to h, the machine R_0 runs in time $M^c \cdot 2^{\gamma\cdot\ell}$, makes non-adaptive queries, and outputs with probability at least $1 - 2^{-3M}$ an oracle circuit that computes h on $1/2 + M^{-3}$ of the inputs when given access to D. The circuit that R_0 outputs has depth $\mathrm{polylog}(M,\ell)$ and makes just one oracle query. Moreover, if γ is a constant and ℓ, M are sufficiently large, then R_0 can be implemented by a logspace-uniform probabilistic oracle circuit of size $M^c \cdot 2^{\gamma\cdot\ell}$ and depth $\mathrm{polylog}(M,\ell)$ that makes non-adaptive queries.*

Theorem 12.4.11 (list-decoding the Hadamard code [GL89])**.** *For any time-computable $a\colon \mathbb{N} \to \mathbb{N}$ satisfying $a(\ell_0) \leq \ell_0$ and $\varepsilon\colon \mathbb{N} \to (0, 1/2)$ there exists a transformation* `Had` *that maps any function*

$g\colon \{0,1\}^{\ell_0} \to \{0,1\}^{a(\ell_0)}$ *to a Boolean function* $\mathtt{Had}(g)\colon \{0,1\}^{\ell_0+a(\ell_0)} \to \{0,1\}$ *such that the following holds.*

1. **Encoding:** *For every* $x \in \{0,1\}^{\ell_0}$ *and* $z \in \{0,1\}^{a(\ell_0)}$ *it holds that* $\mathtt{Had}(g)(x,z) = \langle g(x), z\rangle = \oplus_{i\in[a(\ell_0)]} g(x)_i \cdot z_i$.
2. **Decoding:** *There exists a logspace-uniform circuit* $\mathtt{GL}$ *of size* $\mathrm{poly}(\ell_0/\varepsilon)$ *and depth* $\mathrm{polylog}(\ell_0/\varepsilon)$ *that gets input* 1^{ℓ_0} *and outputs a probabilistic oracle circuit* C *of depth* $\mathrm{polylog}(\ell_0/\varepsilon)$ *that satisfies the following. For every oracle* $\widetilde{\mathtt{Had}}(g)$ *that agrees with* $\mathtt{Had}(g)$ *on* $1/2+\varepsilon$ *of the inputs, the probability over the random coins of* C *and a uniform choice of* $x \in \{0,1\}^{\ell_0}$ *that* $C^{\widetilde{\mathtt{Had}}(g)}(x) = g(x)$ *is at least* $\mathrm{poly}(\varepsilon)$.

We now describe the generator H_f and then later the reconstruction algorithm R. Throughout the description, the algorithms that we describe will be logspace-uniform circuits of bounded depth. We note that in intermediate stages of their execution, these algorithms will compute descriptions of certain circuits and then simulate these circuits; we will always bound the depth of the circuits whose descriptions are computed, in order to verify that the algorithm can then indeed simulate these circuits in small depth.[24]

The hitting-set generator H_f. Given $x \in \{0,1\}^n$, the algorithm H_f enumerates in parallel over $i \in [d]$ and for each i computes the string $P_i(x)$. Thinking of $P_i(x) \in [A]^T$ as a truth-table of a function $p_i\colon \{0,1\}^{\log(T)} \to [A]$, the algorithm computes the truth-table of the function $h_i = \mathtt{Had}(p_i)$, where $\mathtt{Had}$ is the encoding from Theorem 12.4.11. Note that h_i is a Boolean function over $\ell = \log(T) + \log(A) = (1+o(1))\cdot\log(T)$ bits, which means that its truth-table is of size $T^{1+o(1)}$. Finally, for the parameter $\gamma = \gamma(n) \in (0,1)$ chosen in our statement, the algorithm uses the generator G from Theorem 12.4.10 with parameters $(1^\ell, 1^M, \gamma)$ and with access to the function h_i, to output a set of strings of length M. (The hypothesis of Theorem 12.4.10 is satisfied by our assumption that $M \le T^{\gamma/c''}$ for a sufficiently large constant $c'' \ge 1$.)

To bound the complexity of H_f, recall that $P_i(x)$ can be computed by a logspace-uniform circuit of depth $\widetilde{d}$ and size $\bar{T}$. Now, since each entry in the truth-table of h_i can be computed in time $\mathrm{polylog}(A)$ with non-adaptive access to the truth-table of p_i (*i.e.*, to the string P_i), we can compute the entire truth-table of h_i by a logspace-uniform circuit of size $\mathrm{poly}(\bar{T})$ and depth $\mathrm{polylog}(A) \le \mathrm{polylog}(T)$. Finally, the generator from Theorem 12.4.10 can be computed by a

[24] The potential issue here is that a logspace-uniform circuit C of bounded depth can potentially compute a description of a circuit C' of very large depth (in which case C would not be able to simulate C' in bounded depth). However, we will make sure that this does not happen in our specific constructions.

logspace-uniform circuit of size $T^{O(1/\gamma)}$ and depth $O(\log(M \cdot \ell)) = O(\log(T))$. Thus, for each $i \in [d]$ the corresponding output string can be printed by a logspace-uniform circuit of size $\text{poly}(T^{1/\gamma}, \bar{T})$ and depth $\widetilde{d} + \text{polylog}(T)$, and multiplying the size by d (since we enumerate over $i \in [d]$ in parallel) we still get a circuit of size at most $(T^{1/\gamma} + \bar{T})^{c'}$ for a sufficiently large universal constant c'.

The reconstruction algorithm R. Let $D\colon \{0,1\}^M \to \{0,1\}$ be such that $\Pr_{r \in \{0,1\}^M}[D(r) = 1] \geq 1/M$, but for every $i \in [d(n)]$ it holds that D rejects all the strings in the output set of $G^{h_i}(1^\ell, 1^M, \gamma)$. The algorithm R gets input $x \in \{0,1\}^n$ and iteratively, for $i = 0, ..., d(n)$, it finds a small circuit that computes the function whose truth-table is $P_i(x)$. We will now describe the procedure, while accounting both for the complexity of implementing each iteration, and for the complexity of the circuit that each iteration produces.

First, the algorithm constructs a circuit C_0 that computes the function whose truth-table is P_0; by our assumption about P_0, this can be done by a logspace-uniform circuit of size t_0 and depth $\text{polylog}(T)$, and also C_0 has size $\tilde{O}(t_0)$ and depth $\text{polylog}(T)$. Then, for $i \in [d(n)]$, we start the i^{th} iteration with an oracle circuit C_{i-1} of size $|C_{i-1}|$ and depth $\mathsf{Depth}(C_{i-1})$ such that C_{i-1}^D computes p_{i-1}. The following lemma shows that, given C_{i-1} and oracle access to D, we can efficiently compute a small circuit C_i that computes p_i:

Lemma 12.4.12 (the i^{th} iteration: moving from a circuit for p_{i-1} to a circuit for p_i)**.** *There exists a universal constant $c_0 > 1$ such that the following holds. Given the circuit C_{i-1} and oracle access to D, we can compute with probability at least $1 - 1/T^2 - 3 \cdot 2^{-M}$ an oracle circuit C_i that computes p_i when given oracle access to D. This can be done by a logspace-uniform probabilistic circuit of size $t^2 \cdot T^{2\gamma} \cdot (M \cdot |C_{i-1}|)^{c_0}$ and depth $(\log(T) \cdot \mathsf{Depth}(C_{i-1}))^{c_0}$, and the circuit C_i is of size $\tilde{O}(t) \cdot M^{c_0} \cdot T^{2\gamma}$ and depth $\log(T)^{c_0}$.*

Proof. The algorithm will consist of three steps. Loosely speaking, these correspond to the Nisan-Wigderson reconstruction algorithm (as in Theorem 12.4.10); to the Goldreich-Levin list-decoding algorithm (as in Theorem 12.4.11), coupled with a process of weeding the output list to find a single "good" circuit; and to applying worst-case to rare-case self-reducibility. These three steps are depicted in the following three claims.

Claim 12.4.13 (the [NW94] reconstruction)**.** *Given the circuit C_{i-1} and oracle access to D, we can compute with probability at least $1 - 2^{-M}$ an oracle circuit $C_{i,1}$ such that $C_{\mathtt{NW}} = C_{i-1}^D$ computes h_i correctly on $1/2 + M^{-3}$ of the inputs. This step can be implemented by a logspace-uniform probabilistic circuit*

of size $\mathrm{poly}(M, |C_{i-1}|) \cdot T^{2\gamma} \cdot t$ *and depth* $\mathrm{poly}(\log(T), \mathsf{Depth}(C_{i-1}))$, [25] *and the circuit* $C_{i,1}$ *is of size* $\mathrm{poly}(M) \cdot T^{2\gamma}$ *and of depth* $\mathrm{polylog}(M)$.

Proof. We invoke the reconstruction algorithm from Theorem 12.4.10 for h_i with input $(1^\ell, 1^M, \gamma)$. We answer its oracle queries to a distinguisher using D, and we answer its queries to h_i using the fact that $h_i(x, z) = \langle P_i(x), z \rangle$, the downward self-reducibility algorithm for P_i, and the circuit C_{i-1} such that C_{i-1}^D computes P_{i-1}. With probability at least $1 - 2^{-M}$ this yields an oracle circuit $C_{i,1}$ such that $C_{\mathtt{NW}} = C_{i,1}^D$ computes h_i correctly on $1/2 + M^{-3}$ of the inputs.

To bound the complexity of this step, recall that the reconstruction algorithm can be implemented by a logspace-uniform probabilistic oracle circuit of size $\mathrm{poly}(M) \cdot 2^{\gamma \cdot \ell} \leq \mathrm{poly}(M) \cdot T^{2\gamma}$ and depth $\mathrm{polylog}(M)$, and the circuit C_{i-1} that it outputs has depth $\mathrm{polylog}(M \cdot \ell) = \mathrm{polylog}(M)$. The size of $C_{i,1}$ is trivially bounded by $\mathrm{poly}(M) \cdot T^{2\gamma}$. Now, using the downward self-reducibility of the bootstrapping system, each query to h_i can be answered by a logspace-uniform circuit of size t and depth $\mathrm{polylog}(T)$ that makes queries to C_{i-1}^D. Also recall that the queries are non-adaptive. Hence, the circuit size of the reconstruction algorithm is $\left(\mathrm{poly}(M) \cdot T^{2\gamma}\right) \cdot t \cdot \mathrm{poly}(|C_{i-1}|)$ and its depth is $\mathrm{polylog}(M)$. □

Claim 12.4.14 (the [GL89] list-decoding, coupled with weeding the list)**.** *Given the circuit* C_{i-1} *and oracle access to* $C_{\mathtt{NW}}$ *and to* D*, we can compute with probability at least* $1 - 2^{-M}$ *an oracle circuit* $C_{i,2}$ *such that* $C_{\mathtt{GL}} = C_{i,2}^{C_{\mathtt{NW}}}$ *computes* p_i *on at least* $\mu_{GL} \stackrel{\text{def}}{=\!=} \mathrm{poly}(1/M)$ *of the inputs. This step can be implemented by a logspace-uniform circuit of size* $t \cdot \mathrm{poly}(M, |C_{i-1}|)$ *and depth* $\mathrm{poly}(\log(M), \mathsf{Depth}(C_{i-1}))$*, and the circuit* $C_{i,2}$ *is of size* $\mathrm{poly}(M)$ *depth* $\mathrm{polylog}(M)$.

Proof. We invoke the decoding algorithm from Theorem 12.4.11 for $h_i = \mathtt{Had}(p_i)$ with parameter $\varepsilon = M^{-3}$ and input $1^{\log(T)}$. This algorithm produces a probabilistic oracle circuit $C'_{i,2}$ such that the probability over input $y \in \{0,1\}^{\log(T)}$ and the internal randomness of $C'_{i,2}$ that $(C'_{i,2})^{C_{\mathtt{NW}}}(y) = p_i(y)$ is $\varepsilon_{GL} = \mathrm{poly}(1/M)$. This step can be implemented by a logspace-uniform circuit of size $\mathrm{poly}(\log(T), M) = \mathrm{poly}(M)$ and depth $\mathrm{polylog}(M)$, and the circuit $C'_{i,2}$ is of size $\mathrm{poly}(M)$ and depth $\mathrm{polylog}(M)$.

Then, we perform the following experiment for $O(M/\varepsilon_{GL}) = \mathrm{poly}(M)$ trials in parallel:

[25] In the bootstrapping systems that we construct in Proposition 12.4.3 the queries of the downward self-reducibility algorithm (as well as all worst-case to rare-case reduction) are actually *non-adaptive*. Thus, the term $\mathrm{polylog}(T) \cdot \mathsf{Depth}(C_{i-1})$ can actually be replaced by $\mathrm{polylog}(T) + \mathsf{Depth}(C_{i-1})$. We do not apply this optimization since it does not significantly improve the final parameters of the current construction.

1. Randomly choose a fixed random string for $C'_{i,2}$ and hard-wire it into the circuit, to obtain a deterministic circuit $C_{i,2}$.
2. Estimate the agreement of $C_{i,2}^{C_{\mathtt{NW}}}$ with p_i, up to error $\varepsilon_{GL}/10$ and with confidence $1 - 2^{-M^2}$. To do so we sample $\mathrm{poly}(M)$ inputs, and evaluate p_i and $C_{i,2}^{C_{\mathtt{NW}}}$ on each input. Computing p_i is done using the downward self-reducibility algorithm, the circuit C_{i-1}, and our oracle D.
3. Consider $C_{i,2}$ to be good if $C_{i,2}^{C_{\mathtt{NW}}}$ agrees with p_i on at least μ_{GL} of the inputs.

With probability at least $1 - 2^{-M}$, all the estimates are correct and at least one choice of random string yields a good deterministic circuit $C_{i,2}$. We proceed with the first good $C_{i,2}$ that we find among the trials (according to some predetermined efficient ordering of the trials), and denote $C_{\mathtt{GL}} = C_{i,2}^{C_{\mathtt{NW}}}$.

The latter procedure can be implemented by a logspace-uniform circuit of size $t \cdot \mathrm{poly}(M, |C_{i-1}|)$ and depth $\mathrm{poly}(\log(M), \mathsf{Depth}(C_{i-1}))$. This is since for each of the $\mathrm{poly}(M)$ inputs (which are sampled in parallel) we apply a logspace-uniform circuit of size t with oracle access to C_{i-1} and to D. The circuit $C_{i,2}$ that it produces is of the same size and depth as $C'_{i,2}$ (*i.e.*, size $\mathrm{poly}(M)$ and depth $\mathrm{polylog}(M)$), since we just hard-wired randomness into $C'_{i,2}$. □

Claim 12.4.15 (worst-case to rare-case reducibility)**.** *Given the circuit C_{i-1} and oracle access to $C_{\mathtt{GL}}$ and to D, we can compute with probability at least $1 - 2^{-M} - 1/T^2$ a circuit $C_{i,3}$ such that $C_{i,3}^{C_{\mathtt{GL}}}$ computes p_i. This step can be implemented by a logspace-uniform probabilistic circuit of size $t^2 \cdot \mathrm{poly}(M, |C_{i-1}|)$ and depth $\mathrm{poly}(\log(T), \mathsf{Depth}(C_{i-1}))$ and $C_{i,3}$ is of size $t \cdot \mathrm{poly}(M)$ and depth $\mathrm{polylog}(T)$.*

Proof. Recall that p_i is sample-aided worst-case to ρ-rare-case reducible for $\rho = t^{-\alpha}$. Denoting $\mu_{GL} = M^{-k}$ for a universal constant k, note that

$$\rho = t^{-\alpha} \leq M^{-k} = \mu_{GL} \,,$$

where we relied on the hypothesis that $M \leq t^{\alpha/c''}$ for a sufficiently large constant $c'' \geq k/\alpha$. We invoke the sample-aided reduction with input $1^{\log(T)}$ and a sample of $\mathrm{poly}(M)$ labeled examples of p_i that we produce using the downward self-reducibility algorithm for p_i, the circuit C_{i-1}, and our access to D. With probability $1 - 2^{-t^\alpha} > 1 - 2^{-M}$ (where we relied again on the hypothesis $M \leq t^{\alpha/c''} \leq t^\alpha$) this step produces a probabilistic oracle circuit $C'_{i,3}$ of size t such that for every $y \in \{0,1\}^{\log(T)}$ we have that $\Pr\left[(C'_{i,3})^{C_{\mathtt{GL}}}(y) = p_i(y)\right] \geq 2/3$.

This step can be implemented by a logspace-uniform circuit of size $t^2 \cdot \mathrm{poly}(M, |C_{i-1}|)$ and

depth $\mathrm{poly}(\log(T), \mathsf{Depth}(C_{i-1}))$, and the circuit that it produces is of size $t \cdot \mathrm{poly}(M)$ and depth $\mathrm{polylog}(T)$. To see this, recall that by our assumption, the sample-aided worst-case to rare-case reduction can be implemented by a logspace-uniform circuit of size t and depth $\mathrm{polylog}(T)$; and note that (as in the previous step) we can produce each of the t samples that it requires by a logspace-uniform circuit of size $t \cdot \mathrm{poly}(|C_{i-1}|)$ and depth $\mathrm{poly}(\log(M), \mathsf{Depth}(C_{i-1}))$.

Implementing naive error-reduction in $C'_{i,3}$, we decrease its error from $1/3$ to $1/\mathrm{poly}(T)$ at the cost of increasing its size by a multiplicative factor of $O(\log(T))$ and its depth by an additive factor of $O(\log(T))$. We then randomly choose a fixed random string for $C'_{i,3}$ and hard-wire it; with probability at least $1 - 1/T^2$ we obtain a deterministic oracle circuit $C_{i,3}$ such that $C_{i,3}^{C_{\mathtt{GL}}}$ computes p_i. This can be implemented by a logspace-uniform circuit of size $\tilde{O}(t) \cdot \mathrm{poly}(M)$ and depth $\mathrm{polylog}(T)$. □

Let us now see how the combination of the foregoing three steps yields the algorithm for the i^{th} iteration. The three steps of the i^{th} iteration above succeed with probability at least $1 - 1/T^2 - 3 \cdot 2^{-M}$. Assuming all the steps above succeeded, we have the following:

$C_{\mathtt{NW}} = C_{i,1}^{D}$	computes h_i correctly on $1/2 + M^{-3}$ of the inputs
$C_{\mathtt{GL}} = C_{i,2}^{C_{\mathtt{NW}}}$	computes p_i correctly on $\mu_{GL} = \mathrm{poly}(1/M)$ of the inputs
$C_{i,3}^{C_{\mathtt{GL}}}$	computes p_i

This yields an oracle circuit C_i that computes p_i when given oracle access to D (by replacing the oracle gates in $C_{i,3}$ with $C_{i,2}$, replacing the oracle gates in $C_{i,2}$ with $C_{i,1}$, and keeping the oracle gates in $C_{i,1}$ intact). Note that the size of C_i is

$$|C_i| = \tilde{O}(t) \cdot \mathrm{poly}(M) \cdot T^{2\gamma} ,$$

and its depth is $\mathrm{polylog}(T)$. This is since $C_{i,3}$ is of size $\tilde{O}(t) \cdot \mathrm{poly}(M)$, and $C_{i,2}$ is of size $\mathrm{poly}(M)$, and $C_{i,1}$ is of size $\mathrm{poly}(M) \cdot T^{2\gamma}$, and all three circuits are of depth $\mathrm{polylog}(T)$.

Also, by accounting for the complexity of each of the three steps above, the entire i^{th} iteration can be implemented by a logspace-uniform circuit of size

$$t^2 \cdot \mathrm{poly}(M, |C_{i-1}|) + \mathrm{poly}(M, |C_{i-1}|) \cdot T^{2\gamma} \cdot t \leq t^2 \cdot T^{2\gamma} \cdot \mathrm{poly}(M, |C_{i-1}|)$$

and depth $\mathrm{poly}(\log(T), \mathsf{Depth}(C_{i-1}))$.

We stress that the constant powers hiding inside the poly notation in the size and depth bounds above (both for the circuit implementing the i^{th} iteration and for the circuit that it produces) are universal, arising from the universal constants in Theorem 12.4.10, Theorem 12.4.11, Proposition 12.3.10, and the cost of simulating a circuit of bounded depth (given its description) by a logspace-uniform circuit of bounded depth. Thus we will bound the polynomials in all these expressions by the polynomial power c_0 for a universal constant $c_0 > 1$. ■

After d applications of Lemma 12.4.12, with probability at least $1 - d \cdot \left(\frac{1}{T^2} + 3 \cdot 2^{-M}\right)$ this algorithm yields a circuit C_d such that C_d^D computes p_d. By evaluating C_d^D on the inputs corresponding to first n bits of $P_d(x)$, we obtain the value of $f(x)$.

To bound the overall complexity of the algorithm, we separate the first iteration $i = 1$ (in which $|C_0| \leq t_0$) from iterations $i = 2, ..., d$. The first iteration can be implemented by a logspace-uniform circuit of depth $\mathrm{polylog}(T)$ and of size

$$t^2 \cdot T^{2\gamma} \cdot M^{c_0} \cdot t_0^{c_0} \ ,$$

and the other $d - 1$ iterations can be implemented (together) by a logspace-uniform circuit of depth $(d-1) \cdot \mathrm{polylog}(T)$ and size

$$(d-1) \cdot t^2 \cdot T^{2\gamma} \cdot M^{c_0} \cdot (\tilde{O}(t) \cdot M^{c_0} \cdot T^{2\gamma})^{c_0} < (d-1) \cdot (t \cdot T^{\gamma} \cdot M)^{4c_0^2} \ .$$

Also accounting for the last step (of evaluating C_d on n inputs), the procedure in its entirety can be implemented by a logspace-uniform circuit of depth $d \cdot \mathrm{polylog}(T)$ and size

$$\begin{aligned} & t^2 \cdot T^{2\gamma} \cdot M^{c_0} \cdot t_0^{c_0} + (d-1) \cdot (t \cdot T^{\gamma} \cdot M)^{4c_0^2} + n \cdot (\tilde{O}(t) \cdot M^{c_0} \cdot T^{2\gamma})^{c_0} \\ < & (t \cdot T^{\gamma} \cdot M)^{4c_0^2} \cdot (d + n + t_0^{c_0}) \ , \end{aligned}$$

and this concludes the proof if we set the universal constant c' to be at least $4c_0^2$.

12.5 Non-black-box Derandomization from "almost-all-inputs" Hardness

In this section we use the reconstructive targeted HSG from Section 12.4 to prove various hardness-to-randomness results. In Section 12.5.1 we state hardness-to-randomness results for general probabilistic algorithms, and in particular prove Theorem 12.1.3. In Section 12.5.2 we state hardness-to-randomness results for restricted (probabilistic) circuit classes, and in particular prove Theorem 12.1.6. And in Section 12.5.3 we prove Theorem 12.1.5 for the "low-end" setting.

For some of our results (in Section 12.5.1 and Section 12.5.3) we will need the following modified version of our reconstructive HSG from Proposition 12.4.5, which was mentioned prior to the proposition's statement. Compared to the latter HSG, the reconstruction algorithm in the following result can be more efficient, at the cost of a less efficient HSG; specifically, the overhead of T^{δ} in the reconstruction time can now be for a *subconstant* $\delta = o(1)$, at the cost of having an HSG with running time $T^{O(1/\delta)}$, and both the HSG and the reconstruction are not necessarily computable by logspace-uniform circuits of bounded depth.

Proposition 12.5.1 (a reconstructive targeted HSG, a version with low reconstruction overhead)**.** *There exists a universal constant $c > 1$ such that the following holds. Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be computable by logspace-uniform circuits of size $T(n)$ and depth $d(n)$, let $\delta\colon \mathbb{N} \to (0,1)$, and let $M\colon \mathbb{N} \to \mathbb{N}$ such that*

$$c \cdot \log(T(n)) \leq M(n) \leq T(n)^{\delta(n)/c} ,$$
$$\delta(n) \geq c \cdot \frac{\log\log(T)}{\log(T)} .$$

Then, there exist a deterministic algorithm H_f and a probabilistic algorithm R that for every $x \in \{0,1\}^n$ satisfy the following:

1. **Generator.** *The generator H_f gets input x, runs in time $T^{O(1/\delta)}$, and outputs a set of M-bit strings.*
2. **Reconstruction.** *When R gets input x and oracle access to a function $D\colon \{0,1\}^M \to \{0,1\}$ such that $\Pr_{r\in\{0,1\}^M}[D(r) = 1] \geq 1/M$ but D rejects all the strings that $H_f(x)$ prints, it runs in time $(d + n^c) \cdot T^{\delta} \cdot M^c$ and outputs $f(x)$ with probability at least $1 - 1/M$.*

Proof. The proof is identical to the proof of Proposition 12.4.5, with only the following differences. First, we verify that the hypothesized lower bound $T^{\mu} \geq \log^{1/\alpha}(T)$ holds in Proposition 12.4.3 (the lower bound holds by our choice of $\mu = \delta/5c'$ and by our assumption about δ). Secondly, we do

not claim that H_f and R are logspace-uniform circuits of bounded depth, so we just bound their running time (using the exact same bound as the size bound on the circuits). Thirdly, in our conclusion the running time of the generator H_f is $T^{O(1/\delta)}$, which is not necessarily polynomial in T. ■

Throughout this section, when we say that a probabilistic algorithm A computes a multi-output function f on input x, we mean that $\Pr[A(x) = f(x)] \geq 2/3$, where the probability is over the internal coin tosses of A.

12.5.1 Hardness-to-randomness Tradeoffs and Proof of Theorem 12.1.3 and Theorem 12.1.4

We now prove several hardness-to-randomness tradeoffs as a consequence of Proposition 12.4.5 and Proposition 12.5.1, and in particular we deduce Theorem 12.1.3. We first prove Claim 12.1.2, which asserts that "almost-all-inputs" lower bounds are necessary for derandomization. The proof is slightly more subtle than one might expect, since the hard function has multiple output bits but the derandomization hypothesis refers to machines with a single output bit.

Claim 12.5.2 (Claim 12.1.2, restated)**.** *If* $\mathsf{prBPP} = \mathsf{prP}$*, then for every* $c \in \mathbb{N}$ *there exists* $f\colon \{0,1\}^n \to \{0,1\}^n$ *computable in deterministic polynomial time such that* f *cannot be computed in probabilistic time* n^c *on* almost all inputs.

Proof. By our hypothesis, $\mathsf{prBPTIME}[O(n^c)] \subseteq \mathsf{prDTIME}[n^{c'}]$, for a sufficiently large constant c'. The function f gets input $x \in \{0,1\}^n$, and for each $i \in [n]$ it simulates the i^{th} Turing machine M_i for $n^{c'+2}$ steps on input x, and sets $f(x)_i$ to be the opposite of the i^{th} output bit of M_i; that is, $f(x)_i = 1 - M_i(x)_i$ (or $f(x)_i = 0$, in case M_i does not halt after $n^{c'+2}$ steps).

Assume towards contradiction that f can be computed on an infinite set $X \subseteq \{0,1\}^*$ of inputs by a probabilistic machine M_0 that runs in time n^c. Consider the probabilistic machine M_1 that takes as input a pair (x,i) of strings of identical length, and outputs $M_1(x,i) = M_0(x)_i$ (if $i > |x|$ or $M_0(x)$ does not output $|x|$ bits then $M_1(x,i) = 0$). Note that M_1 induces a promise-problem $\Pi \in \mathsf{prBPTIME}[O(n^c)]$, where the promise is that $x \in X$. By our derandomization hypothesis, $\Pi \in \mathsf{prDTIME}[n^{c'}]$. Let M_2 be a deterministic machine that solves Π in time $n^{c'}$, and note that for every $x \in X$ and every $i \in [|x|]$ we have that $M_2(x)_i = f(x)_i$.

Consider the following procedure: Given input $x \in \{0,1\}^n$, we simulate M_2 on inputs $(x,1),...,(x,n)$ and print the concatenation of its outputs. For every $x \in X$, this procedure prints $f(x)$ in time $O(n^{c'+1})$. Now, let i be the index of a Turing machine M_i that implements the foregoing procedure

in time $O(n^{c'+1})$. By the definition of f, for every input $x \in \{0,1\}^*$ of sufficiently large length $n \geq i$ (in particular, for infinitely many inputs $x \in X$) we have that $f(x)_i = 1 - M_i(x)_i$. This is a contradiction. ■

We now state a very general tradeoff. Loosely speaking, and choosing nice parameters, the following result asserts that we can derandomize $\mathsf{prRTIME}[M]$ using a function f that is hard for probabilistic time $\mathrm{poly}(M)$ on almost all inputs, with the following parameters: If f is computable by logspace-uniform circuits of size T and depth $\mathrm{poly}(M)$, then the derandomization time is polynomial in $2^{\log^2(T)/\log(M)}$. The foregoing parametrization is a special case, since the result is further parametrized by the depth d of circuits for f.

Theorem 12.5.3 (non-black-box derandomization, general version)**.** *There exists a universal constant $c > 1$ such that the following holds. Let $T, M, d \colon \mathbb{N} \to \mathbb{N}$ such that $n \leq \max\{d(n), M(n)\} \leq T(n) \leq 2^{M(n)/c}$. Let $\Pi \in \mathsf{prRTIME}[M]$, and let $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions such that $\mathbf{x}_n$ is over $\{0,1\}^n$ and does not violate the promise of Π. Assume that for $\mu \colon \mathbb{N} \to [0,1)$ there exists a function $f \colon \{0,1\}^n \to \{0,1\}^n$ computable by deterministic logspace-uniform circuits of size $T(n)$ and depth $d(n)$ such that for every probabilistic algorithm A running in time $d(n) \cdot M(n)^c$, the probability over $x \sim \mathbf{x}_n$ that $A(x)$ computes $f(x)$ is at most $\mu(n)$. Then, $\Pi \in \mathsf{heur}_{\mathbf{x},1-\mu}\text{-}\mathsf{prDTIME}\left[2^{c\cdot(\log^2(T)/\log(M))}\right]$.*

In particular, if there exists $f \colon \{0,1\}^n \to \{0,1\}^n$ as above such that every probabilistic algorithm running in time $d(n) \cdot M(n)^c$ fails to compute f on almost all inputs*, then there exists a $(1/M)$-targeted HSG for time M with seed length $O(\log(T)^2/\log(M))$ and running time $2^{O(\log^2(T)/\log(M))}$, and consequently*

$$\mathsf{prRTIME}[M(n)] \subseteq \mathsf{prDTIME}\left[2^{c\cdot(\log^2(T)/\log(M))}\right] .$$

Proof. Let $\Pi \in \mathsf{prRTIME}[M(n)]$, let M_Π be a probabilistic linear-time machine that solves Π, and let $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions that do not violate the promise of Π. We instantiate Proposition 12.5.1 with the function f and with parameters T, d, M and with δ such that $M = T^{\delta/c_0}$, where c_0 is the universal constant from Proposition 12.5.1 (*i.e.*, $\delta = c_0 \cdot \log(T)/\log(M)$). Note that the hypothesis of Proposition 12.5.1 is satisfied by our constraints on M and T and by our choice of δ.

Given input $x \in \{0,1\}^n$, we compute the set $H(x) = \left\{r_i \in \{0,1\}^{O(n)}\right\}_{i\in\{0,1\}^{O(\log(T)/\delta)}}$, and accept if and only if there exists $i \in \{0,1\}^{O(\log(T)/\delta)}$ such that M_Π accepts x with randomness r_i. The running time of this deterministic algorithm is $T^{O(1/\delta)} \cdot M(n) = 2^{O(\log^2(T)/\log(M))}$.

Note that foregoing algorithm never accepts "no" instance of Π, and thus may only err by rejecting "yes" instances. Assume towards a contradiction that for some $n \in \mathbb{N}$, with probability more than $\mu(n)$ over $x \sim \mathbf{x}_n$ it holds that x is a "yes" instance of Π that the foregoing algorithm rejects. Then, we can compute f on n-bit inputs with success probability more than $\mu(n)$ over $x \sim \mathbf{x}_n$, as follows. Given x, we use the algorithm R from Proposition 12.5.1 with the function $D_x(r) = M_\Pi(x, r)$ as the distinguisher. By our assumption, for every $x \in S$ the function D_x accepts at least $1/2$ of its inputs, but rejects all the strings that $H(x)$ outputs. Hence, for every "yes" instance that the deterministic algorithm above rejects, our algorithm outputs $f(x)$ with probability $1 - 1/O(n) \gg 2/3$, in time at most

$$O(n + d(n)) \cdot T(n)^\delta \cdot M(n)^{c_0} = O(n + d(n)) \cdot M(n)^{2c_0} < d(n) \cdot M(n)^{3c_0} ,$$

which contradicts the hardness of f if we choose $c \geq 3c_0$.

The "in particular" part of the statement follows since the assumption that f is hard on almost all inputs implies that f is hard for all possible distributions $\mathbf{x}_n$ and with success bound $\mu(n) = 0$. In this case, for every time-M algorithm A and every fixed x of sufficiently large length such that $\Pr_r[A(x, r) = 1] \geq 1/2$, the proof above implies that there exists $r_i \in H(x)$ such that $A(x, r_i) = 1$. Indeed, it follows that the algorithm that gets input x and outputs $H(x)$ is a $(1/M)$-targeted HSG for time M. ■

We now prove Theorem 12.1.4, which assert a hardness-to-randomness tradeoff for derandomization of polynomial-time algorithms. The theorem follows by instantiating Theorem 12.5.3 with $M = \text{poly}(n)$, and deducing derandomization of algorithms with two-sided error via the standard reduction by [Sip83, Lau83] of derandomization of prBPP to *any* derandomization (*i.e.*, not necessarily black-box) of prRP (see, *e.g.*, [BF99] and [GVW11] for an explanation).

Corollary 12.5.4 (non-black-box derandomization of prBPP; Theorem 12.1.4, restated)**.** *There exists a universal constant $c > 1$ such that the following holds. Assume that there exists a function $f : \{0,1\}^n \to \{0,1\}^n$ computable by deterministic logspace-uniform circuits of size $T(n)$ and depth $d(n)$ that cannot be*

computed in probabilistic time $d(n) \cdot n^c$ *on* almost all inputs. *Then, we have that*

$$\mathsf{prRP} \subseteq \bigcup_{a \geq 1} \mathsf{prDTIME}[\bar{T}(n^a)]$$
$$\mathsf{prBPP} \subseteq \bigcup_{a \geq 1} \mathsf{prDTIME}[\bar{T}(\bar{T}(n^{c \cdot a})^a)] \,,$$

where $\bar{T}(m) = 2^{c \cdot \log^2(T(m))/\log(m)}$.

Proof. Let $c > 1$ be the universal constant from Theorem 12.5.3. We can assume wlog that $T(n) \leq 2^{n/c}$, otherwise the conclusion is trivial. This allows us to instantiate Theorem 12.5.3 with $M(n) = O(n)$, and deduce that $\mathsf{prRTIME}[O(n)] \subseteq \mathsf{prDTIME}[\bar{T}(n)]$, where $\bar{T}(n) = 2^{c \cdot (\log^2(T)/\log(n))}$. The derandomization of prRP follows by a padding argument (reducing any problem in $\mathsf{prRTIME}[n^a]$ to $\mathsf{prRTIME}[O(n)]$ by padding the input to length n^a).

For the derandomization of prBPP, we rely on the fact that for some fixed $k \in \mathbb{N}$, every problem in $\mathsf{prBPTIME}[O(n)]$ is probabilistically reducible in probabilistic time $O(n^k)$ and with one-sided error to a corresponding problem in $\mathsf{prRTIME}[O(n)]$ (see [Sip83, Lau83, BF99, GVW11]). Thus, for some constant $b \geq 1$ we have that $\mathsf{prBPTIME}[O(n)] \subseteq \mathsf{prDTIME}[\bar{T}(\bar{T}(n^b)^b)]$, which implies the derandomization of prBPP (by a padding argument as above). ■

Remark 12.5.5. We suspect that the derandomization time of $2^{\log^2(T)/\log(n)}$ in Corollary 12.5.4 can be improved to be only $\mathrm{poly}(T)$. The current overhead comes from applying the Nisan-Wigderson PRG to each row in the bootstrapping matrix, and in particular from the combinatorial designs underlying this PRG. More sophisticated constructions of HSGs and PRGs that avoid this particular overhead are well-known (*e.g.*, by Shaltiel and Umans [SU05] and by Umans [Uma03]), and we suspect that using such constructions instead of the NW PRG might improve the overhead.

Remark 12.5.6. The hypothesis in Corollary 12.5.4 is that $f\colon \{0,1\}^n \to \{0,1\}^n$ is hard for *all* probabilistic algorithm that run in the prescribed time. However, to deduce the derandomization conclusion it suffices to assume that f is hard for *one particular algorithm*, which is the algorithm that is obtained when invoking the reconstruction procedure for f from Proposition 12.5.1 with the distinguisher being the standard machine solving the CAPP problem.[26] (This is the case because if the targeted HSG "fools" this machine, then we can derandomize all of prRP, relying on the completeness of the one-sided error version of CAPP for prRP.) Thus, to deduce the derandomization

[26] That is, the distinguisher $D = D_x$ interprets the input x as a description of a circuit C_x, gets a random (or pseudo-random) string r, and outputs $C_x(r)$.

conclusion it suffices to analyze the hardness of f with respect to a single algorithm.

Our main result in this section (*i.e.*, Theorem 12.1.3) is the special case of Corollary 12.5.4 with $T(n) = \text{poly}(n)$. As mentioned in Section 12.1, we state a more general version of the result, which refers to a hard function of any fixed polynomial depth $d(n) = \text{poly}(n)$ (rather than n^2).

Corollary 12.5.7 (polynomial-time non-black-box derandomization of prBPP; Theorem 12.1.3, restated)**.** *There exists $c > 1$ such that for every $k \in \mathbb{N}$ the following holds. Assume that there exists a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable by deterministic logspace-uniform circuits of polynomial size and depth n^k that cannot be computed in probabilistic time n^{k+c} on* almost all inputs*. Then* $\mathsf{prBPP} = \mathsf{prP}$.

Note that Corollary 12.5.7 follows immediately from Corollary 12.5.4 using the parameters $T(n) = \text{poly}(n)$ and $d(n) = n^k$ (in which case $\bar{T}(n) = \text{poly}(n)$).

A different type of hardness-to-randomness tradeoff. In the hardness-to-randomness results above, to derandomize $\mathsf{prBPTIME}[M]$ we assumed a function that is hard for probabilistic time $\text{poly}(M)$ and that can be computed by uniform circuits of relatively small depth (*i.e.*, circuits of depth d and size $T \gg d$). We now show a different type of tradeoff, where we considerably relax the assumption that the circuits have bounded depth, but we assume hardness for larger probabilistic time (closer to the circuit size T than to the time bound M that we want to derandomize).

This type of hardness-to-randomness tradeoff is particularly appealing in the setting of derandomization in superpolynomial time, since in this setting the assumption about the depth of the circuits can be almost completely eliminated. For simplicity, let us state just one nice instantiation of it, which refers to derandomization in *quasipolynomial* time. (Instantiations with other time bounds can be obtained using the same idea.)

Corollary 12.5.8 (superpolynomial-time non-black-box derandomization of prBPP)**.** *There exists a constant $c > 1$ such that for every constant $\varepsilon > 0$ the following holds. Assume that there exists a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable by deterministic logspace-uniform circuits of quasipolynomial size $T(n) = 2^{\log^c(n)}$ (where $c > 1$) and depth $T(n)^{1-\varepsilon}$ that cannot be computed in probabilistic time $T(n)^{1-\varepsilon/4}$ on* almost all inputs*. Then,*

$$\mathsf{prBPP} \subseteq \mathsf{prQP}\,,$$

where $\mathsf{prQP} = \cup_c \mathsf{prDTIME}[2^{\log^c(n)}]$.

Proof. The proof is very similar in structure to that of Theorem 12.5.3 and of Corollary 12.5.4, so we focus on pointing out the differences in parameters. Given $\Pi \in \mathsf{prRTIME}[O(n)]$, we instantiate Proposition 12.5.1 with parameters T, d, $M = O(n)$, and $\delta = \varepsilon/2$. The derandomization of Π runs in time $\mathrm{poly}(T)$, which is quasipolynomial in n, and it succeeds since the hardness is for time larger than $(d + n^{c_0}) \cdot T^{\varepsilon/2} \cdot n^c < T^{1-\varepsilon+\varepsilon/2} \cdot \mathrm{poly}(n) < T^{1-\varepsilon/4}$. It follows (by padding) that $\mathsf{pr}\mathcal{RP} \subseteq \mathsf{prQP}$, and using [Sip83, Lau83] we deduce that $\mathsf{prBPP} \subseteq \mathsf{prQP}$. ■

12.5.2 Tight Hardness-to-randomness Results for Low-depth Circuits

In this section we extend our hardness-to-randomness results to the setting of derandomizing probabilistic low-depth circuits, and in particular logspace-uniform NC circuits. These results follow using the reconstructive HSG in Proposition 12.4.5, relying on the fact that both the HSG and the reconstruction can be computed by logspace-uniform circuits of low depth.

Let us first define logspace-uniform NC circuits and logspace-uniform probabilistic NC circuits. (Recall that the general notion of probabilistic circuits that we use was defined in Definition 12.3.5.)

Definition 12.5.9 (logspace-uniform NC)**.** *For two constants $i, c \in \mathbb{N}$, we denote by* $\mathsf{logspace\text{-}uniform\text{-}prNC}^i[n^c]$ *the class of promise problems solvable by families of logspace-uniform circuits of depth* $\log^i(n)$ *and size* n^c*.*

Definition 12.5.10 (logspace-uniform probabilistic NC)**.** *We denote by* $\mathsf{logspace\text{-}uniform\text{-}prBP} \cdot \mathsf{NC}^i[n^c]$ *the class of promise problems solvable by families of logspace-uniform* probabilistic *circuits of such depth and size that err with probability at most* $1/3$*. We denote by* $\mathsf{logspace\text{-}uniform\text{-}prR} \cdot \mathsf{NC}^i[n^c]$ *the class of promise problems solvable by families of logspace-uniform* probabilistic *circuits of such depth and size that err only on "yes" instances (i.e., have one-sided error) and with probability at most* $1/2$

Referring to Definition 12.5.9 and Definition 12.5.10, when we omit the size parameter we implicitly refer to some (unspecified) polynomial size, and when we omit the depth parameter we implicitly refer to some (unspecified) polylogarithmic depth.

Our first result asserts that, for a *fixed* i, logspace-uniform NC^i circuits with *one-sided error* can be derandomized by logspace-uniform deterministic circuits, conditioned on a hardness hypothesis that corresponds to i. (Later on we will extend this to all i and to derandomization with two-sided error.) In more detail:

Proposition 12.5.11 (non-black-box derandomization of logspace-uniform NC)**.** *There exists a universal constant $c > 1$ such that for any constant $i \in \mathbb{N}$ the following holds. Assume that for some $j \in \mathbb{N}$*

there exists a function $f\colon \{0,1\}^n \to \{0,1\}^n$ *computable by deterministic logspace-uniform* NC^j *circuits that cannot be computed by logspace-uniform probabilistic circuits of size* n^c *and of depth* $\log^{i+j+c}(n)$ *on* almost all inputs. *Then,*

$$\text{logspace-uniform-prR} \cdot \mathsf{NC}^i \subseteq \text{logspace-uniform-prNC} .$$

Proof. We first show that $\text{logspace-uniform-prR} \cdot \mathsf{NC}^{i+1}[O(n)] \subseteq \text{logspace-uniform-prNC}$, and then deduce the general statement by a padding argument. Fix $\Pi \in \text{logspace-uniform-prR} \cdot \mathsf{NC}^{i+1}[O(n)]$, decidable by a circuit family $\{C_n\}$ that is printed by a uniform machine M_Π. Let f be the hard function from our hypothesis, and denote the size of the circuits for f by n^k and their depth by $\log^j(n)$. Let c' be the universal constant from Proposition 12.4.5.

Our deterministic algorithm uses the HSG H_f from Proposition 12.4.5 with the function f and with $\delta = c'/k$ and $M = O(n)$, then simulates C_n on input x and with each of the M-bit strings that H_f outputs (used as randomness for C_n), and finally outputs the majority value. By Proposition 12.4.5 and the properties of $\{C_n\}$, this algorithm is computable in logspace-uniform NC.

Assuming towards a contradiction that this derandomization fails on infinitely many inputs x, for each such x we have that $C_{|x|}(x,\cdot)$ rejects all strings in the HSG's output set but accepts a random string with probability at least $1/2$. It follows that the reconstruction algorithm from Proposition 12.4.5 computes $f(x)$, with high probability. This reconstruction algorithm is computable by a logspace-uniform probabilistic circuit of size

$$\left(\text{polylog}(n) + n^{c'}\right) \cdot (n^k)^\delta \cdot O(n)^{c'} = O(n^{3c'})$$

and depth $\log^j(n) \cdot \log^{i+1}(n) \cdot \log^{c'}(n^k) < \log^{i+j+3c'}(n)$. This contradicts the hypothesized hardness of f if we set $c > 3c'$.

Finally, for any constant k let $\Pi \in \text{logspace-uniform-prR} \cdot \mathsf{NC}^i[n^k]$. We define a padded version Π^{pad} whose "yes" instances are $\left\{(x, 1^{|x|^k - |x|}) : x \text{ is a yes instance of } \Pi\right\}$ and whose "no" instance are $\left\{(x, 1^{|x|^k - |x|}) : x \text{ is a no instance of } \Pi\right\}$. Observe that $\Pi^{\mathsf{pad}} \in \text{logspace-uniform-prR} \cdot \mathsf{NC}^i[O(n)] \subseteq \text{logspace-uniform-prR} \cdot \mathsf{NC}^{i+1}[O(n)]$, since we can compute in logspace a circuit that checks if the input is of the form $(x, 1^{|x|^k} - |x|)$ and that simulates the circuit for Π on the first part in the input (*i.e.*, on x). Thus, by our hypothesis $\Pi^{\mathsf{pad}} \in \text{logspace-uniform-prNC}$. Then, given an input x, a logspace machine can print a circuit that pads its input x with $1^{|x|^k - |x|}$ and then simulates

the circuit for Π^{pad}. Thus, $\Pi \in$ logspace-uniform-prNC. ■

Let us now extend Proposition 12.5.11 to hold for all i and for derandomization with two-sided error. In the proof below, to leverage derandomization of logspace-uniform-prR $\cdot$ NC to derandomization of logspace-uniform-prBP $\cdot$ NC we rely on the fact that the classical reduction of [Sip83, Lau83] can be implemented by logspace-uniform NC circuits.

Corollary 12.5.12 (non-black-box derandomization of logspace-uniform NC)**.** *There exists a universal constant $c > 1$ such that the following holds. Assume that for every $i \in \mathbb{N}$ there exists $j \in \mathbb{N}$ and a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable by deterministic logspace-uniform NC^j circuits that cannot be computed by logspace-uniform probabilistic circuits of size n^c and of depth $\log^{i+j+c}(n)$ on* almost all inputs. *Then,*

$$\text{logspace-uniform-prBP} \cdot \mathsf{NC} \subseteq \text{logspace-uniform-prNC}\,.$$

Proof. Invoking Proposition 12.5.11 with all $i \in \mathbb{N}$, we deduce that logspace-uniform-prR $\cdot$ NC $\subseteq$ logspace-uniform-prNC. Now, let $\Pi \in$ logspace-uniform-prBP $\cdot \mathsf{NC}^i[n^k]$ for some constant k, solvable by a logspace-uniform circuit family $\{C_n\}$. We implement the standard reduction of [Sip83, Lau83] in logspace-uniform NC, as follows. Consider the logspace-uniform circuit family $\{C'_n\}$ such that C'_n simulates C_n while implementing naive error-reduction, to reduce the error below 2^{-n}; note that $\{C'_n\}$ is also logspace-uniform, and of size $n^{k'}$ for some $k' > k$ and depth $O(\log^i(n))$.

Now, consider the following promise problem Π'. The valid inputs are of the form $(x, s_1, ..., s_{|x|^{k'}})$ where each s_i is a string of length $|x|^{k'}$. The "yes" instances are such that for every $r \in \{0,1\}^{|x|^{k'}}$ there exists $i \in [|x|^{k'}]$ for which $C'_{|x|}(x, r \oplus s_i) = 1$, and the "no" instances are such that for at least half of the $r \in \{0,1\}^{|x|^{k'}}$ it holds that $C'_{|x|}(x, r \oplus s_i) = 0$ for all $i \in [|x|^{k'}]$. Relying on the properties of $\{C'_n\}$ and on the fact that we can check the s_i's in parallel, the problem Π' is solvable by a logspace-uniform family of probabilistic circuits of size $n^{O(k')}$ and depth $O(\log^i(n))$ with one-sided error. Hence, Π' is also solvable by a logspace-uniform family $\{C''_n\}$ of *deterministic* circuits of size $n^{k''}$ and depth $\log^{i'}(n)$, for some constants $k'', i' \in \mathbb{N}$.

It follows that $\Pi \in$ logspace-uniform-prR $\cdot$ NC. To see this, consider a circuit family that gets input x, chooses $(s_1, ..., s^{k'}_{|x|})$ at random, and simulates the circuit C''_n (where $n = |x| + |x|^{2k'}$) on input $(x, s_1, ..., s_{|x|^{k'}})$. The well-known analysis (see, *e.g.*, [Gol08, Theorem 6.9]) shows that this circuit family solves Π with one-sided error of at most $1/2$. Also, this circuit family is of polynomial size

and of polylogarithmic depth. Hence, $\Pi \in$ logspace-uniform-prNC. ■

We complement Proposition 12.5.11 by showing that, analogously to Claim 12.5.2, functions that are hard on almost all inputs are necessary for derandomization of logspace-uniform NC circuits. This result relies on diagonalization, and in particular on diagonalizing against logspace-uniform circuits by other logspace-uniform circuits. To do so we need to quantify the "amount of logspace" used to construct the circuit, and so we introduce the following definitions.

Definition 12.5.13 (α-logspace-uniformity)**.** *For a constant $\alpha \geq 1$, we say that a circuit family $\{C_n\}$ of size $T(n)$ is* α-logspace-uniform *if there exists a Turing machine M of space complexity $\alpha \cdot \log(T(n))$ such that $M(1^n)$ prints C_n.*

The following diagonalization-based proof is similar to that of Claim 12.5.2, but with the additional complication that the diagonalizing function must be computable by logspace-uniform circuits of bounded depth even when it simulates logspace machines whose output is not necessary a bounded-depth circuit.

Proposition 12.5.14 (almost-all-inputs hardness is necessary for derandomization of NC)**.** *Assume that* logspace-uniform-prBP $\cdot$ NC $\subseteq$ logspace-uniform-prNC. *Then, for every $\alpha \geq 1$ and $c_1, c_2 \in \mathbb{N}$ there exists $f\colon \{0,1\}^n \to \{0,1\}^n$ computable by logspace-uniform circuits of polylogarithmic depth and polynomial size such that f cannot be computed by α-logspace-uniform probabilistic circuits of size n^{c_1} and depth $\log^{c_2}(n)$ on* almost all inputs.

Proof. The hard problem f is defined using two sufficiently large constants d_1 and d_2 that depend on c_1 and c_2 and will be determined in a moment. For every $x \in \{0,1\}^n$ we will print a circuit $\{0,1\}^n \to \{0,1\}^n$ that consists of n separate sub-circuits $C_1, ..., C_n$, one per each output bit $i \in [n]$. For each $i \in [n]$,

1. Let M_i be the i^{th} Turing machine (according to some efficient enumeration). We check that $M_i(1^n)$ uses $\alpha \cdot \log(n^{d_1})$ space and halts after $2^{\alpha \cdot \log(n^{d_1})}$ steps.
2. We print a circuit C_i that has the output of $M_i(1^n)$, denoted C'_i, hard-wired. The circuit C_i tries to simulate $C'_i(x)$ assuming that C'_i is a circuit of size n^{d_1} and depth $\log^{d_2}(n)$.
3. If the simulation of $C'_i(x)$ succeeds, then $C_i(x) = 1 - C'_i(x)$; otherwise, $C_i(x) = 0$.

Note that the algorithm for f can be computed by a logspace-uniform circuit family of polynomial size and of polylogarithmic depth. Assume towards a contradiction that f can be computed on an infinite set $X \subseteq \{0,1\}^*$ of inputs by an α-logspace-uniform probabilistic circuit family $\{F_n\}$

of size $T = T(n) = n^{c_1}$ and depth $\log^{c_2}(n)$, and let M_i be a Turing machine that prints this circuit family using $\alpha \cdot \log(T)$ space.

Let M'_i be a logspace machine that prints a circuit family $\{F'_n\}$ in which each F'_n gets input (x, i), simulates F_n on x, and prints the i^{th} bit of $F_n(x)$. Consider the promise-problem of mapping $(x, i) \mapsto f(x)_i$ where the promise is that $x \in X$. By our derandomization hypothesis, for some $d_1, d_2 \in \mathbb{N}$ there exists a logspace machine M''_i that prints a family $\{F''_n\}$ of *deterministic* circuits of size n^{d_1} and depth $\log^{d_2}(n)$ that agree with F'_n on every $x \in X$ and $i \in [|x|]$. If we use these values d_1 and d_2 in the definition of f above, we obtain a contradiction. ■

By combining Proposition 12.5.11 and Proposition 12.5.14 we obtain the following "near-equivalence" result, in which there is a very small gap between a hypothesis that suffices to derandomize logspace-uniform NC and a hypothesis that is necessary for doing so. The following statement implies Theorem 12.1.6.

Corollary 12.5.15 (non-black-box derandomization of NC)**.** *There exists a universal constant $c > 1$ such that, considering the following statements, we have that $(1) \Rightarrow (2) \Rightarrow (3)$.*

1. *(**Sufficient lower bound.**) For every sufficiently large $i \in \mathbb{N}$ there exists $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in logspace-uniform deterministic $\mathsf{NC}^{(.99 \cdot i)}$ that cannot be computed by logspace-uniform probabilistic $\mathsf{NC}^i[n^c]$ on* almost all inputs.
2. *(**Derandomization.**)* $\mathsf{logspace\text{-}uniform\text{-}prBP} \cdot \mathsf{NC} \subseteq \mathsf{logspace\text{-}uniform\text{-}prNC}$.
3. *(**Necessary lower bound.**) For every $i \in \mathbb{N}$ and $\alpha \geq 1$ there exists $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in logspace-uniform deterministic NC that cannot be computed in α-logspace-uniform probabilistic $\mathsf{NC}^i[n^c]$ on* almost all inputs.

Proof. The implication $(2) \Rightarrow (3)$ follows immediately from Proposition 12.5.14. To see that $(1) \Rightarrow (2)$, by Proposition 12.5.11 it suffices to show, for every fixed $i_0 \in \mathbb{N}$, a function f in logspace-uniform NC^j for some $j \in \mathbb{N}$ that cannot be computed by logspace-uniform probabilistic circuits of size n^c and depth $\log^{i_0+j+c}(n)$ on almost all inputs. Such a function exists by our hypothesis, taking j to be sufficiently large such that $j + (i_0 + c) < .99 \cdot j$. ■

12.5.3 "Low-end" derandomization from hard functions without structural constraints

We now prove Theorem 12.1.5, which deduces a fixed-exponential-time derandomization of RP from the existence of a hard function f without any structural constraints on f (*i.e.*, we do not need to assume that f has logspace-uniform low-depth circuits).

Towards presenting the result, we say that a probabilistic algorithm A computes σ on input $x \in \{0,1\}^n$ with zero error and success probability β if $A(x)$ outputs either σ or $\perp$ and $A(x)$ outputs σ with probability at least β. We will also need the following definition, which relaxes "almost-all-inputs" hardness to only require failure on all inputs of certain lengths.

Definition 12.5.16 (infinitely-often almost-all-inputs hardness)**.** *We say that a function f is hard for a class $\mathcal{C}$ of probabilistic algorithms* infinitely often on almost all inputs *if for every $C \in \mathcal{C}$ there exists an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ and $x \in \{0,1\}^n$ it holds that $\Pr[C(x) = f(x)] < 2/3$.*

Analogously to the shorthand notation i.o.- for "infinitely-often", we will use the shorthand notation i.o.--aai to refer to "infinitely-often almost-all-inputs" hardness. The following result, which implies Theorem 12.1.5, asserts that if there exists a function in EXP that is hard for BPP infinitely-often almost-all-inputs, then RP can be derandomized in fixed exponential time; and if RP can be derandomized in fixed exponential time, then there exists a function in EXP that is hard for RP infinitely-often almost-all-inputs. Indeed, the only gap between the hardness hypothesis and the hardness that we conclude from derandomization is that the former is for BPP whereas the latter is for RP.

Theorem 12.5.17 (a near-equivalence in the "low-end" setting)**.** *Considering the following three statements, we have that (1) $\Rightarrow$ (2) $\Rightarrow$ (3).*

1. *(*EXP ***is i.o.--aai hard for*** BPP*.) There exists a universal constant $c_1 \geq 1$ and a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in time $2^{n^{c_1}}$ such that f is hard for all polynomial-time probabilistic algorithms infinitely often on almost all inputs.*
2. *(**Infinitely-often non-trivial derandomization for** prRP**.) There exists a universal constant $c_2 \geq 1$ such that* prRP $\subseteq$ *i.o.-*DTIME$[2^{n^{c_2}}]$.
3. *(*EXP ***is i.o.--aai hard for*** $\mathcal{ZPP}$**.)** There exists a universal constant $c_3 \geq 1$ and a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in time $2^{n^{c_3}}$ such that f is hard for all polynomial-time probabilistic* zero-error *algorithms infinitely often on almost all inputs.*

Proof of (1) $\Rightarrow$ (2). First note that we can assume that EXP $\subset$ P/poly. This since otherwise, by [NW94, BFNW93] we have that prBPP $\subseteq$ i.o.-prSUBEXP [NW94, BFNW93], which in particular implies that prRP $\subseteq$ i.o.-prDTIME$[2^n]$.

Now, recall that EXP $\subset$ P/poly implies that EXP $\subseteq$ MA [BFL91, BFNW93]. Assuming that Item (1) holds for a constant $c_1 > 1$, we have that DTIME$[2^{n^{c_1}}] \subseteq$ MA. This further implies that

$\mathsf{DTIME}[2^{n^{c_1}}]$ is contained in $\mathsf{MATIME}[n^d]$ for a constant $d > 1$ (since $\mathsf{DTIME}[2^{n^{c_1}}]$ has a complete problem under linear-time reductions).

Note that any function in $\mathsf{MATIME}[n^d]$ can be computed by a log-space uniform circuit family of size $T_0(n) = 2^{O(n^d)}$ and depth $\log(T_0)$.[27]. By Item (1), there is a function $f\colon \{0,1\}^n \to \{0,1\}^n$ computable in time $2^{n^{c_1}}$ such that there is an infinite subset $S \subseteq \mathbb{N}$ for which f is instance-wise hard for all polynomial-time probabilistic algorithms infinitely often. Consider the function $f_{\mathsf{idx}}\colon \{0,1\}^n \times [n] \to \{0,1\}$ that maps (x,i) to $f(x)_i$. With a natural Boolean encoding of $[n]$, we have that f_{idx} is computable in $2^{n^{c_1}}$ time using the algorithm for f. Hence, by the discussion above, $f_{\mathsf{idx}} \in \mathsf{MATIME}[n^d]$ and hence it can be computed by a logspace-uniform circuit family of size at most $T_0(2n)$ and depth $\log T_0(2n)$. This implies that for some $T(n) = 2^{O(n^d)}$, f can be computed by logspace-uniform circuit family of size at most T and depth $\log T$.

Now, fix an arbitrary prRP problem $\Pi = (\Pi_{\mathsf{yes}}, \Pi_{\mathsf{no}})$ solvable in time n^t, where $t \geq 1$ is a constant. That is, there is a deterministic Turing machine N taking two inputs x and y satisfying $|y| = n^t$, such that (1) $\Pr_{y \in \{0,1\}^{n^t}}[N(x,y) = 1] \geq 1/2$ for $x \in \Pi_{\mathsf{yes}}$ and (2) $\Pr_{y \in \{0,1\}^{n^t}}[N(x,y) = 1] = 0$ for $x \in \Pi_{\mathsf{no}}$. For convenience, we also define the set $A_x := \{y \in \{0,1\}^{n^t} : N(x,y) = 1\}$.

Let c be the constant in Proposition 12.5.1, let $M(n) = n^t$, and $\delta(n) = \tau \cdot (\log\log T(n) / \log T(n))$ for a large enough constant $\tau \geq 1$ so that $M(n) \leq T(n)^{\delta/c} = O(n^d)^{\tau/c}$ and $\log T(n) = O(n^d) \leq T(n)^{\delta/c}$. Note that without loss of generality, we can assume t is a large enough constant so that $M(n) = n^t > c \log T(n)$. Then, by Proposition 12.5.1 there is a generator H_f running in $T^{O(1/\delta)} = 2^{O(n^{d^2})}$ time such that, for every $x \in \Pi_{\mathsf{yes}}$ the following holds: If $H_f(x)$ does not hit the set A_x, plugging $N(x,\cdot)$ as the oracle of the reconstruction algorithm R, the resulting algorithm $R^{N(x,\cdot)}(x)$ computes $f(x)$ in $(\log T + n^c) \cdot T^\delta \cdot M^c \cdot n^t \leq n^\kappa$ time with probability at least $1 - 1/n$, for a constant $\kappa = \kappa(t,c,d)$.

Now consider the following algorithm A for solving Π: Given an input $x \in \{0,1\}^n$, enumerate all outputs y of the generator H_f; if any of them satisfy $N(x,y) = 1$ $A(x)$ output 1, and otherwise output 0. By Item (1), f is hard for the algorithm $R^{N(x,\cdot)}(x)$ infinitely often on almost all inputs. Let S be the corresponding infinite subset of $\mathbb{N}$ on which f is hard for $R^{N(x,\cdot)}(x)$. In the following, we show that for all $n \in S$, A computes Π on n-bit inputs.

Otherwise, for infinitely many input length $n \in S$, A fails to compute Π on some n-bit input. Note that A never errs on "no" instances, so this means that for every $n \in S$, there exists an input

[27] This is because we can construct a circuit that enumerates all possible n^d-length proofs in parallel, and verifies every proof deterministically by enumerating all possible n^d-bit strings as randomness in parallel. Note that the n^d-time verifier can be simulated by a logspace-uniform circuit of depth $O(n^d)$, using the standard tableau method.

$x_n \in \Pi_{\mathsf{yes}} \cap \{0,1\}^n$ (for x_n in neither Π_{no} nor Π_{yes}, A is allowed to output anything) such that $H_f(x_n)$ does not hit the set A_{x_n}. By previous discussions, this means that $R^{N(x_n,\cdot)}(x_n)$ computes $f(x_n)$ with probability at least $1 - 1/n$, for infinitely many $n \in S$ and the corresponding $x_n \in \{0,1\}^n$, contradicting to the fact that f is hard for the algorithm $R^{N(x,\cdot)}(x)$ infinitely often on almost all inputs. ■

Proof of (2) ⇒ (3). Assume that Item (2) holds for a constant $c_2 \geq 1$. We define a function $f\colon \{0,1\}^n \to \{0,1\}^n$ as follows:

1. Given an input $x \in \{0,1\}^n$ and an output index $i \in [n]$, let M_i be the i-th deterministic Turing machine.
2. Simulate M_i on x for $2^{n^{c_2+1}}$ steps to obtain its output $\sigma \in \{0,1\}^n$. If M_i does not terminate in $2^{n^{c_2+1}}$ steps or its output does not have length exactly n, we set $\sigma = 0^n$.
3. Otherwise, set $f(x)_i = 1 - \sigma_i$.

Let $c_3 \geq 1$ be a constant such that f is computable in $2^{n^{c_3}}$ time (*e.g.*, set $c_3 = c_2 + 2$), and let A be a polynomial-time probabilistic algorithm. We first establish the following claim.

Claim 12.5.18. *There is a $2^{n^{c_2+1}}$-time computable function $f_A\colon \{0,1\}^n \to \{0,1\}^n$ such that for infinitely many input lengths n and for every $x \in \{0,1\}^n$, if $A(x)$ computes $\sigma \in \{0,1\}^n$ with zero error and success probability at least $2/3$, then $f_A(x) = \sigma$.*

Proof. We first consider the following promise problem Π_A: On an input of length m, letting $n = \lfloor m/2 \rfloor$, it treats the first n bits as an input $x \in \{0,1\}^n$, the next $\lceil \log n \rceil$ bits as an index $i \in [n]$[28], and ignores the rest of the input. We say that an input (x,i) is in the promise of Π_A if it satisfies the following: There exists an output $\sigma \in \{0,1\}^n$ such that $\Pr[A(x) = \sigma] \geq 2/3$ and $\Pr[A(x) \in \{\sigma, \perp\}] = 1$. For such an (x,i) in the promise of Π_A, letting σ be the corresponding high-probability output of $A(x)$, we simply define $\Pi_A(x,i) = \sigma_i$. By a straightforward simulation algorithm, we have that $\Pi_A \in \mathsf{prZPP} \subseteq \mathsf{prRP}$.

By our hypothesis in Item (2), there is an infinite subset $S \subseteq \mathbb{N}$ and a $2^{n^{c_2}}$-time algorithm M_A such that for every $m \in S$ it holds that M_A solves Π_A for all inputs of length m. Now we consider the following two cases:

1. There are infinitely many even integers in S. In this case, we define f_A as follows: Given input $x \in \{0,1\}^n$, for each $i \in [n]$, we set $f_A(x)_i = M_A(x, i, 0^{n - \lceil \log n \rceil})$.

[28] If these bit correspond to an integer larger than n, Π_A truncates it to n.

2. There are finitely many even integers in S. Note that this implies there are infinitely many odd integers in S. In this case, we define f_A as follows: Given input $x \in \{0,1\}^n$, for each $i \in [n]$, we set $f_A(x)_i = M_A(x, i, 0^{n+1-\lceil \log n \rceil})$.

It is straightforward to verify that f_A in either cases satisfy the requirement. □

Now we are ready to show that f satisfies Item (3). Let A be a probabilistic randomized algorithm, and let f_A be the corresponding $2^{n^{c_2+1}}$-time algorithm guaranteed by Claim 12.5.18. Suppose that f_A is implemented by the i-th deterministic Turing machine. Then by Claim 12.5.18, for infinitely many input lengths $n \geq i$ and every $x \in \{0,1\}^n$, if $A(x)$ computes $\sigma \in \{0,1\}^n$ with zero error and success probability at least $2/3$, then $f(x)_i = 1 - \sigma_i \neq \sigma_i$. Therefore, f is hard for all polynomial-time probabilistic *zero-error* algorithms infinitely often on almost all inputs. ■

12.6 Non-black-box Derandomization from "non-batch-computability"

In this section we show our uniform hardness-to-randomness tradeoff for the setting of superfast derandomization, and in particular prove Theorem 12.1.7 and Theorem 12.1.8. For convenience we define the following notion of a probabilistic algorithm that *approximately-prints* a multi-output function.

Definition 12.6.1 (approximately-printing a function)**.** *Let A be a probabilistic algorithm, let $g\colon \{0,1\}^n \to \{0,1\}^k$, and let $x \in \{0,1\}^n$. For $\alpha \in [0,1]$, we say that A* approximately-prints $g(x)$ with error α *if with probability at least $1-\alpha$ it holds that $\Pr_{i\in[k]}[A(x)_i = g(x)_i] \geq 1-\alpha$.*

In Section 12.6.1 we show a generic construction of a reconstructive targeted PRG, and prove that it can be instantiated with a suitable hard function to obtain superfast derandomization. In Section 12.6.2 we use the foregoing results to prove Theorem 12.1.7 and Theorem 12.1.8 (in particular, Section 12.6.2 includes the formal definitions of the direct-product hypothesis underlying Theorem 12.1.7). And in Section 12.6.3 we prove that a "non-batch-computable" function is *necessary* for superfast derandomization in the natural special case of highly-uniform formulas.

12.6.1 Derandomization from Non-batch-computable Functions

The following construction of a reconstructive targeted PRG underlies our main results. Loosely speaking, the targeted PRG G is based on a function $g\colon \{0,1\}^n \to \{0,1\}^k$ such that the individual bits of g (*i.e.*, the mapping $(x,i) \mapsto g(x)_i$) can be computed in time T', and the result below shows a

reconstruction algorithm that, given access to x and to a distinguisher D_x for $G(x)$, approximately-prints $g(x)$ with error α' in time $T' \cdot n^{\beta'}$, where α' and β' are arbitrarily small constants.

Proposition 12.6.2 (a reconstructive targeted PRG)**.** *For every $\alpha', \beta' > 0$ and sufficiently small $\eta = \eta_{\alpha',\beta'} > 0$ the following holds. Let $T, k \colon \mathbb{N} \to \mathbb{N}$ be time-computable functions such that $T(n) \geq n$, and let $g \colon \{0,1\}^n \to \{0,1\}^k$ (where we denote $k = k(n)$) such that the mapping of $(x, i) \in \{0,1\}^n \times \{0,1\}^k$ to $g(x)_i$ is computable in time $T'(n)$. Then, there exist a deterministic algorithm G_g and a probabilistic algorithm R that for every $x \in \{0,1\}^n$ satisfy the following:*

1. **Generator.** *When G_g gets as input $x \in \{0,1\}^n$ and $\eta > 0$, it runs in time $k \cdot T'(n) + \mathrm{poly}(k)$ and outputs a set of strings in $\{0,1\}^{k^\eta}$.*
2. **Reconstruction.** *When R gets as input $x \in \{0,1\}^n$ and $\eta > 0$, and gets oracle access to a function $D_x \colon \{0,1\}^{k^\eta} \to \{0,1\}$ that $(1/k^\eta)$-distinguishes the uniform distribution over the output-set of $G_g(x, \eta)$ from a truly uniform string, it runs in time $\tilde{O}(k^{1+\beta'}) + k^{\beta'} \cdot T'(n)$, makes $\tilde{O}(k^{1+\beta'})$ queries to D_x, and with probability at least $1 - 2^{-k^\eta}$ outputs a string that agrees with $g(x)$ on at least $1 - \alpha'$ of the bits.*

Proof. In our proof we will use the following reconstructive PRG, which is based on a combination of parts from the classical constructions of Nisan and Wigderson [NW94] and of Impagliazzo and Wigderson [IW97]. The PRG gets as input a k-bit truth-table of a function $\{0,1\}^{\log(k)} \to \{0,1\}$, outputs a set of $\mathrm{poly}(k)$ strings of length k^η for a sufficiently small $\eta > 0$, and the crucial property for us is that its reconstruction algorithm is both *uniform* and of *very small time complexity* $k^{\beta'}$.

Theorem 12.6.3 (the PRG of [NW94, IW97] with reconstruction as a high-accuracy learning algorithm)**.** *For every two constants $\alpha', \beta' \in (0,1)$ there exist an oracle machine $G^{\mathtt{IW}}$ and a probabilistic oracle machine $R^{\mathtt{IW}}$ such that for every function $g \colon \{0,1\}^{\log(k)} \to \{0,1\}$ and sufficiently small $\eta = \eta_{\alpha',\beta'} > 0$ the following holds.*

- **Generator:** *When given input $(1^k, \eta)$ and oracle access to g, the machine $G^{\mathtt{IW}}$ runs in time $\mathrm{poly}(k)$ and outputs a set of strings in $\{0,1\}^m$, where $m = k^\eta$.*
- **Reconstruction:** *When given input $(1^k, \eta)$ and oracle access to a $(1/m)$-distinguisher D for $G^g(1^k, \eta)$ and to g, the machine $R^{\mathtt{IW}}$ runs in time $O(k^{\beta'})$ and with probability at least $1 - 2^{-2m}$ outputs outputs an oracle circuit that agrees with g on $1 - \alpha'$ of the inputs when given access to D.*

The proof of Theorem 12.6.3 is based on well-known constructions and observations from [NW94, IW98, IW97], but it involves many low-level details, so we defer the full proof to Section 12.7. In high-level, the PRG first encodes the function by the efficient derandomized direct-product

construction of [IW97] and by the Hadamard encoding, and then applies the PRG construction of [NW94] instantiated with very small output length k^η. To design a very fast uniform reconstruction algorithm, we rely on the observation of [IW98] that the reconstruction algorithm for the PRG of [NW94] is a uniform learning algorithm, and further note that when the output length is small (*i.e.*, k^η as in our setting), then the latter algorithm is also very fast (see Theorem 12.7.4 for details). Similarly, the list-decoding algorithms for the the direct-product construction of [IW97] and for the Hadamard encoding are both very fast uniform algorithms (see Theorem 12.7.5 for details). All these algorithms succeed with probability $1/\mathrm{poly}(m)$, and we repeat them to produce a list of candidate circuits that with high probability contains a circuit computing the initial function. We can then use our oracle access to the initial function in order to "weed" these lists and find a circuit that agrees with the function on $1-\alpha'$ of the inputs. See Section 12.7 for details.

Both the generator $G^{\mathtt{IW}}$ and the reconstruction $R^{\mathtt{IW}}$ in Theorem 12.6.3 are auxiliary protocols for the targeted generator G and reconstruction algorithm R that we construct. Specifically, G and R will instantiate $G^{\mathtt{IW}}$ and $R^{\mathtt{IW}}$ (respectively) with the k-bit truth-table $g(x) \in \{0,1\}^k$ that is a function of an n-bit input x, where x is given as explicit input both to G and to R. Details follow.

The pseudorandom generator $G = G_g$. Given as input $x \in \{0,1\}^n$ and parameter η, the algorithm G computes $g(x) \in \{0,1\}^k$ in time $k \cdot T'(n)$, and applies the generator $G^{\mathtt{IW}}$ from Theorem 12.6.3 with the constants α',β',η. The generator runs in time $\mathrm{poly}(k)$ and outputs a set of $\mathrm{poly}(k)$ strings $z_1,...,z_{\mathrm{poly}(k)} \in \{0,1\}^{k^\eta}$.

The reconstruction algorithm R. Given input $x \in \{0,1\}^n$ and parameter $\eta > 0$, the algorithm R runs the reconstruction algorithm $R^{\mathtt{IW}}$ from Theorem 12.6.3 with input $(1^k,\eta)$. The reconstruction algorithm needs oracle access to a $(1/k^\eta)$-distinguisher for $G_g(x,\eta)$, which we provide using D_x; and it needs oracle access to the function whose truth-table is $g(x)$, which we provide using the T'-time algorithm for the output bits of g. The running time of $R^{\mathtt{IW}}$ is $O(k^{\beta'})$, and therefore the number of queries to D_x is at most $O(k^{\beta'})$ and the running time of this step is $O(k^{\beta'} \cdot T')$.

With probability at least $1-2^{-2k^\eta}$, at this point we have a circuit C_i such that $C_i^{D_x}$ correctly computes at least $1-\alpha'$ of the coordinates of $g(x)$. We evaluate C_i on all of its k inputs, answering its queries with D_x, and output the truth-table of C_i. The running time of this step is $\tilde{O}(k^{1+\beta'})$, and thus the total running time is $O(k^{\beta'} \cdot T) + \tilde{O}(k^{1+\beta'})$. ■

The following hardness-to-randomness result is a direct consequence of the reconstructive tar-

geted PRG from Proposition 12.6.2. Loosely speaking, for a time function $T(n)$ and denoting $T' = T \cdot n^\varepsilon$, the hypothesis below is that one-way functions exist, and there exists a function $g\colon \{0,1\}^n \to \{0,1\}^k$ such that the mapping $(x,i) \mapsto g(x)_i$ is computable in time T', but $g(x)$ cannot be approximately-printed with small error in time $T' \cdot n^\varepsilon$, with high probability over x chosen from a distribution $\mathbf{x}$. Given this hypothesis, the result asserts that we can derandomize probabilistic time T with essentially no overhead n^ε, on average with respect to the distribution $\mathbf{x}$.

Theorem 12.6.4 (superfast non-black-box derandomization from a non-batch-computable function)**.** *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a polynomial, let $\delta'\colon \mathbb{N} \to (0,1)$, and let $\mathbf{x} = \{\mathbf{x}_n\}_{n\in\mathbb{N}}$ be a polynomial-time-samplable ensemble of distributions, where $\mathbf{x}_n$ is over $\{0,1\}^n$. Assume that one-way functions exist, and that for every $\varepsilon' > 0$ there exist $\alpha, \beta \in (0,1)$ and a function g mapping n bits to $n^{\varepsilon'}$ bits such that:*

1. *There exists an algorithm that on input $(x,i) \in \{0,1\}^n \times [n^{\varepsilon'}]$ outputs the i^{th} bit of $g(x)$ in time $T(n) \cdot n^{\varepsilon'}$.*
2. *For every probabilistic algorithm A that runs in time $T(n) \cdot n^{(1+\beta)\cdot\varepsilon'}$ and sufficiently large $n \in \mathbb{N}$, the probability over $x \sim \mathbf{x}_n$ that A approximately-prints $g(x)$ with error α is at most $\delta'(n)$.*

Then, for every $\varepsilon > 0$ there exists a deterministic algorithm $D = D_\varepsilon$ that runs in time $n^\varepsilon \cdot T(n)$ and prints $t < n^\varepsilon$ strings $w_1, \ldots, w_t \in \{0,1\}^{T(n)}$ such that the following holds. For every probabilistic machine M that runs in time T, with probability at least $1 - \delta'(n) - n^{-\omega(1)}$ over $x \sim \mathbf{x}_n$ it holds that

$$\left| \Pr_{r\in\{0,1\}^{T(n)}}[M(x,r) = 1] - \Pr_{i\in[t]}[M(x,w_i) = 1] \right| < n^{-\gamma}\,, \tag{12.6.1}$$

where $\gamma = \gamma_{\varepsilon,\beta} > 0$ is a sufficiently small constant. Moreover, the algorithm D does not depend on the distribution $\mathbf{x}$ or on the parameter δ' from the hardness hypotheses.

The algorithm in the conclusion of Theorem 12.6.4 is similar to a targeted PRG (as in Definition 12.3.3), but it only works on average-case over a choice of $x \sim \mathbf{x}_n$ rather than for every x. This "targeted PRG" has seed length $\log(t) < \varepsilon \cdot \log(n)$, and we will use it to deduce average-case derandomization with a multiplicative time overhead of n^ε.

Proof of Theorem 12.6.4. Let $\varepsilon' = \varepsilon/c$ for a sufficiently large constant $c > 1$ and let $k(n) = n^{\varepsilon'}$. Let g be the corresponding function from our hypothesis, and note that with our new notation:

1. Each output bit of g is computable in time $T' = T \cdot k$.
2. For every probabilistic algorithm A running in time $T \cdot k^{1+\beta}$, the probability over $x \sim \mathbf{x}_n$ that A approximately-prints $g(x)$ with error α is at most δ'.

The deterministic algorithm D. Fix a sufficiently small constant $\mu = \mu_{\varepsilon,\beta} > 0$. By Theorem 12.3.4, there exists a negl-PRG $G^{\mathtt{crypto}}$ for polynomial-time algorithms that has stretch $n^{\mu} \mapsto T$ and running time $T(n) \cdot n^{\mu}$. We will use the algorithm $G = G_g$ from Proposition 12.6.2, with parameters $\alpha' = \alpha$ and $\beta' = \beta/2$ and η such that $k^{\eta} = n^{\mu}$ (*i.e.*, $\eta = \mu/\varepsilon'$); note that if μ is sufficiently small then η is sufficiently small. On input $x \in \{0,1\}^n$, our algorithm first computes the output-set of $G_g(x,\eta)$, which consists of $\mathrm{poly}(k)$ strings $z_1, ..., z_{\mathrm{poly}(k)}$. Then, for each z_i it prints the string $w_i = G^{\mathtt{crypto}}(z_i) \in \{0,1\}^{T(n)}$.

Note that the number of strings that D prints is $\mathrm{poly}(k) < n^{\varepsilon}$ (relying on a sufficiently large choice of c) and that the total running time of D is at most

$$k \cdot T'(n) + \mathrm{poly}(k) \cdot T(n) \cdot n^{\mu} < n^{\varepsilon} \cdot T(n) \ .$$

Also note that the algorithm D does not depend on $\mathbf{x}$ or on δ', but only on the hard function g and on the parameters $\varepsilon, \alpha, \beta$.

Proof of correctness. Let M be a probabilistic machine that runs in time $T(n)$, and let γ be any constant smaller than μ. For a fixed input $x \in \{0,1\}^n$, denote by $\mathbf{z}_x$ the uniform distribution over $z_1, ..., z_{\mathrm{poly}(k)}$. Note that for the fixed x, if Eq. (12.6.1) does not hold, then it cannot be that

$$M(x, \mathbf{u}_T) \approx_{1/n} M(x, G^{\mathtt{crypto}}(\mathbf{u}_{n^{\mu}})) \approx_{n^{-\mu}} M(x, G^{\mathtt{crypto}}(\mathbf{z}_x)) \ , \tag{12.6.2}$$

where $\approx_{\alpha}$ means that two distributions are α-close in statistical distance. (This is because the distribution $M(x, G^{\mathtt{crypto}}(\mathbf{z}_x))$ in the RHS of Eq. (12.6.2) equals the distribution of $M(x, w_i)$ for a uniform $i \in [n^{\varepsilon}]$, by the definition of the w_i's and of $\mathbf{z}_x$.)

We first claim that the probability over $x \sim \mathbf{x}_n$ that $M(x, \mathbf{u}_T) \not\approx_{1/n} M(x, G^{\mathtt{crypto}}(\mathbf{u}_{n^{\mu}}))$ is $n^{-\omega(1)}$; that is:

Claim 12.6.5. *Let $S_n = \{x \in \{0,1\}^n : M(x, \mathbf{u}_T)$ is $(1/n)$-far from $M(x, G^{\mathtt{crypto}}(\mathbf{u}_{n^{\mu}}))\}$. Then, $\Pr[\mathbf{x}_n \in S_n] \leq n^{-\omega(1)}$.*

Proof. The proof amounts to constructing a polynomial-time distinguisher for $G^{\mathtt{crypto}}$ under the assumption $\Pr[\mathbf{x}_n \in S_n] \geq 1/\mathrm{poly}(n)$. The crucial point in the construction is that the distinguisher can sample $x \sim \mathbf{x}_n$ and simulate M in polynomial time (since $\mathbf{x}_n$ is polynomial-time samplable and M runs in polynomial time). Details follow.

Assume that $\Pr[\mathbf{x}_n \in S_n] \geq 1/p(n)$ for some polynomial p. For every $x \in \{0,1\}^n$, denote $u(x) = \Pr[M(x, \mathbf{u}_T) = 1]$ and $w(x) = \Pr[M(x, G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 1]$. Our distinguisher F for $G^{\mathtt{crypto}}$ gets input $r \in \{0,1\}^T$, samples $x \sim \mathbf{x}_n$, estimates the values $u(x)$ and $w(x)$ each up to error $1/(4n \cdot p(n))$ and with confidence $1 - 2^{-2n}$, obtaining estimates $\widetilde{u}(x)$ and $\widetilde{w}(x)$ respectively, and outputs

$$F(r) = \begin{cases} M(x,r) & \widetilde{u}(x) > \widetilde{w}(x) \\ 1 - M(x,r) & \widetilde{u}(x) < \widetilde{w}(x) \end{cases} .$$

Note that F runs in polynomial time, since T is a polynomial and $\mathbf{x}$ is polynomial-time-samplable and p is a polynomial. To see that F is an $n^{-O(1)}$-distinguisher for $G^{\mathtt{crypto}}$, let

$$\mathtt{FAR} = \left\{ x : \left| u(x) - w(x) \right| > 1/(2n \cdot p(n)) \right\} ,$$

and note that $S_n \subseteq \mathtt{FAR}$. We further partition $\mathtt{FAR}$ into $\mathtt{FAR}^{(>)} = \{x \in \mathtt{FAR} : u(x) > w(x)\}$ and $\mathtt{FAR}^{(<)} = \{x \in \mathtt{FAR} : u(x) < w(x)\}$. Denote the random variable representing the coins that F uses for estimation by $\mathbf{v}$, and note that with probability at least $1 - 2^{-n}$ over $v \sim \mathbf{v}$ the following event, denoted by $\mathcal{V}$, happens: Conditioned on any choice of $x \in \mathtt{FAR}^{(>)}$ we have that $F(r) = M(x,r)$, and conditioned on any choice of $x \in \mathtt{FAR}^{(<)}$ we have that $F(r) = 1 - M(x,r)$. Then,

$$\begin{aligned}
&\Pr[F(\mathbf{u}_T) = 1] - \Pr[F(G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 1] \\
&\quad = \mathop{\mathbb{E}}_{v \sim \mathbf{v}, x \sim \mathbf{x}_n} [\Pr[F(\mathbf{u}_T) = 1 | v, x] - \Pr[F(G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 1 | v, x]] \\
&\quad \geq \mathop{\mathbb{E}}_{x \sim \mathbf{x}_n} [\Pr[F(\mathbf{u}_T) = 1 | \mathcal{V}, x] - \Pr[F(G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 1 | \mathcal{V}, x]] - 2^{-n} \\
&\quad \geq \Pr\left[\mathbf{x}_n \in \mathtt{FAR}^{(>)}\right] \cdot \mathop{\mathbb{E}}_{x \sim \mathbf{x}_n} \left[\Pr[F(\mathbf{u}_T) = 1] - \Pr[F(G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 1] | x \in \mathtt{FAR}^{(>)}, \mathcal{V}\right] \\
&\qquad + \Pr\left[\mathbf{x}_n \in \mathtt{FAR}^{(<)}\right] \cdot \mathop{\mathbb{E}}_{x \sim \mathbf{x}_n} \left[\Pr[F(\mathbf{u}_T) = 1] - \Pr[F(G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 1] | x \in \mathtt{FAR}^{(<)}, \mathcal{V}\right] \\
&\qquad + \Pr[\mathbf{x}_n \notin \mathtt{FAR}] \cdot (-1/(2n \cdot p(n))) - 2^{-n} \\
&\quad = \Pr\left[\mathbf{x}_n \in \mathtt{FAR}^{(>)}\right] \cdot \mathop{\mathbb{E}}_{x \sim \mathbf{x}_n} \left[\Pr[M(x, \mathbf{u}_T) = 1] - \Pr[M(x, G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 1] | x \in \mathtt{FAR}^{(>)}, \mathcal{V}\right] \\
&\qquad + \Pr\left[\mathbf{x}_n \in \mathtt{FAR}^{(<)}\right] \cdot \mathop{\mathbb{E}}_{x \sim \mathbf{x}_n} \left[\Pr[M(x, \mathbf{u}_T) = 0] - \Pr[M(x, G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu})) = 0] | x \in \mathtt{FAR}^{(<)}, \mathcal{V}\right] \\
&\qquad - 1/(2n \cdot p(n)) - 2^{-n}.
\end{aligned}$$

From the definition of $u(x)$ and $w(x)$, we further have

$$\begin{aligned}
&\Pr[F(\mathbf{u}_T)=1]-\Pr[F(G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu}))=1] \\
&\quad > \Pr\left[\mathbf{x}_n \in \mathtt{FAR}^{(>)}\right] \cdot \mathop{\mathbb{E}}_{x\sim\mathbf{x}_n}\left[u(x)-w(x) | x \in \mathtt{FAR}^{(>)}, \mathcal{V}\right] \\
&\qquad + \Pr\left[\mathbf{x}_n \in \mathtt{FAR}^{(<)}\right] \cdot \mathop{\mathbb{E}}_{x\sim\mathbf{x}_n}\left[(1-u(x))-(1-w(x)) | x \in \mathtt{FAR}^{(<)}, \mathcal{V}\right] \\
&\qquad - 1/(2n\cdot p(n)) - 2^{-n} \\
&\quad \geq \Pr[\mathbf{x}_n \in \mathtt{FAR}] \cdot \mathop{\mathbb{E}}_{x\sim\mathbf{x}_n}\left[\left|u(x)-w(x)\right| | x \in \mathtt{FAR}, \mathcal{V}\right] - 1/p(n)^2 \\
&\quad \geq \Pr[\mathbf{x}_n \in S_n] \cdot \mathop{\mathbb{E}}_{x\sim\mathbf{x}_n}\left[\left|u(x)-w(x)\right| | x \in S_n, \mathcal{V}\right] - 1/(2n\cdot p(n)) - 2^{-n} \\
&\quad \geq 1/(n\cdot p(n)) - 1/(2n\cdot p(n)) - 2^{-n} \,,
\end{aligned}$$

where the last inequality is since $\Pr[\mathbf{x}_n \in S_n] \geq 1/p(n)$ and for every $x \in S_n$ it holds that $\left|u(x)-w(x)\right| > 1/n$. Thus, for a sufficiently large n the algorithm F is a $(1/(3n\cdot p(n)))$-distinguisher for $G^{\mathtt{crypto}}$, a contradiction. □

Now, let $D_x(r) = M(x, G^{\mathtt{crypto}}(r))$, and observe that for every $x \in \{0,1\}^n$ such that $M(x, G^{\mathtt{crypto}}(\mathbf{u}_{n^\mu}))$ is $(n^{-\mu})$-far from $M(x, G^{\mathtt{crypto}}(\mathbf{z}_x))$ it holds that D_x is a $(n^{-\mu})$-distinguisher for $\mathbf{z}_x$. We will now construct a probabilistic algorithm A that runs in time $k^{1+\beta}\cdot T(n)$ and approximately-prints $g(x)$ with error α for any such x. By our assumption, the probability over $x \sim \mathbf{x}_n$ that A approximately-prints $g(x)$ with error α is at most δ'. Thus, by a union bound, we will conclude that the probability over $x \sim \mathbf{x}_n$ that Eq. (12.6.2) does not hold, which upper-bound the probability over $x \sim \mathbf{x}_n$ that our derandomization errs on x, is at most $\delta' + n^{-\omega(1)}$.

Our algorithm A runs the reconstruction algorithm R from Proposition 12.6.2 with inputs (x,η), and while answering its oracles queries to D_x by simulating $G^{\mathtt{crypto}}$ and M. The reconstruction R requires a $(k^{-\eta})$-distinguisher, and recall that $k^\eta = n^\mu$ by our parameter choices. The total running-time of A is

$$\tilde{O}(k^{1+\beta/2}) + k^{\beta/2}\cdot T' + \tilde{O}(k^{1+\beta/2})\cdot(T\cdot n^\mu) = o\left(k^{1+\beta}\cdot T(n)\right) \,,$$

where we relied on the fact that R makes $\tilde{O}(k^{1+\beta/2})$ oracle queries, and that we can answer each oracle query in time $O(T\cdot n^\mu)$. With probability at least $1-2^{-k^\mu} > 1-\alpha$ it outputs a string that agrees with $g(x)$ on at least $1-\alpha$ of the coordinates. ■

12.6.2 Proofs of Theorem 12.1.7, Theorem 12.1.8, and of Corollary 12.1.9

In this section we show that Theorem 12.1.7 and Theorem 12.1.8 follow as corollaries of the hardness-to-randomness tradeoff in Theorem 12.6.4. Let us first formally state Theorem 12.1.8 and prove it:

Theorem 12.6.6 (superfast non-black-box derandomization over all polynomial-time-samplable distributions; Theorem 12.1.8, restated)**.** *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a polynomial. Assume that one-way functions exist, and that for every $\varepsilon > 0$ there exist $\delta > 0$ and a function g mapping n bits to n^ε bits such that:*

1. *There exists an algorithm that gets input $(x, i) \in \{0,1\}^n \times [n^\varepsilon]$ and outputs the i^{th} bit of $g(x)$ in time $T'(n) = T(n) \cdot n^\varepsilon$.*
2. *For every probabilistic algorithm A that runs in time $T'(n) \cdot n^\delta$ and every polynomial-time samplable distribution $\mathbf{x}$ and every sufficiently large n, the probability over $x \sim \mathbf{x}_n$ that A approximately-prints $g(x)$ with error .01 is negligible.*

Then $\mathsf{BPTIME}[T] \subseteq \bigcap_{\varepsilon>0} \mathsf{heur}_{1-\mathsf{negl}}\text{-}\mathsf{DTIME}[n^\varepsilon \cdot T]$, where negl is a negligible function.

Proof. Let $L \in \mathsf{BPTIME}[T]$, let M be a probabilistic machine that decides L in time T, and let $\varepsilon > 0$. We invoke Theorem 12.6.4 with δ' that is a negligible function, with $\alpha = .01$ and $\beta = \delta/\varepsilon$, and with *every* polynomial-time-samplable distribution $\mathbf{x}$. We obtain a deterministic algorithm D_ε printing strings that satisfy Eq. (12.6.1), and by the "moreover" part, the algorithm D_ε does not depend on $\mathbf{x}$, so we obtain the same algorithm D_ε for all choices of $\mathbf{x}$.

Our deterministic simulation A_L gets input $x \in \{0,1\}^n$ and outputs the majority value of $M(x, w_i)$ over all w_i's that D_ε prints. Its running time is $O(n^\varepsilon \cdot T(n))$, since D_ε runs in time $n^\varepsilon \cdot T(n)$ and prints n^ε strings, and its correctness follows by the conclusion of Theorem 12.6.4. ■

Remark 12.6.7. A non-black-box derandomization approach (such as our targeted PRGs) is *necessary* to obtain derandomization as in the conclusion of Theorem 12.6.4; that is, to obtain derandomization with a multiplicative overhead of less than n that works over arbitrary polynomial-time samplable distributions. To see this, fix any HSG G that supposedly derandomizes $\mathsf{prBPTIME}[T]$ on average using $n^{1-\varepsilon}$ seeds (*i.e.*, seed length $(1-\varepsilon) \cdot \log(n)$) and running time T'. Then, for a suitable problem in $\mathsf{prBPTIME}[T]$, we can construct in deterministic time $O(n \cdot T')$ an input on which the corresponding derandomization errs. Specifically, we enumerate the outputs of G and construct a circuit that rejects these strings and accepts all other strings; this circuit is an input for

$\mathsf{CAPP} \in \mathsf{prBPTIME}[n] \subseteq \mathsf{prBPTIME}[T]$ on which the derandomization errs.[29]

Since in Theorem 12.6.4 we construct a targeted PRG rather than just an algorithm that approximates the acceptance probability of a circuit, we are also able to deterministically *find* good random strings for probabilistic machines.[30] As stated in Corollary 12.1.9, this implies in particular that, under a hypothesis as in Theorem 12.6.6, the complexity of explicitly constructing combinatorial objects is nearly identical to the complexity of verifying that the combinatorial object meets the required specification. Let us formally define this notion and prove Corollary 12.1.9:

Definition 12.6.8 (explicit constructions)**.** *Let $\Pi \subseteq \{0,1\}^*$ such that for every sufficiently large $n \in \mathbb{N}$ it holds that $\Pi \cap \{0,1\}^n \neq \emptyset$. We say that a deterministic algorithm A is an* explicit construction of Π *if for every sufficiently large n, when A gets input 1^n it outputs an n-bit string in Π.*

Theorem 12.6.9 (explicit constructions in time that nearly matches the verification time; Corollary 12.1.9, restated)**.** *Let $\Pi \in \mathsf{DTIME}[T]$ such that for every sufficiently large $n \in \mathbb{N}$ it holds that $|\Pi_n| \geq 2^n/n^{o(1)}$, where $\Pi_n = \Pi \cap \{0,1\}^n$. Then, under the assumption of Theorem 12.6.6, for every $\varepsilon > 0$ there exists an explicit construction of Π in time $n^\varepsilon \cdot T(n)$. Moreover, if the assumption of Theorem 12.6.6 holds for* every *polynomial T, then for every polynomial T we can deduce the conclusion while only assuming that $\Pi \in \mathsf{BPTIME}[T]$.*

Proof. Given $\varepsilon > 0$, we invoke Theorem 12.6.4 with parameters δ', α, β as in the proof of Theorem 12.6.6 and with $\mathbf{x}$ such that $\mathbf{x}_n$ is supported only on the string 1^n (this is a polynomial-time-samplable distribution). Let $D_{T,\varepsilon}$ be the deterministic algorithm from the conclusion of Theorem 12.6.4.

Let R_Π be a probabilistic machine that gets input $x \in \{0,1\}^n$, randomly chooses $\pi \in \{0,1\}^n$ and accepts if and only if $\pi \in \Pi$ (regardless of x). Note that R_Π runs in time $O(T(n))$. Our explicit construction of Π gets input 1^n, enumerates over the strings $w_1, ..., w_{n^\varepsilon}$ that $D_{T,\varepsilon}(1^n)$ prints, truncates each string to n bits (recall that the original strings are of length $T(n)$), and outputs the lexicographically-first truncated string r such that $R_\Pi(1^n, r) = 1$.

The foregoing algorithm runs in time $O(n^\varepsilon \cdot T(n))$. To see that it is an explicit construction of Π, note that by Theorem 12.6.4, the probability over $x \sim \mathbf{x}_n$ that Eq. (12.6.1) is violated is less than 1; since $\mathbf{x}_n$ is just supported on 1^n, we deduce that Eq. (12.6.1) is satisfied for the machine R_Π above

[29] With some care, one can strengthen this impossibility argument so that it holds also for $\mathsf{BPTIME}[T]$ rather than just $\mathsf{prBPTIME}[T]$ (to do so we replace CAPP with PIT and construct an arithmetic circuit rather than a Boolean one).

[30] There is a known search-to-decision reduction in this context, first pointed out by Goldreich [Gol11a], but a-priori this reduction has a polynomial runtime overhead. See further discussion after the proof of Theorem 12.6.9.

with input 1^n. Also, by our hypothesis that $|\Pi_n| \geq 2^n/n^{o(1)}$ and our definition of R_Π, we have that $\Pr[R_\Pi(1^n, \mathbf{u}_n) = 1] \geq n^{-o(1)}$. Hence, there always exists w_i such that $M(1^n, w_i) = 1$, which means that our algorithm indeed outputs some $\pi \in \Pi$.

For the "moreover" part, we cannot just define R_Π as above since we do not assume that $\Pi \in \mathsf{DTIME}[T]$. However, we assumed that there exists a probabilistic machine M that decides Π in time T. We define D_Π to be the machine that, given input $\pi \in \{0,1\}^n$, outputs the majority vote of $M(\pi, w_i)$ over the strings $w_1, ..., w_{n^\varepsilon}$ that $D_{T,\varepsilon}(\pi)$ outputs. Note that D_Π runs in time $T'(n) = O(n^\varepsilon \cdot T(n))$ and that no probabilistic polynomial-time algorithm can find a string π such that $D_\Pi(\pi) \neq \Pi(\pi)$, with non-negligible probability.

We now define R_Π as above, using D_Π as the deterministic decision algorithm for Π (*i.e.*, R_Π ignores its input $x \in \{0,1\}^n$, draws a random $\pi \in \{0,1\}^n$, and accepts iff $D_\Pi(\pi) = 1$). Since R_Π now runs in time $T'(n)$, we invoke Theorem 12.6.4 with time bound T' and with the same $\delta', \alpha, \beta, \mathbf{x}$ to obtain a deterministic algorithm $D_{T',\varepsilon}$. Our explicit construction works just as above, only using the output strings of $D_{T',\varepsilon}$ (instead of $D_{T,\varepsilon}$) and evaluating them with the new R_Π that uses D_Π. To see that it is an explicit construction, the precise same analysis as above implies that our algorithm outputs π such that $D_\Pi(\pi) = 1$. Since this is a deterministic polynomial-time algorithm, by the properties of D_Π it cannot be that $\pi \notin \Pi$. ■

Remark 12.6.10. Under the particular hypotheses of Theorem 12.6.9 a different proof approach might be possible. Moreover, this alternative potential approach is more general, and does not need to assume that the superfast derandomization involves targeted PRGs (*i.e.*, it only uses the generic derandomization $\mathsf{BPTIME}[T] \subseteq \bigcap_{\varepsilon>0} \mathsf{heur}_{1-\mathsf{negl}}\text{-}\mathsf{DTIME}[n^\varepsilon \cdot T]$). Specifically, when we assume one-way functions (as in Theorem 12.6.9), we can rely on the fast PRG in Theorem 12.3.4 to reduce the search-space of any explicit construction algorithm A from $\{0,1\}^n$ to $\{0,1\}^{n^\varepsilon}$, for an arbitrarily small $\varepsilon > 0$, without significantly affecting its running time or the relative density of valid outputs.[31] The next step is to apply a search-to-decision reduction as in [Gol11a], relying on the fact that the number of iterations is only n^ε for an arbitrarily small $\varepsilon > 0$, and that it is infeasible to find in polynomial time an input on which the derandomization fails. While this proof approach might work, we prefer the approach presented in Theorem 12.6.9, both due to its simplicity and since it demonstrates the general point that targeted PRGs can be used directly to

[31] This is the case since we can replace the algorithm D_Π that decides Π by an algorithm D'_Π that gets as input a seed $s \in \{0,1\}^{n^\varepsilon}$, expands it to an n-bit string using the PRG from Theorem 12.3.4, and accepts iff D_Π accepts. Thus, instead of finding a good object in $\{0,1\}^n$ (*i.e.*, a string that D_Π accepts), it now suffices to find a good seed in $\{0,1\}^{n^\varepsilon}$ (*i.e.*, a string that D'_Π accepts).

solve search problems (regardless of whether or not OWFs exist).

Let us now turn to proving Theorem 12.1.7. Recall that the result relies on a direct product hypothesis. To introduce this hypothesis we first establish what can be *unconditionally proved*, as a basis for comparison. For a function $f\colon \{0,1\}^n \to \{0,1\}$, denote by $f^{\times k}$ the k-wise direct-product $f^k(x_1,...,x_k) = (f(x_1),...,f(x_k)) \in \{0,1\}^k$. As mentioned in Section 12.1.2, it is *unconditionally* true that if a function f is average-case hard to compute in probabilistic time T, then $f^{\times k}(x)$ cannot be *approximately-printed* in similar time $\Omega(T)$ for the vast majority of inputs x.

To be more accurate, we will need a slightly stronger assumption, which was also needed in previous works that studied uniform direct-product results (see, *e.g.*, [IJKW10], following [STV01, TV07]). Specifically, we will assume that hardness holds with respect to probabilistic algorithms that get a constant number of advice bits that depend on the *randomness* of the algorithm but still do not depend on the specific *input* to the algorithm. The class of problems solvable by probabilistic algorithms running in time T with a bits of randomness-dependent advice is denoted by $\mathsf{BPTIME}[T]//a$ (see [IJKW10] for further details and explanations). Then:

Proposition 12.6.11 (non-strong approximate-direct-product theorem)**.** *There exists a universal constant $c > 1$ such that the following holds. Let $\delta \in (0,1/2)$, let $\alpha > 0$ be sufficiently small, let $T\colon \mathbb{N} \to \mathbb{N}$ be time-computable, and let $k(n) = o(n)$. Then, for any $f \notin \mathsf{avg}_{1-\delta}\text{-}\mathsf{BPTIME}[T]//(1/\alpha)$ and any probabilistic algorithm A that runs in time $T(n) - n^c$, the probability over $z \in \{0,1\}^{k \cdot n}$ that A approximately-prints $f^{\times k}(z)$ with error α is at most δ.*

Proposition 12.6.11 follows as a corollary of the uniform list-decoding algorithms of Impagliazzo *et al.* [IJKW10] for the direct-product code (in particular, of [IJKW10, Theorem 1.8]). Since their paper uses slightly different notions (*i.e.*, it refers to a direct-product function $f^{\times k}(x_1,...,x_k)$ that only takes *distinct* inputs $x_1,...,x_k$, and disallows $x_i = x_j$ for any $i \neq j \in [k]$), we include for completeness a proof of Proposition 12.6.11 in Section 12.9.

The following hypothesis essentially says that Proposition 12.6.11 can be strengthened, for *some* function that is hard for probabilistic time T, to assert that the hardness of $f^{\times k}$ holds not only with respect to algorithms running in time $T(n) - n^c$, but also with respect to algorithms running in time $T(n) \cdot k^\beta$, for some (arbitrarily small) constant $\beta > 0$. Specifically, the definition below is a more general version of the assumption that was stated in Section 12.1.2.

Assumption 12.6.12 (mildly-strong approximate-direct-product hypothesis)**.** *For any $\delta > 0$, the* mildly-strong approximate-direct-product hypothesis for time $T(n)$ and arity $k(n)$ and average-case

success δ *asserts that there exist* $\alpha, \beta > 0$ *and a function* $f \in \mathsf{DTIME}[T]$ *such that the following holds. For any probabilistic algorithm* A *that runs in time* $O(k^\beta \cdot T)$*, the probability over* $z \in \{0,1\}^{k \cdot n}$ *that* A *approximately-prints* $f^{\times k}(z)$ *with error* α *is at most* δ*.*

Now we prove Theorem 12.6.13: Assuming that the mildly-strong approximate-direct-product hypothesis holds, and that one-way functions exist, probabilistic algorithms running in time T can be deterministically simulated with essentially no overhead, *i.e.* in time $T \cdot n^\varepsilon$, on average over the uniform distribution.

Theorem 12.6.13 (superfast non-black-box derandomization from mildly-strong approximate-direct-product; Theorem 12.1.7, restated)**.** *Let* $T\colon \mathbb{N} \to \mathbb{N}$ *be a polynomial. Assume that there exist one-way functions secure against polynomial-time algorithms, and that for some* $\delta > 0$ *and every* $\varepsilon_0, \varepsilon_1 > 0$ *the mildly-strong approximate-direct-product hypothesis holds for time* $T(n^{1-\varepsilon_0}) \cdot n^{\varepsilon_1}$ *and arity* n^{ε_1} *and average-case success* δ*. Then* $\mathsf{BPTIME}[T] \subseteq \bigcap_{\varepsilon>0} \mathsf{avg}_{1-\delta'}\text{-}\mathsf{DTIME}[n^\varepsilon \cdot T]$*, where* $\delta'(n) = \delta(n) + n^{-\omega(1)}$*.*

Proof. We show that for any $\varepsilon' > 0$ there exist $\alpha, \beta > 0$ and a function g as in the hypothesis of Theorem 12.6.4 with $\mathbf{x}$ being the uniform distribution (*i.e.*, for every $n \in \mathbb{N}$ it holds that $\mathbf{x}_n \equiv \mathbf{u}_n$). To see this, fix any $\varepsilon' > 0$, and for $m(n) = n^{1/(1-\varepsilon')}$ let $T'\colon \mathbb{N} \to \mathbb{N}$ such that $T'(n) = T(m(n)) \cdot m(n)^{\varepsilon'}$ (*i.e.*, $T'(n) = T(n^{1/(1-\varepsilon')}) \cdot n^{\varepsilon'/(1-\varepsilon')}$). Let $f \in \mathsf{DTIME}[T']$ be a function satisfying the mildly-strong approximate-direct-product hypothesis for time T' and arity $k(n) = m(n)^{\varepsilon'} = n^{\varepsilon'/(1-\varepsilon')}$ and average-case success δ. By our hypothesis, the output bits of the function $g = f^{\times k}$ can be computed in time $T'(n) = T(m) \cdot m^{\varepsilon'}$, whereas there exist $\alpha, \beta > 0$ such that $g(z)$ cannot be approximately-printed with error α on more than δ of the inputs z in probabilistic time $T'(n) \cdot m(n)^\beta = T(m(n)) \cdot m(n)^{(1+\beta)\cdot\varepsilon'}$.

Let $\varepsilon > 0$, let D_ε be the deterministic algorithm from Theorem 12.6.4, and for every $x \in \{0,1\}^n$ let $w_1(x), ..., w_{n^\varepsilon}(x)$ be the strings that $D_\varepsilon(x)$ prints. For any $L \in \mathsf{BPTIME}[T]$ and $\varepsilon > 0$, our deterministic algorithm outputs the majority value of $M_L(x, w_i(x))$ over all $i \in [n^\varepsilon]$, where M_L is a probabilistic T-time machine that decides L. This algorithm runs in time $O(n^\varepsilon \cdot T(n))$ and correctly decides L with probability at least $1 - \delta'$ over choice of $x \in \{0,1\}^n$. ■

12.6.3 Necessity of Non-batch-computable Functions

We now show one natural setting in which superfast derandomization necessitates the existence of a multi-output function $g\colon \{0,1\}^n \to \{0,1\}^{n^\varepsilon}$ whose individual bits can be computed in time T, but that cannot be computed in time significantly less than $T \cdot n^\varepsilon$.

The computational model is that of balanced formulas of fan-in two over the full binary basis that are highly uniform (see the precise uniformity condition below). The size of a formula is the number of leaves, and is typically denoted by T. We assume for simplicity that the size is always a power of two, that internal gates are over the full binary basis, and that negations appear only in the bottom level (*i.e.*, we only negate variables).

Probabilistic uniform formulas can be defined in several ways, since randomness can be used either by the machine that constructs the formula or by the formula itself (or by both); moreover, when randomness is used by the formula itself, it can be either read-once or reusable.[32] Our result works in any of these models, and in none of the models does there seem to be a trivial derandomization with the requirements that are specified below (*i.e.*, it does not seem trivial to simulate highly-uniform probabilistic formulas by slightly larger highly-uniform deterministic formulas). For concreteness, we define probabilistic formulas as ones in which the formula uses randomness in a read-once manner: That is, some leaves are labeled by a special symbol R, which indicates that the leaf tosses an independent random coin (and the probabilistic computation of the formula on a fixed input is over a uniform distribution for the values of leaves that are labeled by R).

Fixing a canonical indexing of the $[2T-1]$ gates in a balanced formula of size T such that the leaves are indexed by $[T]$, the formula is uniquely determined by the labels of the internal gates (*i.e.*, the function that they compute) and by the labels of the leaves (*i.e.*, each leaf is labeled by a literal, or by a constant in $\{0,1\}$, or by R). In particular, such a formula is uniquely described by its label function:

Definition 12.6.14 (label function)**.** *For a formula F_n of size T over n variables, the* labels function of F_n*, denoted $\Phi_F \colon [2T-1] \to [2n] \cup \{\mathsf{T},\mathsf{F},\mathsf{R}\} \cup \mathfrak{B}_2$, maps each gate in F_n to its label, where a label in $[2n]$ is interpreted as a literal, the labels T and F stand for the constants 1 and 0 respectively, the label R indicates that this is a leaf gate that tosses a random coin, and $\mathfrak{B}_2$ is the full binary basis (i.e., the set of all functions $\{0,1\}^2 \to \{0,1\}$).*

The uniformity condition that we impose on the formulas is similar to what is known as DPOLYLOGTIME uniformity. Specifically, we require that the label function of a size-T formula can be computed in time $\mathrm{polylog}(T)$ (*i.e.*, polynomial in the input length to the label function).

[32] That is, in the former case leaves that are random bits are independent, and in the latter case the same random bit can reappear as the label of several leafs.

Definition 12.6.15 (well-structured formulas)**.** *For $c \geq 1$, we say that an n-bit formula F of size T is* c-well-structured *if F is balanced and its label function Φ_F is computable in time $\log(T)^c$.*

The following statement asserts that superfast derandomization of well-structured formulas implies the existence of a "non-batch-computable" function as we wanted.

Proposition 12.6.16 (necessity of non-batch-computable functions for derandomization of well-structured formulas)**.** *Let c be any sufficiently large constant, let $c_1 > c$, let $\varepsilon, \delta \in (0,1)$ such that $\varepsilon > \delta$, and let $T \colon \mathbb{N} \to \mathbb{N}$ be a polynomial.*

- **Assumption:** *Assume that for every polynomial $T_0 \colon \mathbb{N} \to \mathbb{N}$, every function computable by a c-well-structured family of probabilistic formulas of size T_0 is also computable, for every sufficiently large $n \in \mathbb{N}$ and on more than 0.99 of the inputs $x \in \{0,1\}^n$, by a c_1-well-structured family of deterministic formulas of size $T_0(n) \cdot n^\delta$.*
- **Conclusion:** *Then, there exists $g \colon \{0,1\}^n \to \{0,1\}^{n^\varepsilon}$ such that the mapping $(x,i) \mapsto g(x)_i$ is computable in time $\tilde{O}(T)$, but the following holds. For any c-well-structured family $\{D_n\}$ of probabilistic formulas of size $o(T(n) \cdot n^{\varepsilon-\delta})$ there exists infinitely many input lengths $n \in \mathbb{N}$ such that the fraction of $x \in \{0,1\}^n$ for which $\Pr[D_n(x) = g(x)] \geq 2/3$ is less than 0.01.*

High-level proof idea. Let us explain the proof idea before presenting the full proof. For $T' = T \cdot n^\varepsilon$, consider well-structured formulas of size T' whose top part computes the parity function of the gates at depth $k = \varepsilon \cdot \log(n)$. We first diagonalize against this class using a function $L^{\mathtt{diag}}$ that is itself computable by a well-structued formula family $\left\{F_n^{\mathtt{diag}}\right\}_{n \in \mathbb{N}}$ of the same size T'. Specifically, for every $n \in \mathbb{N}$, let F_n be the well-structured formula that is described by the uniform machine whose index is n. [33] The diagonalizing formula $F_n^{\mathtt{diag}}$ has the precise same structure as F_n, the only difference being that exactly one sub-formula in level k computes the negation of the corresponding sub-formula in F_n. This is indeed a diagonalization, since whenever the top $k-1$ levels compute the parity function we have that $F_n^{\mathtt{diag}}(x) \neq F_n(x)$ for all inputs $x \in \{0,1\}^n$.

Now, the hard function g gets input $x \in \{0,1\}^n$ and outputs the values of gates in level k of $F_n^{\mathtt{diag}}(x)$. Observe that each output bit of g is computable in time $\tilde{O}(T)$, since the formulas for $L^{\mathtt{diag}}$ are well-structured (*i.e.*, balanced and with an efficiently-computable label function), which means that each output bit corresponds to a subformula of size $T'/n^\varepsilon = T$. To see that g is hard, assume that it can be computed by a family of well-structured probabilistic formulas $\{D_n\}$ of size

[33] We may not assume that every machine describes a well-structured formula, but we can nevertheless correct non-valid outputs in some canonical way.

$T_0 = o(T')$ that is "too small". By our assumption, it can also be computed by a family $\{D'_n\}$ of well-structured deterministic formulas, which (for simplicity) we now assume are of precisely the same size T_0. We now define a well-structured formula family $\{D''_n\}$ that first uses D'_n to compute g, and then computes the parity of the outcomes. For every n that is associated with the uniform machine that defines D''_n, on the one hand we have that D''_n computes the same function as $F_n^{\mathtt{diag}}$ (since both of them compute the parity of the values of g), but on the other hand D''_n is of size $o(T')$ – a contradiction to the diagonalizing properties of $F^{\mathtt{diag}}$.

We are left with just an issue, which is that our derandomization does not succeed in worst-case, but only on $\mu > 0.99$ of inputs. To deal with this issue, recall that the function $\left\{F_n^{\mathtt{diag}}\right\}$ diagonalizes against every M on *all* inputs of the corresponding length n. Thus, it suffices to assume that the derandomization of the probabilistic formulas of size T_0 that compute g succeeds even just on *one input* on which the foregoing formulas correctly compute g.[34]

Proof of Proposition 12.6.16. Let $c_2 > c_1$ be a sufficiently large constant, and let $T'(n) = T(n) \cdot n^{\varepsilon}$. Fix a standard representation of Turing machines by integers such that each machine is associated with infinitely many input lengths, and for $n \in \mathbb{N}$, let M_n be the machine associated with n.

The hard function. We will not need to formally define $F_n^{\mathtt{diag}}$ that was mentioned above, and instead we just directly define $g\colon \{0,1\}^n \to \{0,1\}^{n^{\varepsilon}}$, by the following algorithm. On input $x \in \{0,1\}^n$, we think of each $i \in [n^{\varepsilon}]$ as the index of a gate v_i in the k^{th} level of a balanced formula on n variables of size $T'(n)$, where k is the level that is $\varepsilon \cdot \log(n)$ below the output gate.[35] Since we consider a balanced formula, the subformula rooted at v_i is of size $T'(n)/2^k = T(n)$. We run M_n on all inputs j such that j is the index of a gate in the subtree of v_i, each time for at most $\log(T'(n))^{c_2}$ steps, and correct the outputs of M_n if they are not valid in some fixed canonical way.[36] Then, we define $g(x)_1 = 1 - v_1(x)$ and for $i \in \{2, ..., n^{\varepsilon}\}$ we define $g(x)_i = v_i(x)$.

By the definition of g, the mapping $(x, i) \mapsto g(x)_i$ can be computed in time $\tilde{O}(T(n))$. Now, denote by F_n the formula that is described by the M_n (after correcting invalid outputs in the canonical way above) and for $x \in \{0,1\}^n$ denote by $F_{n,k}(x)$ the values of the gates in the k^{th} level of F_n at

[34] This follows since the "towards-a-contradiction" hypothesis is that the probabilistic formulas succeed on 0.01 of the inputs, and the derandomization hypothesis is that the deterministic simulation succeeds on $\mu > 0.99$ of the inputs.

[35] Recall that we fixed a canonical ordering of the gates in a balanced formula. This canonical ordering induces an ordering on the gates in the k^{th} level, and i is the index of v_i in that ordering.

[36] That is, if $M_n(u) \notin \mathfrak{B}_2$ for an internal node u we let $M_n(u) = \wedge$, and if $M_n(u) \notin [2n] \cup \{\mathsf{T}, \mathsf{F}\}$ for a leaf u then $M_n(u) = 1$, representing x_1.

input x. Then, we have that

$$\oplus_i F_{n,k}(x)_i = 1 - \oplus_i g(x)_i \ . \tag{12.6.3}$$

Analysis. Assume towards a contradiction that there is a family $\{D_n\}$ of c-well-structured probabilistic formulas of size $T_0(n) = o(T'(n)/n^\delta)$ such that for every sufficiently large $n \in \mathbb{N}$, the fraction of inputs $x \in \{0,1\}^n$ such that $\Pr[D_n(x) = g(x)] \geq 2/3$ is at least 0.01. By our assumption, there is a family $\{D'_n\}$ of c_1-well-structured deterministic formulas of size $T_0(n) \cdot n^\delta = o(T'(n))$ such that for every sufficiently large $n \in \mathbb{N}$ there exists $x_n \in \{0,1\}^n$ for which $D'_n(x) = g(x)$. Let $M^{(1)}$ be the uniform machine describing $\{D'_n\}$.

For $T_1(n) = T_0(n) \cdot n^\delta < T'(n)$, consider the following machine $M^{(2)}$. Given $i \in [2T_1(n) - 1]$, if i is an index of a gate in the k^{th} level of a formula of size $T_1(n)$ or in a level beneath it, then $M^{(2)}(i) = M^{(1)}(i)$; and otherwise (if i is the index of a gate in the top $k-1$ levels), then $M^{(2)}(i)$ outputs the parity function. Finally let $M^{(3)}$ be a machine with essentially the same functionality as $M^{(2)}$, the only difference being that $M^{(3)}$ describes a "padded" formula of size $T'(n)$ with only $T_1(n)$ non-trivial gates at its top, and a trivial copying mechanism at its bottom levels (in order for the formula to be of size precisely $T'(n)$).[37]

Note that for all $n \in \mathbb{N}$ and every input in $[2T'(n) - 1]$ it holds that $M^{(3)}$ runs in time at most $\log(T_0(n) \cdot n^\delta)^{c_1} + \log^2(n) \leq \log(T'(n))^{c_2}$, where $c_2 = c_1 + 2$ and the additional term of $\log^2(n)$ accounts for the overheads of $M^{(3)}$ on top of $M^{(1)}$ (*i.e.*, deciding whether the input requires simulating $M^{(1)}$, or outputting the parity function, or implementing a trivial copying mechanism). Thus, $M^{(3)}$ describes a c_2-well-structured formula family $\{D''_n\}$ of size T'. Denoting by $D''_{n,k}(x)$ the values of gates in the k^{th} level of D''_n at input x, we have that

$$D''_n(x) = \oplus_i D''_{n,k}(x)_i = \oplus_i D'_n(x)_i \ . \tag{12.6.4}$$

Now, fix any $n \in \mathbb{N}$ such that $M^{(3)}$ is associated with input length n. Since $M^{(3)}$ describes a c_2-well-structured family and by the definition of F_n (when describing the hard function g above), for any such n we have that $F_n = D''_n$ and $F_{n,k} = D''_{n,k}$. Also, by our assumption, there exists x_n

[37] That is, $M^{(3)}(i)$ behaves like $M^{(2)}(i)$ on the first $T_1(n) - 1$ gates. Then the next $T_1(n)$ gates, which are the leaves of the formula described by $M^{(2)}$ and are denoted by $\mathcal{L}$, are $\vee$ gates. Similarly all other gates except for the leaves are $\vee$ gates. Finally $M^{(3)}$ labels each leaf ℓ by the label that $M^{(2)}$ assigns to the unique node in $\mathcal{L}$ that is an ancestor of ℓ.

such that $D'_n(x_n) = g(x_n)$. Hence, by combining Eqs. (12.6.3) and (12.6.4) we have that

$$\oplus_i g(x_n)_i = 1 - \oplus_i F_{n,k}(x_n)_i = 1 - \oplus_i D''_{n,k}(i) = 1 - \oplus_i D'_n(x)_i = 1 - \oplus_i g(x)_i \ ,$$

a contradiction. ■

We note several shortcomings of Proposition 12.6.16. First, the function g is computable in time $\tilde{O}(T)$ but not necessarily by well-structured formulas of such size. Secondly, our assumption referred to superfast derandomization of search problems, rather than only decision problems. And thirdly, we only deduced that $g(x)$ cannot be printed, whereas in Theorem 12.1.7 and Theorem 12.1.8 we needed a lower bound even on printing a "slightly-corrupted" version of $g(x)$.

12.7 Instantiations of Known PRGs and Code Constructions

In this appendix we state and prove various instantiations of known constructions of PRGs and of codes. The point in stating and proving these is that we instantiate the known constructions with non-standard parameters and with specific efficiency constraints that are useful for our purposes. Specifically, the appendix includes proofs of three claims:

1. We prove Theorem 12.4.10, which is an instantiation of the Nisan-Wigderson PRG in which both the output length and the running time of the reconstruction algorithm are very small, and both the PRG and the reconstruction can be computed by NC circuits that are *logspace-uniform* (assuming that the PRG gets explicit access to the hard function, and that the reconstruction algorithm gets oracle access to a distinguisher).
2. We prove Theorem 12.6.3 from Section 12.6, which combines the foregoing construction with parts of the Impagliazzo-Wigderson [IW97] construction, and asserts that the reconstruction algorithm for the obtained PRG is a uniform learning algorithm (following [IW98]).
3. We prove Theorem 12.4.11, which asserts that the list-decoding algorithm of Goldreich and Levin [GL89] can be implemented by small-depth circuits.

In Section 12.7.1 we prove Theorem 12.4.10. Then, in Section 12.7.2 we state and prove an instantiation of the derandomized direct-product construction of [IW97], where the main point is that the reconstruction algorithm is uniform and efficient. In Section 12.7.3 we combine the two foregoing constructions to prove Theorem 12.6.3. And in Theorem 12.4.11, which can be read independently of the rest of this appendix, we prove Theorem 12.4.11.

12.7.1 The Nisan-Wigderson PRG using Logspace-uniform Circuits

To compute the Nisan-Wigderson PRG by logspace-uniform NC circuits, we will use a logspace algorithm that constructs combinatorial designs and "hard-wires" the designs into an NC circuits. The logspace construction of designs appears in Section 12.7.1, and the corresponding instantiation of the NW PRG appears in Section 12.7.1.

Designs in logspace

The construction of combinatorial designs that we present works in space that is logarithmic in the number of sets. We follow an idea that appeared in [HR03] and was attributed to Salil Vadhan; an earlier construction appeared in [KvM02]. Let us first recall the definition of an explicit standard combinatorial design, and then state and prove the result itself.

Definition 12.7.1 ([NW94])**.** *A family of sets $S_1, \ldots, S_m \subseteq [d]$ is called an (m, ℓ, ρ, d)-design if each of the sets is of size $|S_i| = \ell$, and any two distinct sets S_i, S_j satisfy $|S_i \cap S_j| \leq \log(\rho)$. The computational complexity of the design is the complexity of the function that gets input $i \in [m]$ and outputs the set S_i.*

The proof idea is to create a designs by simulating a "code concatenation" procedure akin to the Justensen code. Loosely speaking, we combine a good and explicit "outer" error-correcting code with an optimal "inner" combinatorial design (that we can find by an exhaustive search). The result of this combination is stated in the following lemma from [HR03]:

Lemma 12.7.2 (from codes to designs; see [HR03, Lemma 5.5])**.** *For $a, \ell_0, \rho_0, d_0, \ell_1, k, m \in \mathbb{N}$, assume that there exist:*

1. *An (a, ℓ_0, ρ_0, d_0)-design computable in space $O(\log(m))$.*
2. *An error-correcting code $C \colon [m] \to [a]^{\ell_1}$ with (absolute) distance $\ell_1 - k$ that is computable in space $O(\log(m))$.*

Then, there exists an (m, ℓ, ρ, d)-design computable in space $O(\log(m))$, where $\ell = \ell_1 \cdot \ell_0$ and $d = \ell_1 \cdot d_0$ and $\log(\rho) = k \cdot \ell_0 + (\ell_1 - k) \cdot \log(\rho_0)$.

We obtain the following logspace-computable designs by combining the Reed-Solomon code with an inner design that we find using exhaustive search.

Lemma 12.7.3 (designs in logspace)**.** *There exists a universal constant $c \geq 1$ such that the following holds. For any constant $\alpha \in (0, 1)$ and sufficiently large integer ℓ there exists an (m, ℓ, ρ, d)-design, where $m \geq 2^{(1/c) \cdot \alpha \cdot \ell}$ and $d \leq c \cdot \ell / \alpha$ and $\log(\rho) = \alpha \cdot \ell$, that is computable in space $O(\log(m))$.*

Proof. Let $\ell_1 = O(\ell/\log(\ell))$ be the minimal power of two such that $\ell' = \ell_1 \cdot \log(\ell_1) \geq \ell$. As the code we take the Reed-Solomon code $\mathbb{F}_{\ell_1}^{k+1} \to \mathbb{F}_{\ell_1}^{\ell_1}$ of degree $k = \lfloor \alpha/8 \rfloor \cdot \ell_1$, whose number of codewords is $m = \ell_1^{k+1} > 2^{(\alpha/8)\cdot\ell'}$. We will combine this code with a design of ℓ_1 sets of size $\ell_0 = \log(\ell_1)$ in a universe of size $d_0 = O(\ell_0/\alpha)$ that pairwise-intersect on $\log(\rho_0) = (\alpha/8) \cdot \ell_0$ coordinates (see, *e.g.*, [Vad12, Problem 3.2] for the existence of such a design).

Using Lemma 12.7.2, the resulting design has $m > 2^{(\alpha/8)\cdot\ell'}$ sets of size ℓ' in a universe of size $O(\ell'/\alpha)$ whose pairwise-intersections are of size at most

$$k \cdot \ell_0 + (\ell_1 - k) \cdot \log(\rho_0) \leq (\alpha/8) \cdot \ell' + (\alpha/8) \cdot \ell' - k \cdot \log(\rho_0) < \alpha/4 \cdot \ell' \,.$$

We truncate each set in the design to have exactly ℓ elements. Note that $\ell' < 4\ell$,[38] and therefore the pairwise-intersections are of size at most $\alpha \cdot \ell$ and the number of sets is $m \geq 2^{(\alpha/32)\cdot\ell}$.

Note that the Reed-Solomon code $\mathbb{F}_{\ell_1}^{\lfloor \alpha/8 \rfloor \cdot \ell_1 + 1} \to \mathbb{F}_{\ell_1}^{\ell_1}$ can be computed in space $O(\ell_1) = O(\log(m))$. Also, we can find an optimal design of ℓ_1 sets of size $\log(\ell_1)$ in a universe of size $O(\log(\ell_1)/\alpha)$ by an exhaustive search, in space $O(\ell_1 \cdot \log(\ell_1)) = O(\log(m))$. By Lemma 12.7.2, the space complexity of the final design is $O(\log(m))$. ■

The NW PRG with uniform reconstruction

We now prove Theorem 12.4.10. As mentioned above, this is an instantiation of the NW PRG in which the output length is small, the reconstruction time is small, and both the PRG and the reconstruction algorithm are computable by logspace-uniform NC circuits (assuming they are given access to the hard function and to a distinguisher, respectively).

Theorem 12.7.4 (the NW PRG with reconstruction as a learning algorithm)**.** *There exists a universal constant $c > 1$, an oracle machine G, and a probabilistic oracle machine R_0, such that the following holds:*

1. **Generator:** *When given input $(1^{\ell_k}, 1^m, \alpha)$ such that $m \leq 2^{(\alpha/c)\cdot\ell_k}$ oracle access to $h\colon \{0,1\}^{\ell_k} \to \{0,1\}$, the machine G runs in time $2^{c\cdot\ell_k/\alpha}$ and outputs a set of strings in $\{0,1\}^m$. Moreover, if α is constant and ℓ_k and m are sufficiently large, then G can be implemented by logspace-uniform oracle circuits of size $2^{c\cdot\ell_k/\alpha}$ and depth $O(\log(m, \ell_k))$.*
2. **Reconstruction:** *When given input $(1^{\ell_k}, 1^m, \alpha)$ and oracle access to a $(1/m)$-distinguisher D for $G^h(1^{\ell_k}, 1^m, \alpha)$ and to h, the machine R_0 runs in time $m^c \cdot 2^{\alpha\cdot\ell_k}$, makes non-adaptive queries, and outputs with probability at least $1 - 2^{-3m}$ an oracle circuit that computes h on $1/2 + m^{-3}$ of the*

[38] This is since $\ell_1 \cdot \log(\ell_1) < 4(\ell_1/2) \cdot \log(\ell_1/2)$ for a sufficiently large $\ell_1 = O(\ell/\log(\ell))$.

inputs when given access to D. The circuit that R_0 outputs has depth $\mathrm{polylog}(m, \ell_k)$ *and makes just one oracle query. Moreover, if α is a constant and ℓ_k and m are sufficiently large, then R_0 can be implemented by a logspace-uniform probabilistic oracle circuit of size $m^c \cdot 2^{\alpha \cdot \ell_k}$ and depth* $\mathrm{polylog}(m, \ell_k)$ *that makes non-adaptive queries.*

Proof. Given input $(1^{\ell_k}, 1^m)$ and access to $h \in \{0,1\}^{2^{\ell_k}}$, and assuming that c in our hypothesis is sufficiently large, the machine G constructs the following combinatorial design: The design consists of m sets $S_1, ..., S_m \subseteq [d]$ of size $|S_i| = \ell_k$ that pairwise-intersect on at most $\alpha \cdot \ell_k$ coordinates in a universe of size $d = O(\ell_k/\alpha)$. The machine then enumerates in parallel over seeds $z \in \{0,1\}^d$, and for every z it queries the oracle m times and outputs the m-bit string $h(z{\restriction}_{S_1}) \circ ... \circ h(z{\restriction}_{S_m})$ (where $z{\restriction}_{S_i}$ is the projection of z to the coordinates specified by S_i).

If α is constant and ℓ_k, m are sufficiently large, then the foregoing procedure can be implemented by a logspace-uniform circuit of size $2^d \cdot \mathrm{poly}(m) \leq 2^{c \cdot \ell_k/\alpha}$ and depth $O(\log(m, \ell_k))$, assuming again that c is sufficiently large. Specifically, to compute the designs by logspace-uniform circuits we use Lemma 12.7.3; the logspace algorithm that constructs the circuit computes the designs in advance and hard-wires them into the circuit. In general (*i.e.*, without assuming that α is a constant), the foregoing procedure can be implemented in time $2^{(c/\alpha) \cdot \ell_k}$, using a standard construction of designs in time $\mathrm{poly}(m)$ (see, *e.g.*, [Vad12, Problem 3.2]).

The machine R_0 gets input $(1^{\ell_k}, 1^m, \alpha)$ and constructs the same design that G constructed. It then repeats the following expriment for $\mathrm{poly}(m)$ attempts in parallel:

1. Randomly choose an index $i \in [m]$, values $z' \in \{0,1\}^{[d] \setminus S_i}$, values $r_{i,...,m} \in \{0,1\}^{m-i+1}$, and a random bit σ. Query h on the $m \cdot 2^{\alpha \cdot \ell_k}$ points corresponding to intersections of S_i with other sets.
2. Create a circuit that gets input $x \in \{0,1\}^{\ell_k}$, combines x and z' to a seed $z \in \{0,1\}^d$ (by placing x in the positions indexed by S_i and using z' to fill the other positions), looks up $r_1, ..., r_{i-1} = h(z{\restriction}_{S_1}) \circ ... \circ h(z{\restriction}_{S_{i-1}})$, creates a string $r = r_1, ..., r_m$, makes one oracle call to the distinguisher D with input r, and outputs $D(r) \oplus \sigma$.
3. Use $\mathrm{poly}(m)$ oracle calls to h to test whether or not the circuit agrees with h on at least $1/2 + m^{-3}$ of the inputs, with confidence $1 - 2^{-4m}$.

After performing the $\mathrm{poly}(m)$ experiments, the machine R_0 checks if one of them succeeded, and if so it outputs the circuit produced by one of the successful experiments. Otherwise, the machine R_0 halts and outputs "fail".

By a standard reconstruction argument following [NW94], in any one of the experiments, with

probability at least $1/O(m)$ the circuit correctly computes h on $1/2 + 1/O(m^2) > 1/2 + m^{-3}$ of the inputs. Thus, the probability that at least one of the $\text{poly}(m)$ attempts will succeed is at least $1 - 2^{-3m}$ the machine R_0.

The procedure R_0 runs in time $\text{poly}(m) \cdot 2^{\alpha \cdot \ell_k}$ and makes non-adaptive oracle queries. The circuit that R_0 prints has depth $\text{polylog}(m, \ell_k)$ and makes just one oracle call to D. Moreover, if α is constant then R_0 can be implemented by a logspace-uniform circuit, since we compute the designs in logspace (as we did for M), the first two steps are computationally very simple, and in the third step we just need to simulate a logspace-uniform circuit (the one we constructed in the second step); the size of this circuit is $\text{poly}(m) \cdot 2^{\alpha \cdot \ell_k}$, and its depth is $\text{polylog}(m, \ell_k)$. ■

12.7.2 The Derandomized Direct Product of IW with Uniform Reconstruction

We now state and prove an instantiation of the derandomized direct-product construction of Impagliazzo and Wigderson [IW97]. The point of the statement and proof below is simply to verify that the reconstruction algorithm for the [IW97] construction is both *uniform* and *very quick*.

Theorem 12.7.5 (the locally list-decodable "derandomized direct-product" code of [IW97])**.** *For every two constants $\delta, \gamma > 0$, let $c = c_{\delta,\gamma} > 1$ be a sufficiently large constant. Then, for every sufficiently large k and $r \leq \log(k)$ and $\eta = 2^{-r/c}$ there exists a mapping of $g \in \{0,1\}^k$ to $h \in \{0,1\}^{\text{poly}(k)}$ that satisfies the following:*

1. **Low complexity overhead:** *There exists an algorithm that gets input $i \in [\text{poly}(k)]$ and outputs the i^{th} entry of h in time $\text{polylog}(k)$ with r oracle queries to g.*
2. **Reconstruction:** *There exists a probabilistic algorithm that gets input 1^k and oracle access to g, runs in time $k^{\gamma/2} \cdot \text{poly}(\log(k), 1/\eta)$, and outputs $O(1/\eta^2)$ oracle circuits such that the following holds. For every function $\widetilde{h}$ that agrees with h on $1/2 + \eta$ of the inputs, with probability at least $1 - \exp(-1/\eta)$ one of the oracle circuits correctly computes g on $1 - \delta$ of the inputs with at most $\text{poly}(\log(k)/\eta)$ queries to $\widetilde{h}$.*

Proof. To specify our construction we need to recall the following construction of a generator G_0 from [IW97] that maps a random seed of length k' (where the requirements on k' will be clarified below) to an output of length $r \cdot \log(k)$. The generator XORs the output of two algorithms, which are applied independently to the same seed:

1. The first algorithm is a randomness-efficient sampler that maps its seed to r samples of $\log(k)$ bits such that any set in $\{0,1\}^{\log(k)}$ of density δ is sampled with accuracy $\delta/2$ and

with confidence $\eta^4/9 = 2^{-\Omega(r)}$.

2. The second algorithm is a nearly-disjoint subsets generator (a-la [NW94]). This algorithm considers a family of subsets $S_1, ..., S_r \subseteq [k']$, each of length $\log(k)$, that pairwise intersect on at most $(\gamma/2) \cdot \log(k)$ coordinates. Given a seed z, the algorithm outputs the r substrings $(z\restriction_{S_i})_{i\in[r]}$ (where $z\restriction_{S_i}$ is the projection of z to the locations specified by S_i).

For a seed length k' that is the maximum among the seed lengths of the two algorithms above, let $h_{\mathsf{IW}}\colon \{0,1\}^{k'} \to \{0,1\}^r$ be the function $h_{\mathsf{IW}}(z) = (g(G_0(z)_1), ..., g(G_0(z)_r))$, where $G_0(z)_i$ is the i^{th} output string of G_0, which is of length $\log(k)$. The main lemma from [IW97] is the following reconstruction algorithm, which transforms a function that agrees with h_{IW} on $\mathrm{poly}(\eta)$ of the inputs to a function that agrees with g on almost all inputs:

Lemma 12.7.6. *There exists a probabilistic oracle machine R_{IW} that, given access to g, produces in time $\tilde{O}(k^{\gamma/2}/\eta^9)$ a deterministic oracle circuit $\widetilde{g}$ such that with probability $1 - \exp(-1/\eta)/2$ the following holds: When given access to a function that agrees with h_{IW} on at least η^4 of the inputs, the circuit $\widetilde{g}$ makes $O(\log(k)/\eta^9)$ queries and correctly computes g on at least $1 - \delta$ of the inputs.*

Proof sketch. We follow the proof of Theorem 15 in [IW97] with parameters $\varepsilon = \eta^4$ and $\rho = \delta/2$ and $q = \frac{\varepsilon}{4\delta/\rho+1} = \eta^4/9 > 2^{-\rho\cdot r/6}$ (our sampler indeed satisfies the hypotheses of the theorem with these parameters[39]) and with $M = k^{\gamma/2}$ (our nearly-disjoint generator satisfies the hypotheses of the theorem with this parameter).

The proof in [IW97] describes a probabilistic algorithm R_{IW} that runs in time $\tilde{O}(k^{\gamma/2})$ and constructs a probabilistic oracle circuit C with one oracle gate that satisfies the following. For every function $\widetilde{h}_{\mathsf{IW}}$ that agrees with h_{IW} on ε of the inputs there exists a set $X \subset \{0,1\}^{\log(k)}$ of density at least $1 - \delta$ such that for every $x \in X$, the expectation (over random coins of R_{IW}) of the probability (over random coins of C) that the circuit with access to $\widetilde{h}_{\mathsf{IW}}$ outputs $h_{\mathsf{IW}}(x)$ is at least $1/2 + q$. We modify R_{IW} so that it chooses fixed random coins for C, in which case for every $x \in X$, with probability at least $1/2 + q$ over coins for R_{IW} the resulting deterministic oracle circuit outputs $h_{\mathsf{IW}}(x)$ when given access to $\widetilde{h}_{\mathsf{IW}}$.

By running R_{IW} for $t' = O(q^{-2} \cdot \log(k) \cdot (1/\eta)) = O(\log(k)/\eta^9)$ times and printing a circuit that outputs the majority decision of the t' circuits, with probability at least $1 - \exp(-1/\eta)/2$ the

[39] In [IW97] the requirement from a sampler is stronger, and non-standard in modern terms: It specifies that any product of δ-dense sets $T_1 \times ... \times T_r \in (\{0,1\}^{\log(k)})^r$ should be sampled approximately well (see [IW97, Definition 5]). However, their proof only uses the property that is considered standard today, which refers to the special case in which there is a single δ-dense set $T \subseteq \{0,1\}^{\log(k)}$ and $T_i = T$ for all $i \in [r]$. Moreover, the confidence parameter of the sampler that we use is much better than what is needed in the [IW97] proof.

printed circuit computes h_{IW} for every $x \in X$, while making at most t' oracle calls to $\widetilde{h}_{\mathsf{IW}}$. □

The mapping of g to h. We are given $g \in \{0,1\}^k$ and wish to map it to $h \in \{0,1\}^{\mathrm{poly}(k)}$. We instantiate the generator G_0 with the following two algorithms. For a sampler, we can use any strongly-explicit expander with constant spectral gap: The seed is thus of length $\log(k) + O(r)$ (specifying a vertex in $[k]$ and r edge indices), and we rely on an appropriate expander Chernoff bound (see, *e.g.*, [AB09, Theorem 21.15]), using the fact that r is larger than a sufficiently large constant, to deduce that any set of density δ is sampled with accuracy $\delta/2$ and with confidence $2^{-r/10}$. For a nearly-disjoint subsets generator we use a standard construction of combinatorial designs in time $\mathrm{poly}(r, \log(k/\gamma))$ with universe size $O(\log(k/\gamma))$ (see, *e.g.*, [Vad12, Problem 3.2]). Combining these constructions yields seed length $k' = \max\{\log(k) + O(r), O(\log(k)/\gamma)\} = O(\log(k))$, and a generator computable in time $\mathrm{poly}(k')$.

Recall that $h_{\mathsf{IW}}(z) = (g(G_0(z)_1), ..., g(G_0(z)_r))$. We define h to be the truth-table of the Hadamard encoding of h_{IW}; that is, h is the truth-table of $h\colon \{0,1\}^{k'+r} \to \{0,1\}$ such that $h(z,\alpha) = \oplus_{i\in[r]}\alpha_i \cdot h_{\mathsf{IW}}(z)_i$. Note that the truth-table of h is of length $2^{k'+r} = \mathrm{poly}(k)$, and that we can compute each entry in this truth-table in time $\mathrm{polylog}(k)$ with r queries to g.

The reconstruction algorithm. Let $\widetilde{h}\colon \{0,1\}^{k'+r} \to \{0,1\}$ be a function that agrees with h on $1/2+\eta$ of the inputs. The algorithm of [GL89] runs in time $\mathrm{poly}(\log(k)/\eta)$, and with probability at least $1 - \exp(-1/\eta)/2$ outputs an oracle circuit $\widetilde{h}_{\mathsf{IW},\mathtt{list}}$ that gets input $(z,i) \in \{0,1\}^{k'} \times [t]$, where $t = O(1/\eta^2)$, and satisfies the following: For at least $\Omega(\eta)$ of the inputs z to h_{IW} there exists $i \in [t]$ such that $\widetilde{h}^{\widetilde{h}}_{\mathsf{IW},\mathtt{list}}(z,i) = h_{\mathsf{IW}}(z)$ (see, *e.g.*, [Gol08, Theorem 7.8][40]). When the algorithm of [GL89] succeeds, by an averaging argument there exists $i \in [t]$ such that the circuit $\widetilde{h}^{(i)}_{\mathsf{IW}}$ defined by $\widetilde{h}^{(i)}_{\mathsf{IW}}(z) = \widetilde{h}_{\mathsf{IW},\mathtt{list}}(z,i)$, when given oracle access to $\widetilde{h}$, agrees with h_{IW} on at least $\Omega(\eta/t) = \Omega(\eta^3) > \eta^4$ of the inputs.

We run the probabilistic oracle machine R_{IW} from Lemma 12.7.6, while answering its queries using our oracle to g, and this machine outputs an oracle circuit $\widetilde{g}$. We output the t oracle circuits obtained by composing $\widetilde{g}$ with each of the circuits $\widetilde{h}^{(i)}_{\mathsf{IW}}$, for $i \in [t]$. The running time of our algorithm is $\tilde{O}(k^{\gamma/2}) \cdot \mathrm{poly}(1/\eta)$, and each of the t circuits makes at most $\mathrm{poly}(\log(k)/\eta)$ queries to $\widetilde{h}$. With probability at least $1 - \exp(-1/\eta)$, both algorithms (of [GL89] and of [IW97]) succeeded, in

[40] The algorithm of [GL89] is probabilistic, and for every z in a set Z of density $\Omega(\eta)$, with probability at least $1 - \exp(-1/\eta^2)$ there exists i such that the i^{th} potential output of the algorithm equals the correct result. By choosing randomness and fixing it into this probabilistic algorithm, with probability at least $1 - \exp(-1/\eta)/2$ we obtain a circuit that succeeds (in the sense above) on at least half of the inputs in Z.

which case one of the t circuits computes g on $1-\delta$ of the inputs with oracle access to $\widetilde{h}$. ■

12.7.3 The Combined Construction

We now state Theorem 12.6.3 and prove it. The proof amounts to a straightforward combination of Theorem 12.7.4 and Theorem 12.7.5.

Theorem 12.7.7 (the PRG of [NW94, IW97] with reconstruction as a high-accuracy learning algorithm)**.** *For every two constants $\delta, \gamma > 0$ there exist an oracle machine G and a probabilistic oracle machine R such that for every function $g\colon \{0,1\}^{\log(k)} \to \{0,1\}$ and sufficiently small $\varepsilon = \varepsilon_{\delta,\gamma} > 0$ the following holds.*

- **Generator:** *When given input $(1^k, \varepsilon)$ and oracle access to g, the machine G runs in time $\mathrm{poly}(k)$ and outputs a set of strings in $\{0,1\}^m$, where $m = k^\varepsilon$.*
- **Reconstruction:** *When given input $(1^k, \varepsilon)$ and oracle access to a $(1/m)$-distinguisher D for $G^g(1^k, \varepsilon)$ and to g, the machine R runs in time $O(k^\gamma)$ and with probability at least $1 - 2^{-2m}$ outputs an oracle circuit that agrees with g on $1-\delta$ of the inputs when given access to D.*

Proof. We instantiate Theorem 12.7.5 with parameters γ and $\delta/2$. Let $r = 3c \cdot \log(m)$, where $c = c_{\delta/2,\gamma} > 1$ is the constant from Theorem 12.7.5, and note that $r \leq \log(k)$ by our assumption that ε is sufficiently small, and that $\eta = 2^{-r/c} = 1/m^3$. Let $h\colon \{0,1\}^{\ell_k} \to \{0,1\}$ be the corresponding function from Theorem 12.7.5, where $\ell_k = O(\log(k))$. The machine G is the one from Theorem 12.7.4, instantiated with h and with the parameter m. Note that the running time of G is polynomial in $2^{\ell_k} = \mathrm{poly}(k)$.

The machine R gets oracle access to a distinguisher D and to g. It first runs the machine R_0 from Theorem 12.7.4, obtaining an oracle circuit C'_0. (Note that R_0 requires oracle access to h, and R only has oracle access to g, but R can answer each query to h using $r \leq \log(k)$ queries to g.) Then, R runs the reconstruction algorithm from Theorem 12.7.5, which yields $t = O(1/\eta^2) = O(m^6)$ oracle circuits $C'_1, \ldots, C'_t$. It replaces the oracle gates in each C'_i with the circuit C'_0 (which contains an oracle gate to D), and outputs the t resulting circuits $C_1, \ldots, C_t$.

By Theorem 12.7.4, with probability at least $1 - 2^{-3m}$ the function $(C'_0)^D$ agrees with h on $1/2 + m^{-3} = 1/2 + \eta$ of the inputs. Conditioned on this event, by Theorem 12.7.5 with probability at least $1 - \exp(-1/\eta) > 1 - 2^{-3m}$ there exists $i \in [t]$ such that C_i^D agrees with g on $1 - \delta/2$ of the inputs. For $i \in [t]$, we estimate the fraction of inputs on which C_i^D agrees with g, up to error $\delta/2$ and with confidence $1 - 2^{-3m}/t$ (by randomly sampling $O(m)$ inputs, making oracle calls to

g, and evaluating C_i with oracle calls to D). With probability at least $1 - 2^{-3m}$, after this step we find a single C_i such that C_i^D agrees with g on at least $1 - \delta$ of the inputs. The running time of R is $\mathrm{poly}(m) \cdot k^{\gamma/2} < k^\gamma$, and its error probability is at most $3 \cdot 2^{-3m} < 2^{-2m}$. ■

12.7.4 List-decoding the Hadamard Code by Small-depth Circuits

We now state and prove Theorem 12.4.11, which asserts that the list-decoding algorithm of Goldreich and Levin [GL89] can be implemented by small-depth circuits. The proof amounts to verifying that the standard construction of [GL89] satisfies this property.

Theorem 12.7.8 (list-decoding the Hadamard code [GL89]; Theorem 12.4.11, restated)**.** *For any time-computable $a\colon \mathbb{N} \to \mathbb{N}$ satisfying $a(\ell_0) \leq \ell_0$ and $\varepsilon\colon \mathbb{N} \to (0, 1/2)$ there exists a transformation* $\mathtt{Had}$ *that maps any function $g\colon \{0,1\}^{\ell_0} \to \{0,1\}^{a(\ell_0)}$ to a Boolean function $\mathtt{Had}(g)\colon \{0,1\}^{\ell_0 + a(\ell_0)} \to \{0,1\}$ such that the following holds.*

1. **Encoding:** *For every $x \in \{0,1\}^{\ell_0}$ and $z \in \{0,1\}^{a(\ell_0)}$ it holds that $\mathtt{Had}(g)(x,z) = \langle g(x), z\rangle = \oplus_{i \in [a(\ell_0)]} g(x)_i \cdot z_i$.*
2. **Decoding:** *There exists a logspace-uniform circuit* $\mathtt{GL}$ *of size* $\mathrm{poly}(\ell_0/\varepsilon)$ *and depth* $\mathrm{polylog}(\ell_0/\varepsilon)$ *that gets input 1^{ℓ_0} and outputs a probabilistic oracle circuit C of depth* $\mathrm{polylog}(\ell_0/\varepsilon)$ *that satisfies the following. For every oracle $\widetilde{\mathtt{Had}}(g)$ that agrees with $\mathtt{Had}(g)$ on $1/2 + \varepsilon$ of the inputs, the probability over the random coins of C and a choice of $x \in \{0,1\}^{\ell_0}$ that $C^{\widetilde{\mathtt{Had}}(g)}(x) = g(x)$ is at least* $\mathrm{poly}(\varepsilon)$.

Proof. For convenience we denote $n = \ell_0$ and $H = \widetilde{\mathtt{Had}}(g)$. For $k = O(\log(n/\varepsilon))$, the probabilistic oracle circuit C circuit gets input $x \in \{0,1\}^n$ and chooses at random k bits $w_1, \ldots, w_k \in \{0,1\}$ and k vectors $s^1, \ldots, s^k \in \{0,1\}^{a(n)}$. For each $i \in [a(n)]$ in parallel, the circuit C computes $y_i = \mathtt{MAJ}\{b_{i,S}\}_{S \subseteq [k]}$, where $b_{i,S} = H(x, \sum_{j \in S} s^j + e_i) \oplus \sum_{j \in S} w_j$ and $e_i \in \{0,1\}^{a(n)}$ is the indicator vector of the i^{th} position. The circuit outputs the n-bit string $y_1, \ldots, y_n$. A standard analysis shows that the success probability of the resulting circuit is at least $\mathrm{poly}(\varepsilon)$ (see, *e.g.*, [AB09, Proof of Theorem 9.12]). The depth of C is $\mathrm{polylog}(n, 1/\varepsilon)$, since for each output bit C only requires a constant number of iterated addition operations on $\mathrm{poly}(n/\varepsilon)$ vectors of n bits. ■

12.8 Algorithms in Logspace-uniform NC for Polynomial Problems

Our goal in this appendix is to prove that low-degree polynomials are sample-aided worst-case to rare-case reducible by logspace-uniform NC circuits. To prove this we follow the known algo-

rithms underlying such a result and show that each of these algorithms can be implemented by logspace-uniform NC circuits.

The main idea in the argument (which appears in Section 12.8.2) was suggested to us by Madhu Sudan [Sud21], to whom we are very grateful. In high-level, we first show how to extract linear factors from bivariate polynomials, then use this to obtain a list-decoder for the Reed-Solomon (RS) code, then deduce a local list-decoder for the Reed-Muller (RM) code, and finally obtain the worst-case to rare-case reduction. In more detail:

1. First, in Section 12.8.1, we recall some standard results asserting that logspace-uniform NC circuits can solve basic arithmetic and linear-algebraic problems.
2. In Section 12.8.2 we show how, given a bivariate polynomial $p \in \mathbb{F}[x, y]$, we can find all of its linear factors of the form $y - p(x)$ in logspace-uniform NC. This is the crux of the argument, which was suggested by Madhu Sudan.
3. In Section 12.8.3 we plug the foregoing factoring result into Sudan's [Sud97] list-decoder for the RS, to show that it can be implemented in logspace-uniform NC; and deduce a local list-decoder for the RM code in logspace-uniform NC, relying on the reduction by Sudan, Trevisan and Vadhan [STV01] of this problem to the problem of list-decoding the RS code.
4. Finally, in Section 12.8.4 we use the foregoing list-decoder for the Reed-Muller code with an additional simple argument to obtain a sample-aided worst-case to rare-case reduction for low-degree polynomials that is computable in logspace-uniform NC.

Throughout the appendix, we assume that a bivariate polynomial $Q \in \mathbb{F}[x, y]$ with degree D is given as input by listing the coefficients of all of its $\binom{D+2}{2}$ monomials in the lexicographical order.[41] Similarly, a univariate polynomial $f \in \mathbb{F}[x]$ is also given as a list of coefficients, from the lowest power to the highest power.

12.8.1 Arithmetic and Linear Algebra in Logspace-uniform NC

Logspace-uniform NC circuits (and even more restricted circuit classes such as logtime-uniform TC^0) can perform basic arithmetic operations, and solve standard linear-algebraic problems. We now recall several of these results that we will use in our proofs:

Lemma 12.8.1 (arithmetic in logspace-uniform NC)**.** *There exist logspace-uniform* NC *circuit families for the following problems over a prime field* $\mathbb{F}$*:*

[41] In more detail, the monomials are ordered first by their x-degrees, and then by their y-degrees.

- **Iterated addition.** *The input is a list of n field elements and the output is their sum. In this case the circuit is of linear size and of depth* $\log(n) \cdot \log(|\mathbb{F}|)$.
- **Iterated multiplication.** *The input is a list of n field elements and the output is their multiplication. In this case the circuit is of depth* $\log(n \cdot p)$.
- **Iterated addition of polynomials.** *The input is a list of univariate or bivariate polynomials over* $\mathbb{F}$ *with degree bound D, and the output is the polynomial that computes their sum.*
- **Multiplication of polynomials.** *The input is two univariate or bivariate polynomials over* $\mathbb{F}$ *with degree bound D, and the output is the polynomial that computes their multiplication.*

Proof. The circuit for iterated addition is just a tree that iteratively adds pairs of integers. Given n integers that are each represented by $\log(p)$ bits, the tree consists of $\log(n)$ levels, and in each level $i = 1, ..., \log(n)$ we add $n/2^{i+1}$ pairs of elements in size $(n/2^{i+1}) \cdot (\log(p))$ and depth $\log(p)$. The overall size is thus $O(n \cdot \log(p))$ and the overall depth is $\log(n) \cdot \log(p)$.

For iterated multiplication, Hesse, Allender and Barrington [HAB02] showed a construction in logtime-uniform TC^0. Their construction is stronger, since any logtime-uniform TC^0 circuit yields a logspace-uniform NC^1 circuit, by replacing each linear threshold gate by a tree for addition and a top "greater-than" gadget.

Addition of two polynomials over $\mathbb{F}$ that are given as lists of coefficients reduces in logspace-uniform NC to addition of pairs of elements over $\mathbb{F}$. Additional of n polynomials over $\mathbb{F}$ reduces to addition of two polynomials over $\mathbb{F}$, via an addition tree as above (note that the degree bound remains identical throughout the procedure).

For multiplication of degree-D univariates we can again use the logtime-uniform TC^0 circuits of [HAB02]. In case the polynomials are bivariate, we construct a circuit that maps the variable y to x^{2D}, multiplies the resulting two univariates, and transform them back to bivariates. ■

Systems of linear equations over a prime field $\mathbb{F}$ can be solved by logspace-uniform randomized NC circuits. As a first ingredient we verify that the determinant of a matrix can be computed in logspace-uniform NC; then we show that systems with unique solutions can be solved in logspace-uniform NC; and finally we reduce the problem of solving general linear systems to the problem of solving systems with unique solutions. The reduction was shown by Borodin, von zur Gathen, and Hopcroft [BvzGH82], while the unique solution case was established by Csanky [Csa76].

Lemma 12.8.2 (computing the determinant in logspace-uniform NC)**.** *There exists a logspace-uniform family of* NC *circuits that gets as input a matrix and computes its determinant.*

Proof. As shown in [Csa76], the determinant of a matrix can be computed in NC. Our goal here is to further verify that the NC algorithm for determinant in [Csa76] can indeed be implemented by log-space uniform NC. To do so we closely follow the presentation in [Koz92, Lecture 31].

First, relying on Lemma 12.8.1, matrix multiplication is in logspace-uniform NC; and using repeated squaring, matrix powering to a polynomial power can also be computed in logspace-uniform NC. We begin by establishing a simple logspace-uniform NC circuit for computing the inverse of an invertible lower triangular matrix A (that is, $A_{i,j} = 0$ for $i < j$). Without loss of generality assume A is of size $2^\ell \times 2^\ell$ for $\ell \in \mathbb{N}$, we write $A = \begin{bmatrix} B & 0 \\ C & D \end{bmatrix}$, where B, C and D are matrices of size $2^{\ell-1} \times 2^{\ell-1}$ and B and D are lower triangular. One can verify that $A^{-1} = \begin{bmatrix} B^{-1} & 0 \\ -D^{-1}CB^{-1} & D^{-1} \end{bmatrix}$. Hence, one can first compute B^{-1} and D^{-1} by recursion, and then compute A^{-1} by the aforementioned formula. This recursive algorithm can be straightforwardly implemented in logspace-uniform NC, as it has ℓ levels (recall that A is of size $2^\ell \times 2^\ell$ and ℓ is logarithmic in terms of input size) and each level can be implemented by logspace-uniform NC.

Next, given a general matrix A of size $n \times n$, the algorithm defines a vector $s = (s_1, s_2, \ldots, s_n) \in \mathbb{F}^n$ as follows: Denoting $s_0 = 1$, for every $k \in [n]$ we define[42]

$$s_k = \frac{1}{k} \cdot \left(\sum_{j=1}^{k} s_{k-j} \cdot \mathrm{tr}(A^j) \cdot (-1)^{j-1} \right)$$

In other words, plugging $s_0 = 1$, for each $k \in [n]$, we have the following linear equation

$$s_k - \frac{1}{k} \cdot \left(\sum_{j=1}^{k-1} s_{k-j} \cdot \mathrm{tr}(A^j) \cdot (-1)^{j-1} \right) = \mathrm{tr}(A^k) \cdot (-1)^{k-1}.$$

As proved in [Csa76], we have that $s_n = \det(A)$. Thus, our goal is to output s_n. To do so, note that there is a invertible lower triangular matrix $M \in \mathbb{F}^{n \times n}$ and a vector $c \in \mathbb{F}^n$ such that $Ms = c$;

[42] Recall that for an $n \times n$ matrix M, $\mathrm{tr}(M)$ is defined as $\sum_{i=1}^{n} M_{i,i}$.

in more detail, we define

$$M_{k,k} = 1 \text{ and } M_{k,k-j} = \frac{1}{k} \cdot (-1)^j \cdot \mathrm{tr}(A^j) \text{ for } j \in [k-1]$$
$$c_k = \mathrm{tr}(A^k) \cdot (-1)^{k-1} .$$

By the above discussion both M and c can be computed by logspace-uniform NC circuits (since $|\mathbb{F}| > n$ and $1/k$ is defined as $k \neq 0$ for every $k \in [n]$). Also, we can solve the linear system $Ms = c$ by computing $s = M^{-1}c$, using the established logspace-uniform NC circuit for inverting lower triangular matrices. We output s_n. ■

Lemma 12.8.3 (solving linear systems with unique solution in logspace-uniform NC)**.** *There exists a logspace-uniform family of* NC *circuits that gets as input a linear system with n variables and n equations over a prime field $\mathbb{F}$ such that $|\mathbb{F}| > n$ and the linear system is guaranteed to have a unique solution, and outputs the unique solution.*

Proof. Denoting the linear system by $Ax = b$, and recalling that it has a unique solution (*i.e.*, that A has full rank), we know that the solution is $x = A^{-1}b$. By Cramer's rule, matrix inversion reduces to computing the determinants of its minors (as well as the determinant of the matrix itself), and this reduction can indeed be computed in logspace-uniform NC (since each output element is the determinant of a minor of a predetermined submatrix, divided by $\pm$ of the determinant of the matrix itself). The claim follows by Lemma 12.8.2. ■

Lemma 12.8.4 (solving general linear systems in logspace-uniform randomized NC)**.** *There exists a logspace-uniform family of randomized* NC *circuits that get as input a linear system over a prime field $\mathbb{F}$, and with probability at least $1 - 2^{-|\mathbb{F}|}$ output a solution if one exists.*

Proof. The proof of [BvzGH82, Theorem 5] reduces solving general linear systems $Ax = b$ where $A = n \times n$ to solving systems with unique solutions. Their reduction works as follows:

1. Find a basis for the column-space of A among the columns, and find a basis for the row-space of A among the rows.
2. Solve the linear system corresponding to the submatrix of bases, which has a unique solution $v \in \mathbb{F}^{\mathtt{rank}(A)}$.
3. Extend v to a vector $y \in \mathbb{F}^n$ by placing zeroes in the yet-unspecified $n - \mathtt{rank}(A)$ locations. If $Ay = b$, output y, otherwise output $\perp$.

By Lemma 12.8.3 and Lemma 12.8.1 we can perform the second and third steps (respectively) in logspace-uniform NC. It remains to verify that the first step is in logspace-uniform probabilistic NC.

To find a basis among a set of vectors $\{v_1, ..., v_n\}$ for their span, it suffices to include v_i iff $\texttt{rank}(\{v_1, ..., v_{i-1}\}) < \texttt{rank}(\{v_1, ..., v_i\})$ (observe that this reduction from finding a basis to computing the rank is in logspace-uniform NC). Following the proof of [BvzGH82, Theorem 3], we can probabilistically compute the rank of an $m \times m$ matrix A' by taking the maximum among the outputs of $O(1)$ parallel repetitions of the following experiment:

1. Choose two random $m \times m$ matrices B, C.
2. For all $i \in [m]$, compute the determinant of the principal $i \times i$ minor of $BA'C$.
3. Output $\max\{i \in [n] : f_i \neq 0\}$, or 0 if no such i exists.

Relying on Lemma 12.8.2, the entire procedure above can be carried out by logspace-uniform probabilistic NC circuits. ■

12.8.2 Factoring Linear Factors from Bivariates

Our goal in this section is to construct a logspace-uniform NC circuit that gets as input a bivariate polynomial $Q \in \mathbb{F}[x, y]$ and finds all of its factors that are of the form $y - p(x)$ for some univariate polynomial $p(x)$. Specifically, we prove the following:

Theorem 12.8.5 (factoring linear factors from bivariate polynomials in logspace-uniform $\mathcal{NC}$)**.** *Let $\mathbb{F}$ be a prime field and let $D \geq 1$ be an integer such that $|\mathbb{F}| > 2D^2$. Then, there exists an algorithm $\mathsf{factorize}_{D,\mathbb{F}}$ that gets as input $Q \in \mathbb{F}[x, y]$ with degree at most D and advice $(r, a, t) \in [D] \times \mathbb{F}^2$ and satisfies the following:*

1. *For every $\tau(x) \in \mathbb{F}[x]$ such that $(y - \tau(x))$ is a factor of Q, there exists $(r, a, t) \in [D] \times \mathbb{F}^2$ such that $\mathsf{factorize}_{D,\mathbb{F}}(Q, r, a, t) = \tau(x)$.*
2. *The algorithm $\mathsf{factorize}_{D,\mathbb{F}}$ can be implemented by logspace-uniform NC circuits.*

High-level description of the algorithm

The idea behind the above theorem is to use the Hensel lifting technique, adapted to our particular setting. Loosely speaking, given a bivariate polynomial $Q(x, y)$ and a factor $(y - t)$ of $Q(0, y) \in \mathbb{F}[y]$, one can "lift" $(y - t)$ to a factor $(y - \tau(x))$ of $Q(x, y) \in \mathbb{F}[x, y]$, under the condition that $y - t$ has multiplicity exactly 1 in $Q(0, y)$, and $t = \tau(0)$ (see Proposition 12.8.11). While we do not know

$t = \tau(0)$ (as we do not know τ in advance), our circuit simply tries all possible $t \in \mathbb{F}$ in parallel, as we are allowed to use size poly$(|\mathbb{F}|)$.

To deal with the requirement that $y - t$ has multiplicity 1 in $Q(0, y)$, we atke the following preprocessing steps. Suppose that Q has a factor $(y - \tau(x))$ with multiplicity r. We first observe that $y - \tau(x)$ would be a factor of $\frac{\partial^{r-1} Q}{\partial^{r-1} y}$ with multiplicity 1 (see Lemma 12.8.8). So we enumerate all possible multiplicities r at the beginning, and deal with all $Q^{(r-1)} = \frac{\partial^{r-1} Q}{\partial^{r-1} y}$ (in parallel) instead of Q. But this does not guarantee that $y - \tau(0)$ has multiplicity 1 in $Q(0, y)$.[43] We further consider all "shifted versions" of $Q^{(r-1)}$, defined by $Q_a^{(r-1)} = Q^{(r-1)}(x + a, y)$. Note that factor $y - \tau(x)$ of $Q^{(r-1)}$ is in one-to-one correspondence to factor $y - \tau(x - a)$ of $Q_a^{(r-1)}$. We show that there must exist a shift a such that $y - \tau(x - a)$ has multiplicity 1 in $Q_a^{(r-1)}(0, y)$ (see Lemma 12.8.9). Hence, to ensure the multiplicity condition, we enumerate all possible powers r and all possible shifts a, and then apply the Hensel lifting to every derived polynomials $Q_a^{(r-1)}$ (see Algorithm 12.3).

Preliminaries

In this section, when we refer to rings we always mean commutative rings. For a polynomial $Q \in \mathbb{F}[x, y]$ such that $Q = \sum_{a,b} c_{a,b} x^a y^b$, we use $\deg(Q)$, $\deg_x(Q)$ and $\deg_y(Q)$ to denote the degree of Q, the x-degree of Q and the y-degree of Q, respectively; that is, $\deg(Q)$ is defined as $\max\{a + b \mid c_{a,b} \neq 0\}$, and $\deg_x(Q)$ (resp. $\deg_y(Q)$) is defined as $\max\{a \mid c_{a,b} \neq 0\}$ (resp. $\max\{b \mid c_{a,b} \neq 0\}$).

For a ring R and an ideal $I \subseteq R$, we say that elements $g, h \in R$ are coprime mod I if there exist $a, b \in R$ such that $ag + bh \equiv 1 \pmod{I}$.

Properties of polynomials

Towards presenting the algorithm, we state several elementary facts about polynomials.

Lemma 12.8.6 (division by linear factors)**.** *Let R be a unique factorization domain, let $f \in R[y]$, and let $t \in R$. Then, there exists a unique $h \in R[y]$ such that*

$$f(y) - f(t) = (y - t)h(y).$$

We also need the following algorithmic version of a special case of Lemma 12.8.6, in which $R = \mathbb{F}[x]$.

[43] For example, it could be the case that Q has another factor $y - \tilde{\tau}(x)$ such that $\tau(x) \neq \tilde{\tau}(x)$ but $\tilde{\tau}(0) = \tau(0)$. In this case $y - \tau(0)$ would have multiplicity at least 2 in $Q(0, y)$.

Lemma 12.8.7 (algorithmic division by linear factors)**.** *Let $\mathbb{F}$ be a prime field, and let $D \geq 1$ be an integer. There is an algorithm* $\mathsf{SimpleDivision}_{D,\mathbb{F}}$ *that gets as input $f \in \mathbb{F}[x,y]$ with degree at most D and $\tau \in \mathbb{F}[x]$, and outputs $h \in \mathbb{F}[x,y]$ such that*

$$f(x,y) = (y - \tau(x)) \cdot h(x,y) + f(x,\tau(x)).$$

Moreover, $\mathsf{SimpleDivision}_{D,\mathbb{F}}$ *can be implemented by logspace-uniform* NC *circuits.*

Proof. First, by Lemma 12.8.6 (with the same f and $R = \mathbb{F}[x]$ and $t = \tau(x)$), we know that there exists a unique h satisfying our requirement. We get as input

$$f(x,y) = \sum_{a,b:\ a+b \leq D} c_{a,b} x^a y^b.$$

Letting $\gamma(x) = f(x,\tau(x)) = \sum_{a,b:\ a+b\leq D} c_{a,b} x^a \tau(x)^b$, note that we can compute a representation of γ as a list of monomials by logspace-uniform NC circuits. In more detail, we can use polynomial powering to compute $c_{a,b} x^a \tau(x)^b$ (multiplying by $c_{a,b} x^a$ is simple), and then sum up all the resulting $O(D^2)$ polynomials by polynomial addition, using Lemma 12.8.1.

Now we have

$$f(x,y) - \gamma(x) = (y - \tau(x)) \cdot h(x,y). \tag{12.1}$$

Note that we have $\deg(h) \leq \deg(f) - 1 \leq D - 1$. Examining the requirement (12.1), since we already know f, γ and τ, we can treat their coefficients as known constants, and (12.1) can be equivalently written as $\binom{D+1}{2}$ linear equations, involving coefficients of h as the only variables.

In more detail, denoting $h = \sum_{a+b\leq D-1} h_{a,b} x^a y^b$, the equation implies that

$$f(x,y) - \gamma(x) = (y - \tau(x)) \cdot \sum_{a+b\leq D-1} h_{a,b} x^a y^b = \sum h_{a,b} x^a y^{b+1} - \sum h_{a,b} x^a \tau(x) y^b.$$

After gathering coefficients on the RHS, we are left with a pair of equal polynomials (one on the RHS and one on the LHS), and each coefficient of the polynomial on the RHS is a linear combination of the $h_{a,b}$. Since we already computed representations of f, of γ, and of τ, we treat their coefficients as known constants, and this yields a linear system with the $h_{a,b}$'s as variables and with $\binom{D+1}{2}$ equations.

By the above discussions, (12.1) has a unique solution, and therefore the linear system we just constructed also has a unique solution. We can construct the system in logspace-uniform NC by

adding field elements (using Lemma 12.8.1), and solve the system in logspace-uniform NC (using Lemma 12.8.3). ■

Lemma 12.8.8. *Let $Q \in \mathbb{F}[x,y]$ such that $\deg(Q) < \mathsf{char}(\mathbb{F})$, and let $\tau \in \mathbb{F}[x]$ such that $y - \tau(x)$ is a factor of Q with multiplicity $r \geq 1$. Then, $y - \tau(x)$ is a factor of $\frac{\partial^{r-1} Q}{\partial^{r-1} y}$ with multiplicity 1.*

Proof. Let us prove the lemma by induction. Clearly it holds when $r = 1$. For $r \geq 2$, suppose the lemma holds when $r - 1$. Since $y - \tau(x)$ is a factor of Q with multiplicity r, we can write

$$Q(x,y) = (y - \tau(x))^r \cdot H(x,y)$$

such that $H \in \mathbb{F}[x,y]$, and $y - \tau(x)$ does not divide H. By the chain rule of partial derivatives,

$$\frac{\partial Q}{\partial y}(x,y) = \left[(y - \tau(x))^r \cdot \frac{\partial H}{\partial y}(x,y)\right] + \left[r \cdot (y - \tau(x))^{r-1} \cdot H(x,y)\right]. \tag{12.2}$$

Now, observe that $r \neq 0$ (since $\mathsf{char}(\mathbb{F}) > \deg(Q) \geq r$). This means that $(y - \tau(x))^r$ does not divide (12.2), and consequently $y - \tau(x)$ is a factor of $\frac{\partial Q}{\partial y}$ with multiplicity $r - 1$. The lemma then follows from the induction hypothesis. ■

Lemma 12.8.9. *Let $Q \in \mathbb{F}[x,y]$ such that $2\deg(Q)^2 < |\mathbb{F}|$, and let $(y - \tau(x))$ be a factor of Q with multiplicity 1. Then, there exists $a \in \mathbb{F}$ such that $(y - \tau(a))$ is a factor of $Q(a,y) \in \mathbb{F}[y]$ with multiplicity 1.*

Proof. Let $H \in \mathbb{F}[x,y]$ such that

$$Q(x,y) = (y - \tau(x)) \cdot H(x,y).$$

Since $(y - \tau(x))$ has multiplicity 1 in $Q(x,y)$, it means that $(y - \tau(x))$ does not divide $H(x,y)$. By Lemma 12.8.6 with $f = H(x,y) \in (\mathbb{F}[x])[y]$ and $t = \tau(x)$, we have

$$H(x,y) = \tilde{H}(x,y) \cdot (y - \tau(x)) + \gamma(x),$$

for $\gamma(x) = H(x, \tau(x))$. Note that $\gamma(x)$ has degree at most $\deg_x(H) + \deg_y(H) \cdot \deg(\tau) \leq \deg(Q) + \deg(Q)^2 \leq 2\deg(Q)^2$. Also note that $\gamma(x)$ is not a zero polynomial.

For any $a \in \mathbb{F}$, plugging in $x = a$, we have

$$Q(a, y) = (y - \tau(a)) \cdot H(a, y) = (y - \tau(a)) \cdot (\tilde{H}(a, y) \cdot (y - \tau(a)) + \gamma(a)).$$

As long as a satisfies $\gamma(a) \neq 0$, we have that $(y - \tau(a))$ is a factor of $Q(a, y)$ with multiplicity 1. (Also note that $\tilde{H}(a, y) \cdot (y - \tau(a)) + \gamma(a)$ is a non-zero polynomial by examining the highest y-power.) Such an a exists as $\deg(\gamma) \leq 2 \deg(Q)^2 < |\mathbb{F}|$. ■

Hensel lifting

The following lemma capturing the well-known Hensel lifting technique will be crucial for our algorithms.

Lemma 12.8.10. *Let R be a ring, let $I \subseteq R$ be an ideal, and let $f \in R$. Suppose that there are elements $g, h \in R$ such that*

(H1) $f \equiv gh \pmod{I}$,

(H2) *g and h are coprime mod I.*

Then, the following holds:

1. ***Lifting:*** *There exist $\tilde{g}, \tilde{h} \in R$ such that*

   (C1) $f \equiv \tilde{g}\tilde{h} \pmod{I^2}$,

   (C2) *$\tilde{g}$ and $\tilde{h}$ are coprime mod I^2.*

   (C3) $g \equiv \tilde{g} \pmod{I}$ *and* $h \equiv \tilde{h} \pmod{I}$

2. ***Uniqueness:*** *For every elements $\tilde{g}, \tilde{h} \in R$ such that (C1) – (C3) holds, if there are other two elements $g', h' \in R$ satisfying (C1) and (C3), then there exists $u \in I$ such that*

$$g' \equiv \tilde{g}(1 + u) \pmod{I^2}, \quad \text{and} \quad h' \equiv \tilde{h}(1 - u) \pmod{I^2}.$$

For completeness, we give below a standard proof of this lemma, following [Sap17, Theorem 12.1] and [Sud15, Lemma 4].

Proof. Denote $f = gh + q$ for some $q \in I$, and let $a, b \in R$ and $r \in I$ such that $ag + bh = 1 + r$. We define

$$\tilde{g} = g + bq, \quad \tilde{h} = h + aq \quad .$$

Observe that $\tilde{g} = g \pmod{I}$ and $\tilde{h} = h \pmod{I}$, and also that

$$\begin{aligned}\tilde{g}\tilde{h} &= gh + gaq + hbq + baq^2 \\ &= f - q + q(1+r) + baq^2 \\ &= f + qr + baq^2 \\ &= f \pmod{I^2}.\end{aligned}$$

To show that suitable $\tilde{a}, \tilde{b}$ exist, note that for some $s \in I$ it holds that $a\tilde{g} + b\tilde{h} = ag + bh + abq + baq = 1 + r + 2qab = 1 + s$; we define $\tilde{a} = a(1-s)$ and $\tilde{b} = b(1-s)$, and note that $\tilde{a}\tilde{g} + \tilde{b}\tilde{h} = (1-s)(a\tilde{g} + b\tilde{h}) = 1 - s^2 = 1 \pmod{I^2}$.

For the uniqueness part, we will now assume (g_1, h_1, a_1, b_2) satisfy (C1) - (C3), and let (g_2, h_2) be two elements satisfying (C1) and (C3). Since both (g_1, h_1) and (g_2, h_2) satisfy (C3), there exists $\alpha, \beta \in I$ such that $g_2 = g_1 + \alpha$ and $h_2 = h_1 + \beta$. Also, from (C1), we have that $g_1 h_1 \equiv f \equiv g_2 h_2 \pmod{I^2}$. Then, it follows that

$$\begin{aligned}0 \equiv g_2 h_2 - g_1 h_1 &\equiv (g_1 + \alpha)(h_1 + \beta) - g_1 h_1 \pmod{I^2} \\ &\equiv \alpha h_1 + g_1 \beta \pmod{I^2} && (\alpha\beta \in I^2)\end{aligned}$$

which implies that

$$\begin{aligned}& b_1 \alpha h_1 + b_1 g_1 \beta \equiv 0 \pmod{I^2} \\ \implies & \alpha(1 - a_1 g_1) + b_1 g_1 \beta \equiv 0 \pmod{I^2} && \text{(see below)} \\ \implies & \alpha = g_1(a_1 \alpha - b_1 \beta) \pmod{I^2}\end{aligned}$$

where the equality $b_1 h_1 \equiv 1 - a_1 g_1 \pmod{I^2}$ above is since $a_1 g_1 + b_1 h_1 \equiv 1 \pmod{I^2}$.

Let $u = a_1 \alpha - b_1 \beta$, note that since $\alpha, \beta \in I$, we also have that $u \in I$. The above shows that $\alpha = g_1 u \pmod{I^2}$, and since $g_2 = g_1 + \alpha$, it follows that

$$g_2 = g_1(1+u) \pmod{I^2}.$$

By a symmetric argument we have that $\beta = h_1(b_1\beta - a_1\alpha) = -h_1 u \pmod{I^2}$, and since $h_2 = h_1 + \beta$ we have that $h_2 = h_1(1-u)$. ∎

The foregoing Hensel lifting procedure can be concisely described by the following algorithm.

Algorithm 12.1: $\mathsf{HenselLift}_{R,I}(f,g,h,a,b)$

Parameters: A ring R and an ideal I.

Input: Given $f,g,h \in R$ satisfying (H1) and (H2), and $a,b \in R$ such that $ag+bh \equiv 1 \pmod{I}$.

1 $q \leftarrow f - gh$;

2 $\tilde{g} \leftarrow g + bq, \quad \tilde{h} = h + aq$;

3 $s \leftarrow a\tilde{g} + b\tilde{h} - 1$;

4 $\tilde{a} \leftarrow a(1-s), \quad \tilde{b} = b(1-s)$;

Output: A tuple $(\tilde{g},\tilde{h},\tilde{a},\tilde{b})$ satisfying (C1) - (C3) and $\tilde{a}\tilde{g} + \tilde{b}\tilde{h} \equiv 1 \pmod{I^2}$.

Hensel lifting for factoring linear terms from polynomials

We now describe a modification of the generic Hensel lifting procedure that is particularly suited to our setting. Specifically, in our setting the ring R is $\mathbb{F}[x,y]$, and we are looking to factor a given polynomial $Q(x,y)$ such that one factor will be of the form $(y-\tau(x))$ (rather than to factor the polynomial arbitrarily). Moreover, we wish to perform this procedure efficiently, by logspace-uniform circuits of low depth.

Proposition 12.8.11. *Let $\mathbb{F}$ be a prime field and let $D \geq 1$ be an integer. Then, there exists a procedure $\mathsf{PolyLift}_{D,\mathbb{F}}$ that gets as input $Q \in \mathbb{F}[x,y]$ of degree at most D and $t \in \mathbb{F}$, and satisfies the following.*

1. *If Q has a factor $y-\tau(x)$ such that $t = \tau(0)$, and the multiplicity of $y-t$ in $Q(0,y)$ is 1. Then, $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$ outputs $\tau(x)$.*
2. *The procedure $\mathsf{PolyLift}$ can be implemented by logspace-uniform NC circuits.*

We stress that the assumption in Proposition 12.8.11 that the multiplicity of $y-t$ in $Q(0,y)$ is 1 implies that there is only one factor $y-\tau(x)$ satisfying $t=\tau(0)$. The first item in Proposition 12.8.11 asserts that $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$ outputs $\tau(x)$ (*i.e.*, essentially outputs this factor). In the rest of the section we prove Proposition 12.8.11, by first describing $\mathsf{PolyLift}_{D,\mathbb{F}}$ and then analyzing it.

Description of $\mathsf{PolyLift}_{D,\mathbb{F}}$

Condition check. $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$ first checks whether $(y-t)$ is a factor of $Q(0,y)$ with multiplicity 1. It returns $\perp$ immediately if the condition fails to hold. Otherwise, since $(y-t)$ is a factor

of $Q(0,y)$, there is a unique polynomial $h(y) \in \mathbb{F}[y]$ such that

$$Q(0,y) = (y-t) \cdot h(y).$$

High-level description of the main loop. Let K be the smallest integer so that $2^K > D \geq \deg_x(Q)$. The goal of the algorithm $\mathsf{PolyLift}_{D,\mathbb{F}}$ is to iteratively construct, for every $0 \leq k \leq K$, a polynomial $p_k \in \mathbb{F}[x]$ and $q_k, a_k, b_k \in \mathbb{F}[x,y]$ such that

$$Q(x,y) \equiv (y - p_k(x)) \cdot q_k(x,y) \pmod{x^{2^k}}, \tag{12.3}$$

$$a_k(y - p_k(x)) + b_k q_k(x,y) \equiv 1 \pmod{x^{2^k}}. \tag{12.4}$$

$$y - p_k(x) \equiv y - t \pmod{x} \quad \text{and} \quad q_k(x,y) \equiv h(y) \pmod{x}. \tag{12.5}$$

We will also show that, if Q has a factor of the form $(y - \tau(x))$ such that $\tau(0) = t$, then for every k it holds that $\tau(x) \equiv p_k(x) \pmod{x^{2^k}}$. To put it succinctly, we will show that

$$\tau(x) \equiv p_k(x) \pmod{x^{2^k}} \quad \text{if } y - \tau(x) \text{ divides } Q \text{ and } \tau(0) = t. \tag{12.6}$$

The algorithm will finally return $p_K(x)$.

Base step. The goal of the base case is to construct $p_0 \in \mathbb{F}[x]$, $q_0, a_0, b_0 \in \mathbb{F}[x,y]$ so that they satisfy (12.3), (12.4), (12.5) and (12.6) for $k = 0$.

Letting $q_0(x,y) = h(y)$ and $p_0(x) = t$ (they clearly satisfy (12.5) for $k = 0$), we have

$$Q(x,y) \equiv (y - p_0(x)) \cdot q_0(x,y) \pmod{x},$$

which establishes (12.3) for $k = 0$.

Next we show how to construct two polynomials $a_0, b_0 \in \mathbb{F}[y]$ so that

$$a_0(y-t) + b_0 h(y) = 1,$$

thereby establishing (12.4) for $k = 0$. By Lemma 12.8.6, we can write $h(y) = (y-t) \cdot \tilde{h}(y) + \beta$, where $\beta = h(t) \neq 0$. (We know that $(y-t)$ does not divide $h(y)$, since $(y-t)$ has multiplicity 1 in $Q(0,y)$ and $Q(0,y) = (y-t)h(y)$.) Then we can simply set $a_0 = (-\beta^{-1}) \cdot \tilde{h}(y)$ and $b_0 = \beta^{-1}$.

Finally, suppose that Q has a factor $(y - \tau(x))$ such that $\tau(0) = t$. Then clearly $p_0(x) \equiv t \equiv \tau(x) \pmod{x}$, establishing (12.6) for $k = 0$.

Induction. For an integer $1 \leq k \leq K$, suppose that we have $p_{k-1} \in \mathbb{F}[x]$, $q_{k-1}, a_{k-1}, b_{k-1} \in \mathbb{F}[x, y]$ such that (12.3), (12.4), (12.5) and (12.6) hold for $k - 1$. In the following we show how to construct (p_k, q_k, a_k, b_k) so that (12.3), (12.4), (12.5) and (12.6) hold for k. We do so as follows:

1. Construction of $\tilde{g}, \tilde{h}, \tilde{a}, \tilde{b}$. For this part we will simply apply the generic Hensel lifing algorithm HenselLift. More precisely, let $R = \mathbb{F}[x, y]$ and $I = (x^{2^{k-1}})$, we let

$$(\tilde{g}, \tilde{h}, \tilde{a}, \tilde{b}) = \mathsf{HenselLift}_{R,I}(Q, y - p_{k-1}, q_{k-1}, a_{k-1}, b_{k-1}).$$

By Lemma 12.8.10, $\tilde{g}, \tilde{h}, \tilde{a}, \tilde{b}$ have the following properties:

$$Q \equiv \tilde{g}\tilde{h} \pmod{x^{2^k}}, \quad \tilde{g} \equiv y - t \pmod{x}, \quad \text{and} \quad \tilde{h} \equiv h(y) \pmod{x},$$

and

$$\tilde{a}\tilde{g} + \tilde{b}\tilde{h} \equiv 1 \pmod{x^{2^k}}.$$

(Note that Lemma 12.8.10 indeed shows that $\tilde{g} \equiv y - p_{k-1}(x) \pmod{x^{2^{k-1}}}$, which implies $\tilde{g} \equiv y - t \pmod{x}$ together with our induction hypothesis (12.5). Similarly we also have $\tilde{h} \equiv h(y) \pmod{x}$ from $\tilde{h} \equiv q_{k-1}(x) \pmod{x^{2^{k-1}}}$ and our induction hypothesis (12.5).)

Additionally, following the algorithm of $\mathsf{HenselLift}_{R,I}(Q, y - p_{k-1}, q_{k-1}, a_{k-1}, b_{k-1})$, we also set

$$q = Q - (y - p_{k-1})q_{k-1}. \tag{12.7}$$

and we have that

$$\tilde{g} = (y - p_{k-1}) + b_{k-1}q \text{ and } \tilde{h} = q_{k-1} + a_{k-1}q, \tag{12.8}$$

according to $\mathsf{HenselLift}_{R,I}(Q, y - p_{k-1}, q_{k-1}, a_{k-1}, b_{k-1})$.

2. Construction of p_k, q_k, a_k, b_k. Next, we show how to modify $\tilde{g}$ and $\tilde{h}$ to construct p_k and q_k. For a suitable $r \in \mathbb{F}[x, y]$ such that $x^{2^{k-1}} | r$ that will be defined in a moment, we will show that there exists p_k such that

$$\tilde{g}(x, y) \cdot (1 - r) \equiv y - p_k(x) \pmod{x^{2^k}}$$

and we will define $q_k = \tilde{h} \cdot (1 + r)$, which means that

$$\tilde{h}(x, y) \cdot (1 + r) \equiv q_k(x, y) \pmod{x^{2^k}}.$$

The key observation here is that for a suitable r, since $x^{2^{k-1}} | r$, we have that

$$(1 - r) \cdot (1 + r) = 1 - r^2 \equiv 1 \pmod{x^{2^k}},$$

and hence

$$(y - p_k(x)) \cdot q_k(x, y) \equiv \tilde{g}(x, y)\tilde{h}(x, y) \equiv 1 \pmod{x^{2^k}}.$$

We now proceed as follows:

- *Defining r.* Let $\tau(x, y) = (b_{k-1}q)/x^{2^{k-1}}$. (Recall that $x^{2^{k-1}} | q$ from (12.7)) We think of $\tau(x, y)$ as an element in $(\mathbb{F}[x])[y]$. Letting $R = \mathbb{F}[x]$, by Lemma 12.8.7, we can compute $\bar{h}(x, y) \in \mathbb{F}[x, y]$ and $\gamma(x) = \tau(x, p_{k-1}(x))$ in logspace-uniform NC such that

  $$\tau(x, y) = (y - p_{k-1}(x)) \cdot \bar{h}(x, y) + \gamma(x).$$

  Plugging in $b_{k-1}q$, we have

  $$\begin{aligned}(b_{k-1}q)(x, y) &= [(y - p_{k-1}(x))\bar{h}(x, y) + \gamma(x)] \cdot x^{2^{k-1}}. \\ &= [g\bar{h}(x, y) + \gamma(x)] \cdot x^{2^{k-1}}\end{aligned}$$

  Now we set $r = x^{2^{k-1}} \cdot \bar{h}(x, y)$.
- *Verifying that $\tilde{g} \cdot (1 - r)$ is of the form $y - p_k$.* We now verify that

  $$\tilde{g} \cdot (1 - r) \mod x^{2^k}$$

can indeed be written as a polynomial of the form $y - p_k(x)$, as follows:

$$\begin{aligned}
&\tilde{g}(1-r) \\
=&(g + b_{k-1}q)(1-r) && \text{(by (12.8))} \\
=&(g + [g\bar{h}(x,y) + \gamma(x)] \cdot x^{2^{k-1}})(1 - x^{2^{k-1}} \cdot \bar{h}(x,y)) \\
\equiv& g + [g\bar{h}(x,y) + \gamma(x)] \cdot x^{2^{k-1}} - x^{2^{k-1}} \cdot \bar{h}(x,y) \cdot g \pmod{x^{2^k}} \\
\equiv& g + \gamma(x) \cdot x^{2^{k-1}} \pmod{x^{2^k}} \\
\equiv& y - p_{k-1}(x) + \gamma(x) \cdot x^{2^{k-1}} \pmod{x^{2^k}}.
\end{aligned}$$

Hence, we can set

$$p_k(x) = p_{k-1}(x) - \gamma(x) \cdot x^{2^{k-1}} \mod x^{2^k}$$

- *Modifying $\tilde{a}, \tilde{b}$ to a_k, b_k accordingly.* Finally, we set $a_k = \tilde{a} \cdot (1+r) \mod x^{2^k}$ and $b_k = \tilde{b} \cdot (1-r) \mod x^{2^k}$, we have

$$\begin{aligned}
a_k(y - p_k(x)) + b_k q_k(x,y) &\equiv \tilde{a} \cdot (1+r)\tilde{g} \cdot (1-r) + \tilde{b} \cdot (1-r)\tilde{h} \cdot (1+r) \pmod{x^{2^k}} \\
&\equiv (1-r^2) \cdot (\tilde{a}\tilde{g} + \tilde{b}\tilde{h}) \pmod{x^{2^k}} \\
&\equiv 1 \pmod{x^{2^k}}.
\end{aligned}$$

3. Verifying (12.6). Suppose that Q has a factor $(y - \tau(x))$ such that $\tau(0) = t$, and denote $Q(x,y) = (y - \tau(x)) \cdot H(x,y)$ for some $H \in \mathbb{F}[x,y]$. Since (12.6) holds for $k-1$, we have that $y - \tau(x) \equiv y - p_{k-1}(x) \pmod{x^{2^{k-1}}}$.

We first show that $H(x,y) \equiv q_{k-1}(x,y) \pmod{x^{2^{k-1}}}$. To see this, note that our assumption $y - \tau(x) \equiv y - p_{k-1}(x) \pmod{x^{2^{k-1}}}$ implies that

$$\begin{aligned}
Q = (y - \tau(x)) \cdot H(x,y) &\equiv (y - p_{k-1}(x)) \cdot q_{k-1}(x,y) \pmod{x^{2^{k-1}}} \\
&\equiv (y - \tau(x)) \cdot q_{k-1}(x,y) \pmod{x^{2^{k-1}}},
\end{aligned}$$

which further implies that

$$(y - \tau(x)) \cdot (H(x,y) - q_{k-1}(x,y)) \equiv 0 \pmod{x^{2^{k-1}}}.$$

Examining the highest y-power at the left-hand, the above is only possible when $H(x,y) - q_{k-1}(x,y) \equiv$

$0 \pmod{x^{2^{k-1}}}$, which proves our claim.

Now, to prove that $\tau(x) \equiv p_k(x) \pmod{x^{2^k}}$ we instantiate Lemma 12.8.10 with the following parameters:

$$
\begin{aligned}
&R = \mathbb{F}[x, y], && I = (x^{2^{k-1}}), && f = Q,\\
&g = y - p_{k-1}, && h = q_{k-1}, &&\\
&\tilde{g} = y - p_k, && \tilde{h} = q_k, &&\\
&g' = y - \tau(x), && h' = H. &&
\end{aligned}
$$

Note that $(\tilde{g}, \tilde{h})$ satisfy (C1) – (C3) of Lemma 12.8.10 and that (g', h') satisfy (C1) and (C3) of Lemma 12.8.10. It follows that there exists $u \in I$ (that is, $x^{2^{k-1}}$ divides u) such that

$$(y - \tau(x)) \equiv (y - p_k(x)) \cdot (1 + u) \pmod{x^{2^k}}. \tag{12.9}$$

Again, we will examine the highest y-power on both sides to show that $\tau(x) \equiv p_k(x) \pmod{x^{2^k}}$. First, one can see that $\deg_y(u) = 0$, as otherwise the right side of (12.9) would have y-degree greater than 1, but the left side of (12.9) has y-degree exactly 1. Next, looking at the coefficients of y at both sides, we have $1 \equiv 1 + u \pmod{x^{2^k}}$, meaning that $u \equiv 0 \pmod{x^{2^k}}$, and consequently $\tau(x) \equiv p_k(x) \pmod{x^{2^k}}$. This verifies the condition (12.6).

Output. Finally, $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$ returns $p_K(x)$. The whole algorithm can be succinctly descried as follows.

Algorithm 12.2: $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$

Parameters: D is the maximum degree parameter and $\mathbb{F}$ is a prime field.

Input: Given $Q \in \mathbb{F}[x,y]$ with degree at most D and $t \in \mathbb{F}$.

if *$y-t$ does not divide $Q(0,y)$ or $(y-t)^2$ divides $Q(0,y)$* **then**
  return $\perp$
$h(y) \leftarrow Q(0,y)/(y-t)$ // Base step $q_0 \leftarrow h$, $p_0 \leftarrow t$;
$\tilde{h} \leftarrow (h(y)-h(t))/(y-t)$, $\beta \leftarrow h(t)$;
$a_0 \leftarrow -\beta^{-1}\tilde{h}$, $b_0 \leftarrow \beta^{-1}$;
$K = \lceil \log(D+1) \rceil$; // Induction step
for $k \leftarrow 1, \ldots, K$ **do**
  $(\tilde{g},\tilde{h},\tilde{a},\tilde{b}) \leftarrow \mathsf{HenselLift}_{\mathbb{F}[x,y],(x^{2^{k-1}})}(Q, y-p_{k-1}, q_{k-1}, a_{k-1}, b_{k-1})$; // Computing $\tilde{g},\tilde{h},\tilde{a},\tilde{b}$
  $q \leftarrow Q - (y-p_{k-1})q_{k-1}$;
  $\tau \leftarrow b_{k-1}q/x^{2^{k-1}}$; // Start computing p_k, q_k, a_k, b_k
  $\gamma(x) \leftarrow \tau(x, p_{k-1}(x))$, $\bar{h} \leftarrow (\tau-\gamma)/(y-p_{k-1})$;
  $r \leftarrow x^{2^{k-1}}\bar{h}$;
  $p_k \leftarrow p_{k-1} - \gamma \cdot x^{2^{k-1}} \mod x^{2^k}$, $q_k \leftarrow \tilde{h}\cdot(1+r) \mod x^{2^k}$;
  $a_k \leftarrow \tilde{a}\cdot(1+r) \mod x^{2^k}$, $b_k \leftarrow \tilde{b}\cdot(1-r) \mod x^{2^k}$;
return p_K

Analysis of $\mathsf{PolyLift}_{D,\mathbb{F}}$

We now analyze the algorithm $\mathsf{PolyLift}_{D,\mathbb{F}}$ and prove Proposition 12.8.11.

Proof of Proposition 12.8.11. We first prove the correctness of the algorithm (*i.e.*, the first item in the statement) and then argue about its efficiency (*i.e.*, prove the second item in the statement).

Correctness. Let $t = \tau(0)$. Since the multiplicity of $y-t$ in $Q(0,y)$ is 1, the condition check phase of $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$ passes successfully. Next, recall that $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$ sets K so that $2^K > \deg(Q)$. By (12.6) and the assumption, we have that $\tau(x) \equiv p_K(x) \pmod{x^{2^K}}$. Since $2^K > \deg(\tau)$ as well, this means that $\tau(x) = p_K(x)$.

Complexity. According to Algorithm 12.2 (also with its subroutine Algorithm 12.1), $\mathsf{PolyLift}_{D,\mathbb{F}}(Q,t)$ proceeds in $O(\log D)$ iterations, and each iteration requires a constant number of arithmetic operations on polynomials from $\mathbb{F}[x,y]$ with at most $O(D)$ degree. Relying on Lemma 12.8.1, multiplication, addition, subtraction and division by $x^{2^{k-1}}$ can be implemented by logspace-uniform NC circuits. It remains to implement the divisions on line 5 and line 12 of Algorithm 12.2, which can be handled by using Lemma 12.8.7. ■

Final algorithm

Our full algorithm $\mathsf{factorize}_D(Q)$ works as follows:

Algorithm 12.3: $\mathsf{factorize}_{D,\mathbb{F}}(Q,r,a,t)$

Parameters: D is the maximum degree parameter and $\mathbb{F}$ is a prime field. We require that $|\mathbb{F}| > 2D^2$.

Input: Given $Q \in \mathbb{F}[x,y]$ of degree at most D and $(r,a,t) \in [D] \times \mathbb{F}^2$.

1 Compute $Q^{(r-1)} = \frac{\partial^{r-1}Q}{\partial^{r-1}y}$ (where $Q^{(0)} = Q$);

2 Let $Q_a^{(r-1)}(x,y) \leftarrow Q^{(r-1)}(x+a,y)$;

Output:

if $\mathsf{PolyLift}_{D,\mathbb{F}}(Q_a^{(r-1)},t) = \perp$ **then**
  return $\perp$

else
  $\tilde{\tau}(x) \leftarrow \mathsf{PolyLift}_{D,\mathbb{F}}(Q_a^{(r-1)},t)$;
  return $\tilde{\tau}(x-a)$

Theorem 12.8.12 (reminder of Theorem 12.8.5)**.** *Let $\mathbb{F}$ be a prime field and let $D \geq 1$ be an integer such that $|\mathbb{F}| > 2D^2$. Then, there exists an algorithm $\mathsf{factorize}_{D,\mathbb{F}}$ that gets as input $Q \in \mathbb{F}[x,y]$ with degree at most D and advice $(r,a,t) \in [D] \times \mathbb{F}^2$ and satisfies the following:*

1. *For every $\tau(x) \in \mathbb{F}[x]$ such that $(y-\tau(x))$ is a factor of Q, there exists $(r,a,t) \in [D] \times \mathbb{F}^2$ such that $\mathsf{factorize}_{D,\mathbb{F}}(Q,i_{\mathsf{adv}}) = \tau(x)$.*
2. *The algorithm $\mathsf{factorize}_{D,\mathbb{F}}$ can be implemented by logspace-uniform NC circuits.*

Proof. Let $y-\tau(x)$ be a factor of Q with multiplicity r (note that $1 \leq r \leq \deg(Q) \leq D$). By Lemma 12.8.8 and the assumption on $\mathsf{char}(\mathbb{F})$ (recall that $\mathsf{char}(\mathbb{F}) = |\mathbb{F}|$ since $\mathbb{F}$ is a prime field), it holds that $y-\tau(x)$ is a factor of $Q^{(r-1)} = \frac{\partial^{r-1}Q}{\partial^{r-1}y}$ with multiplicity 1.

Now, by Lemma 12.8.9 and the assumption on $|\mathbb{F}|$, there exists $a \in \mathbb{F}$ such that $(y-\tau(a))$ is a factor of $Q^{(r-1)}(a,y)$ with multiplicity 1. Equivalently, we have that $(y-\tau(0+a))$ is a factor

of $Q_a^{(r-1)}(0, y)$ with multiplicity 1. Letting $t = \tau(a)$, by Proposition 12.8.11, it holds that in this case $\mathsf{PolyLift}_{D,\mathbb{F}}(Q_a^{(r-1)}, t)$ returns the polynomial $\tau(x + a)$, and $\mathsf{factorize}_{D,\mathbb{F}}(Q, r, a, t)$ reports $\tau(x)$ as desired.

Note that by Proposition 12.8.11, $\mathsf{PolyLift}_{D,\mathbb{F}}(Q_a^{(r-1)}, t)$ can be computed by logspace-uniform NC circuits. Hence, to show the efficiency of $\mathsf{factorize}_{D,\mathbb{F}}$, it suffices to show that in logspace-uniform NC, one can compute $Q_a^{(r-1)}$ from Q, and $\tilde{\tau}(x - a)$ from $\tilde{\tau}(x)$.

Recall that Q is given as a list of coefficients $\{c_{a,b}\}$ such that

$$Q = \sum_{a,b:\ a+b \le D} c_{a,b} x^a y^b.$$

From the rule of computing partial derivative, we have

$$Q^{(r-1)} = \frac{\partial^{r-1} Q}{\partial^{r-1} y} = \sum_{a,b:\ a+b \le D,\ b \ge r-1} c_{a,b} \prod_{j=0}^{r-1} (b - j) \cdot x^a y^{b-r+1}.$$

Since $\prod_{j=0}^{r-1}(b - j)$ can be computed straightforwardly in $O(\log |\mathbb{F}|)$ space, one can compute the coefficient list of $\frac{\partial^{r-1} Q}{\partial^{r-1} y}$ in logspace-uniform NC.

Next, given the coefficient list of $Q^{(r-1)}(x, y)$, we can compute the coefficient list of $Q_a^{(r-1)}(x, y) = Q^{(r-1)}(x + a, y)$ by expanding out every power $(x + a)^t$ in $Q^{(r-1)}(x + a, y)$ and summing everything up, which can be implemented in logspace-uniform NC by Lemma 12.8.1. Hence one can compute $Q_a^{(r-1)}$ from $Q^{(r-1)}$ by logspace-uniform NC. Similarly, we can also compute $\tilde{\tau}(x - a)$ from $\tilde{\tau}$ by logspace-uniform NC. This concludes the proof. ■

12.8.3 (Local) List-decoding for Polynomial Codes

In this section we utilize the factorization algorithm from Theorem 12.8.5 to construct logspace-uniform list decoders for both the Reed-Solomon (RS) codes and the Reed-Muller (RM) codes.

First, in Section 12.8.3, we plug Theorem 12.8.5 into the standard list decoding algorithm for the RS code [Sud97], to obtain a logspace-uniform NC list-decoding algorithm for the RS code. Next, in Section 12.8.3, we observe the standard reduction from local list-decoding the RM code to list-decoding the RS code yields a logspace-uniform NC local list-decoder for the RM code.

List-decoding RS codes in logspace-uniform NC

The following logspace-uniform NC list-decoding algorithm for the RS code is obtained by using the standard algorithm of [Sud97] with the factorization algorithm from Theorem 12.8.5. We include a proof for completeness, which essentially follows [AB09, Theorem 19.24] while pointing out how the algorithmic steps can be implemented by logspace-uniform randomized NC circuits.

Theorem 12.8.13 ([Sud97, Sud21]). *Let $\mathbb{F}$ be a prime field. There is an algorithm* RS-DecNC$_{\mathbb{F}}$ *that gets as input a set of m pairs $\{(a_i, b_i) \in \mathbb{F}^2\}_{i=1}^m$ and integers $d, t \geq 1$ such that $t > 2\sqrt{dm}$ and $|\mathbb{F}| > 8dm$, and outputs a list of* $\mathrm{poly}(|\mathbb{F}|)$ *polynomials in $\mathbb{F}[x]$ such that the following holds:*

1. *With probability at least $1 - 2^{-|\mathbb{F}|}$, the output list of* RS-DecNC$_{\mathbb{F}}(\{(a_i, b_i)\}_{i=1}^m, d, t)$ *contains every polynomial $\tau \in \mathbb{F}[x]$ of degree at most d satisfying $\left|\{i \in [m] : \tau(a_i) = b_i\}\right| \geq t$.*
2. *The algorithm* RS-DecNC$_{\mathbb{F}}$ *can be computed by logspace-uniform randomized circuits of size* $\mathrm{poly}(|\mathbb{F}|)$ *and depth* $\mathrm{polylog}(|\mathbb{F}|)$.

Proof. We first find a non-zero polynomial $Q \in \mathbb{F}[x, y]$ with $\deg_x(Q) \leq \sqrt{dm}$ and $\deg_y(Q) \leq \sqrt{m/d}$, such that $Q(a_i, b_i) = 0$ for all $i \in [m]$. This condition can be formulated by m linear equations in the $(\sqrt{dm} + 1) \cdot (\sqrt{m/d} + 1) > m$ coefficients of Q. Since this linear system is homogeneous and there are more coefficients than equations, one can find a non-zero solution with probability at least $1 - 2^{-|\mathbb{F}|}$ by solving the linear system in randomized logspace-uniform NC, relying on Lemma 12.8.4. The algorithm then enumerates all $\vec{\mathsf{adv}} \in [2\sqrt{dm}] \times \mathbb{F}^2$, adds $\mathsf{factorize}_{2\sqrt{dm},\mathbb{F}}(Q, \vec{\mathsf{adv}})$ to the list if it is not $\perp$, and finally returns the list.

To show the first condition, let $\tau \in \mathbb{F}[x]$ be a polynomial such that $\left|\{i \in [m] : \tau(a_b) = b_i\}\right| \geq t$, and let $\gamma(x) = Q(x, \tau(x))$. Since $\deg_x(Q) \leq \sqrt{dm}$ and $\deg_y(Q) \leq \sqrt{m/d}$ and $\deg(\tau) \leq d$, we have that $\deg(\gamma) \leq 2\sqrt{dm} < t$. On the other hand, by our assumption on τ, for at least t a_i's we have $\gamma(a_i) = Q(a_i, \tau(a_i)) = Q(a_i, b_i) = 0$, which means γ has at least t roots. Since $\deg(\gamma) < t$ it follows that γ is the zero polynomial, and hence $(y - \tau)$ is a factor of Q.

By Theorem 12.8.5, it follows that there exists $\vec{adv} \in [2\sqrt{dm}] \times \mathbb{F}^2$ such that $\mathsf{factorize}_{2\sqrt{dm},\mathbb{F}}(Q, \vec{\mathsf{adv}})$ returns τ, and therefore τ is contained in the list with probability at least $1 - 2^{-|\mathbb{F}|}$. The second condition follows from Theorem 12.8.5, together with the randomized logspace-uniform NC circuits for solving homogeneous linear systems (from Lemma 12.8.4). ∎

Local List-decoding RM codes in logspace-uniform NC

Next we show that local list-decoding the RM code can be done in logspace-uniform NC. The proof of this result implements the reduction from local list-decoding the RM code to list-decoding the RS code by Sudan, Trevisan and Vadhan [STV01] in logspace-uniform NC.

We start by noting that the standard decoder for the RM code is implementable in logspace-uniform NC, and then argue that the *local* list-decoder for the RM code is implementable with such complexity.

Theorem 12.8.14 (decoding the RM code in logspace-uniform NC)**.** *For every prime field $\mathbb{F}$ and integer $\ell \geq 1$ there is a probabilistic procedure $\mathsf{RM\text{-}DecNC}_{\mathbb{F},\ell}$ that gets as input a degree parameter $d \leq |\mathbb{F}|/2$ and a vector $x \in \mathbb{F}^\ell$, and gets oracle access to a function $f\colon \mathbb{F}^\ell \to \mathbb{F}$, and satisfies the following:*

1. *If f agrees with a degree-d polynomial $P\colon \mathbb{F}^\ell \to \mathbb{F}$ on a 0.9 fraction of the inputs, then*

$$\Pr\left[\mathsf{RM\text{-}DecNC}^f_{\mathbb{F},\ell}(d,x) = P(x)\right] \geq 1 - 2^{-2|\mathbb{F}|}\,.$$

2. *The procedure $\mathsf{RM\text{-}DecNC}_{\mathbb{F},\ell}$ can be implemented by a family of logspace-uniform oracle circuits of size $\mathrm{poly}(|\mathbb{F}|,\ell)$ and depth $\mathrm{polylog}(|\mathbb{F}|)$.*

Proof. Recall that the standard decoder for the RM code uniformly chooses $z \in \mathbb{F}^\ell$, queries its oracle f on the input-set $L_{x,z} = \{x + tz : t \in \mathbb{F}\}$, runs the RS list-decoder on the set of pairs $\{(y, f(y)) : y \in L_{x,z}\}$ to obtain a list of univariate polynomials $Q_1, ..., Q_{\mathrm{poly}(|\mathbb{F}|)}\colon \mathbb{F} \to \mathbb{F}$, evaluates each Q_i on $L_{x,z}$, and outputs $Q(0)$ where Q is the polynomial in the list whose agreement with f on $L_{x,z}$ is maximal.

The foregoing procedure can indeed be performed by logspace-uniform probabilistic circuits of size $\mathrm{poly}(|\mathbb{F}|,\ell)$ and depth $\mathrm{polylog}(|\mathbb{F}|)$, relying on Theorem 12.8.13. The standard analysis (see, *e.g.*, [AB09, Proof of Theorem 19.19]) shows that when given access to f that agrees with some degree-d polynomial P on at least $1 - (1 - d/|\mathbb{F}|)/6 > 0.9$ of inputs, this procedure outputs $P(x)$ with probability at least 0.6. To amplify the success probability we can repeat the process in parallel for $\mathrm{poly}(|\mathbb{F}|)$ times and take the most frequent result. ■

Theorem 12.8.15 (local list-decoding the RM code in logspace-uniform NC)**.** *For every prime field $\mathbb{F}$ and integer $\ell \geq 1$, there is an algorithm $\mathsf{RM\text{-}ListDecNC}_{\mathbb{F},\ell}$ that gets as input a degree parameter $d \geq 1$, a pair $(x_0, y_0) \in \mathbb{F}^\ell \times \mathbb{F}$, and a vector $x \in \mathbb{F}^\ell$, and gets oracle access to a function $f\colon \mathbb{F}^\ell \to \mathbb{F}$, and satisfies the following:*

1. *If f agrees with a degree-d polynomial $P\colon \mathbb{F}^\ell \to \mathbb{F}$ on $10\sqrt{d/|\mathbb{F}|}$ fraction of the inputs, then with probability at least $1/\mathrm{poly}(|\mathbb{F}|)$ over random pairs (x_0, y_0) from $\mathbb{F}^\ell \times \mathbb{F}$,*

$$\Pr[\mathsf{RM\text{-}ListDecNC}^f_{\mathbb{F},\ell}(d, x_0, y_0, x) = P(x)] \geq 1 - 2^{-2|\mathbb{F}|} \quad \textit{for every } x \in \mathbb{F}^\ell. \tag{12.10}$$

2. $\mathsf{RM\text{-}ListDecNC}_{\mathbb{F},\ell}$ *can be implemented by a family of logspace-uniform oracle circuits of size* $\mathrm{poly}(|\mathbb{F}|, \ell)$ *and depth* $\mathrm{polylog}(|\mathbb{F}|, \ell)$.

Proof. Relying on Theorem 12.8.14, it suffices to describe a relaxed decoder such that (12.10) holds for at least 0.9 fraction of $x \in \mathbb{F}^\ell$ instead of all possible x (note that we can assume, without loss of generality, that $d \leq |\mathbb{F}|/2$). The decoder of [STV01] works by first constructing a univariate degree-3 curve $q : \mathbb{F} \to \mathbb{F}^\ell$ that passes through x and x_0 (that is, $q(0) = x$ and $q(r) = x_0$ for a random element $r \in \mathbb{F} \setminus \{0\}$), then running the RS list-decoder on the set $\{(t, f(q(t))) : t \in \mathbb{F}\}$ to obtain a list $g_1, ..., g_{\mathrm{poly}(|\mathbb{F}|)}$ of polynomials, and finally evaluating all g_i's on a given point, comparing the results to see if there is a unique g_i such that $g_i(r) = y_0$, and if so outputs $g_i(0)$ (see [AB09, Theorem 19.26] for a detailed description and proof of correctness).

The RS list-decoder from Theorem 12.8.13 is in logspace-uniform NC, and thus it suffices to show that the curve q can be constructed by log-space uniform oracle circuit of size $\mathrm{poly}(|\mathbb{F}|, \ell)$ and depth $\mathrm{polylog}(|\mathbb{F}|, \ell)$. To do this the decoder fixes two distinct elements $u, v \in \mathbb{F} \setminus \{0, r\}$, draws two uniformly random vectors x_1 and x_2 from $\mathbb{F}^\ell$, and writes a linear system over 4ℓ variables (the coefficients of q) expressing the four conditions $q(0) = x$, $q(r) = x_0$, $q(u) = x_1$, and $q(v) = x_2$. This linear system has a unique solution, and thus by Lemma 12.8.3 it can be solved in logspace-uniform NC. ■

12.8.4 Sample-aided Worst-case to Rare-case Reductions for Polynomials

Proposition 12.8.16 (low-degree polynomials are uniformly sample-aided worst-case to rare-case reducible; Proposition 12.3.10, restated)**.** *Let $q\colon \mathbb{N} \to \mathbb{N}$ be a function mapping integers to primes, let $\ell\colon \mathbb{N} \to \mathbb{N}$ such that $n \geq \ell(n) \cdot \log(q(n))$, and let $d\colon \mathbb{N} \to \mathbb{N}$. Let $f = \{f_n\}_{n\in\mathbb{N}}$ 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-rare-case reducible by logspace-uniform oracle circuits of size* $\mathrm{poly}(q, \ell)$ *and depth* $\mathrm{polylog}(q, \ell)$ *with error* $1 - 2^{-q}$ *and* $\mathrm{poly}(q)$ *samples, where* $\rho = 10\sqrt{d(n)/q(n)}$.

Proof. We construct a logspace-uniform probabilistic oracle circuit M that gets input 1^n, and oracle access to a function $\widetilde{f_n}$ that agrees with f_n on a $\rho(n)$ fraction of the inputs, and also $\mathrm{poly}(1/\rho)$

labeled samples for f_n, and with probability $1 - 2^{-q}$ outputs a circuit $C\colon \mathbb{F}^\ell \to \mathbb{F}$ such that for every $x \in \mathbb{F}^\ell$ it holds that $\Pr_r[C^{\widetilde{f_n}}(x,r) = f_n(x)] \geq 2/3$. Specifically, M works as follows:

1. The first step is to construct in parallel $t = \text{poly}(q)$ logspace-uniform circuits such that each circuit implements the algorithm from Theorem 12.8.15 with an independent uniform choice of (x_0, y_0) (and with degree parameter $d = d(n)$).

   This yields a list of t probabilistic oracle circuits $C_1, ..., C_t$ such that with probability at least $1 - 2^{-2q}$, there exists $i \in [t]$ for which $\Pr[C_i^{\widetilde{f_n}}(x) = f_n(x)] \geq 1 - 2^{-2q}$ for all $x \in \{0,1\}^n$. We call any circuit that satisfies the latter condition good.
2. The second step is to choose random coins for each circuit C_i and hard-wire them, transforming C_i into a deterministic circuit. Specifically, for each C_i we independently try $w = \text{poly}(q)$ different random strings, obtaining a set of w deterministic circuits $C_{i,1}, ..., C_{i,w}$. This yields $t' = t \cdot w = \text{poly}(q)$ deterministic circuits $D_1, ..., D_{t'}$ such that with probability $1 - 2^{-2q}$ there exists a circuit D_i satisfying $\Pr_x[D_i^{\widetilde{f_n}}(x) = f_n(x)] \geq 0.99$. [44]
3. The third step 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_n on 0.95 of the inputs. To do so we iterate over the list in parallel, and for each circuit D_j we estimate the agreement of $D_j^{\widetilde{f_n}}$ with f_n with accuracy 0.01 and confidence $1 - 2^{-2q}$, using $\text{poly}(q)$ random samples.
4. The final step is to compose D_i with the decoder from Theorem 12.8.14 to obtain a probabilistic circuit that correctly computes f at each input with probability at least $1 - 2^{-2q}$.

The success probability of the foregoing procedure is $1 - O(2^{-2q}) > 1 - 2^{-q}$. We now argue that the foregoing procedure M can be implemented as a logspace-uniform probabilistic circuit of size $\text{poly}(q, \ell)$ and depth $\text{polylog}(q, \ell)$; for simplicity, below we just state that circuits are "logspace-uniform", and by this we implicitly mean circuits of such size and depth.

Note that in the first step we just need to compute the description of $t = \text{poly}(q)$ circuits $C_1, ..., C_t$, whose random coins are the ones chosen by our logspace-uniform circuit M. Since each C_i is logspace-uniform, the part of M that computes their descriptions is also logspace-uniform. The second step is computationally very easy given the descriptions of the C_i's. The third step amounts to simulating the circuits whose descriptions were already computed, so this step can indeed be executed by a logspace-uniform circuit. And the fourth step is dominated by the cost of computing the description of a fixed logspace-uniform circuit, and replacing its oracle gates by

[44] This success bound assumes that $\mathbb{F}$ is sufficiently large such that the success probability of a good C_i satisfies $1 - 2^{-2q} > .999$. If $\mathbb{F}$ is too small then we can amplify the success probability of each C_i in the first step by implementing naive error reduction.

the descriptions of the circuit D_i found in the previous step. ■

12.9 An Unconditional Approximate-direct-product Result

We include for completeness a proof of Proposition 12.6.11. As mentioned in Section 12.6, the proof follows from the results of Impagliazzo *et al.* [IJKW10], with one caveat being that we refer to standard direct-product functions, whereas their direct-product function $f^{\times k}(x_1, ..., x_k)$ only takes *distinct* inputs $x_1, ..., x_k$ (*i.e.*, it disallows $x_i = x_j$ for any $i \neq j \in [k]$).

Proposition 12.9.1 (non-strong approximate-direct-product theorem)**.** *There exists a universal constant $c > 1$ such that the following holds. Let $\delta \in (0, 1/2)$, let $\alpha > 0$ be sufficiently small, let $T : \mathbb{N} \to \mathbb{N}$ be time-computable, and let $k(n) = o(n)$. Then, for any $f \notin \mathsf{avg}_{1-\delta}\text{-}\mathsf{BPTIME}[T]//(1/\alpha)$ and any probabilistic algorithm A that runs in time $T(n) - n^c$, the probability over $z \in \{0,1\}^{k \cdot n}$ that A approximately-prints $f^{\times k}(z)$ with error α is at most δ.*

Proof. Given $\delta > 0$, let $\delta' = \delta/12$, let $\gamma < \delta'$ be a sufficiently small constant and let $\alpha < \gamma$ be a sufficiently small constant. We will assume that $k(n)$ is larger than $\delta'/2\gamma$. Assume towards a contradiction that an algorithm A as in the statement exists, and let $Z \subseteq \{0,1\}^{k \cdot n}$ be the set of inputs z such that with probability at least $1 - \alpha$ we have $\Pr_{i \in [k]}[A(z)_i = g(z)_i] \geq 1 - \alpha$. We denote by $A(z, r)$ the decision of A on input z with random coins r.

Choosing a set $S \subseteq [k]$. The first step of our algorithm is to randomly choose a set $S \subseteq [k]$ of size $\delta'/4\gamma < k/2$. Our hope is that there are many inputs z on which A, with high probability over internal choice of random coins, correctly computes *all the coordinates in S* of $g(z)$; in other words, we hope that there are many z's such that A, with high probability over coins, correctly computes f on *all* the inputs $(z_i)_{i \in S}$. For any input z, and for a fixed $i \in [k]$, we denote $\mu_i(z) = \Pr_r [A(z, r)_i \neq g(z)_i]$; similarly, for a fixed $S \subseteq [k]$ we denote $\mu_S(z) = \Pr_r [A(z, r)_S \neq g(z)_S]$. Then, we claim that:

Claim 12.9.2. *With probability more than $1 - \delta'$ over choice of S, there exists a set Z' of size at least $|Z|/2$ such that for every $z \in Z'$ we have that $\mu_S(z) \leq 2\alpha/\gamma$.*

Proof. For every fixed $z = (x_1, ..., x_k) \in Z$ we have that $\mathbb{E}_{i \in [k]} [\mu_i(z)] = \Pr_{i \in [k], r} [A(z, r)_i \neq g(x)_i] \leq$

2α, and hence

$$\Pr_{i\in[k]}[\mu_i(z) \geq 2\alpha/\gamma] \leq \gamma\,. \tag{12.9.1}$$

By Eq. (12.9.1), for every fixed $z \in Z$, when choosing the first element $i_1 \in [k]$ for S, the probability that $\mu_{i_1}(z) < 2\alpha/\gamma$ is at most γ. When choosing the second elements i_2, the probability that $\mu_{i_2}(z) \geq 2\alpha/\gamma$ is at most $\frac{\gamma\cdot k}{k-1}$; by induction, when adding each element i_j, the probability that $\mu_{i_j}(z) \geq 2\alpha/\gamma$ is less than $\frac{\gamma\cdot k}{k-|S|}$. Since k is sufficiently large, we can bound this probability by $\frac{\gamma\cdot k}{k-|S|} < \frac{\gamma\cdot k}{k/2} = 2\gamma$, where we used the fact that $|S| < k/2$. By a union-bound, the probability over S that $\mu_S(z) \geq 2\alpha/\gamma$ is at most $2\gamma|S| < \delta'/2$.

For every fixed $z \in Z$, and for a random choice of S, denote the event that $\mu_S(z) < 2\alpha/\gamma$ by $\mathcal{G}(z)$. The above shows that for every $z \in Z$ we have that $\Pr_S[\neg\mathcal{G}(z)] < \delta'/2$. It follows that $\mathbb{E}_S\left[\Pr_{z\in Z}[\neg\mathcal{G}(z)]\right] < \delta'/2$, and hence the probability over S that $\Pr_{z\in Z}[\neg\mathcal{G}(z)] < 1/2$ is at most δ'. □

Choosing fixed random coins r. As a second step our algorithm will hard-wire random coins r into A, yielding a deterministic algorithm $A_r(z) = A(z,r)$. We claim that with probability at least $1-\delta'$ over choice of random coins r, there exists a set $Z'' \subseteq Z'$ of density $1 - \frac{2\alpha}{\gamma\cdot\delta'} > 1/2$ in Z' such that for every $z \in Z''$ we have that $A_r(z)_S = g(z)_S$. To see this, note that

$$\mathop{\mathbb{E}}_{r}\left[\Pr_{z\in Z'}[A(z,r)_S \neq g(z)_S]\right] = \mathop{\mathbb{E}}_{z\in Z'}\left[\Pr_{r}[A(z,r)_S \neq g(z)_S]\right] \leq 2\alpha/\gamma\,,$$

and hence the probability over r that $\Pr_{z\in Z'}[A(z,r)_S \neq g(z)_S] > \frac{2\alpha}{\gamma\cdot\delta'}$ is at most δ'.

Invoking the algorithm of [IJKW10] with the fixed S and r. For $m = |S|$, let $h = f^{\times m}$ be the m-wise direct-product of f. We now use the following algorithm by Impagliazzo *et al.* [IJKW10]:

Theorem 12.9.3 (uniform list-decoding of the direct-product code)**.** *There is a constant $c_{\mathsf{IJKW}} > 1$ and a probabilistic algorithm $\mathtt{Dec}$ such that for any $f\colon \{0,1\}^n \to \{0,1\}$ and $m \in \mathbb{N}$ and $\varepsilon,\delta \in (0,1)$ satisfying $\varepsilon > e^{-\delta m/c_{\mathsf{IJKW}}}$ the following holds. Let $h = f^{\times m}$. Then, when $\mathtt{Dec}$ is given oracle access to a function $\widetilde{h}\colon \{0,1\}^{m\cdot n} \to \{0,1\}^m$ satisfying $\Pr_{z\in\{0,1\}^{m\cdot n}}[\widetilde{h}(z) = h(z)] \geq \varepsilon$, with probability $\Omega(\varepsilon)$ it outputs an oracle circuit $C\colon \{0,1\}^n \to \{0,1\}$ such that $\Pr_{x\in\{0,1\}^n}[C^{\widetilde{h}}(x) = f(x)] \geq 1-\delta$. Moreover, the algorithm $\mathtt{Dec}$ is a uniform randomized NC^0 algorithm that makes just one oracle query to $\widetilde{h}$, and the circuit C is an AC^0 circuit of size $\mathrm{poly}(n,m,\log(1/\delta),1/\varepsilon)$ with $O(\log(1/\delta)/\varepsilon)$ oracle gates to $\widetilde{h}$.*

We invoke the algorithm Dec from Theorem 12.9.3 with parameter values $\varepsilon = \delta/4$ and δ' and m, while giving it oracle access to the function $\widetilde{h}(x_1, ..., x_m) = A_r(z)_{S_1}, ..., A_r(z)_{S_m}$. The constraint on the parameters in Theorem 12.9.3 is

$$\varepsilon > e^{-\delta' m / c_{\text{IJKW}}} \iff \delta/4 > e^{-(\delta')^2/(4\gamma \cdot c_{\text{IJKW}})} ,$$

where the " $\iff$ " is since $m = |S| = \delta'/4\gamma$. The constraint above is then satisfied due to our choice of a sufficiently small γ.

With probability at least $\Omega(\delta)$ the algorithm outputs an oracle circuit that computes f on $1 - \delta'$ of the inputs when given access to A_r. Thus, if we run this algorithm for $O((1/\delta) \cdot \log(1/\delta'))$ times, with probability at least $1 - \delta'$ at least one of the resulting circuits will compute f correctly on $1 - \delta'$ of the inputs when given access to A_r. We get as advice an index for the circuit with maximal agreement with f, which we denote from now on by C. The size of C is at most $\text{poly}(n, m, \log(1/\delta'), 1/\varepsilon) = \text{poly}(n)$, and it makes at most $O(\log(1/\delta')/\varepsilon) = O(1)$ queries to A_r.

The final algorithm. Our algorithm constructs C as above, which is a procedure that does not depend on the input $x \in \{0,1\}^n$, and then outputs $C(x)$. We claim that there exists a set $X \subset \{0,1\}^n$ of density $1 - 9\delta' = 1 - \delta$ such that for every $x \in X$ we have that $\Pr_C[C(x) = f(x)] \geq 2/3$. To see this, recall that the probability that all the three steps in the construction of C above succeed (*i.e.*, we choose a good S, choose a good r, and the invocations of the algorithm from Theorem 12.9.3 succeed) is at least $1 - 3\delta'$. Whenever that happens, the circuit C correctly computes f on $1 - \delta'$ of the inputs x. Thus, we have that

$$\mathop{\mathbb{E}}_{x}\left[\Pr_{C}\left[C(x) \neq f(x)\right]\right] = \mathop{\mathbb{E}}_{C}\left[\Pr_{x}\left[C(x) \neq f(x)\right]\right] \leq 4\delta' ,$$

and hence the probability over x that $\Pr[C(x) \neq f(x)] \geq 1/3$ is at most $12\delta' = \delta$.

Indeed, the algorithm above is probabilistic, and on at least $1 - \delta$ of the inputs computes f with probability at least $2/3$. The running time of the algorithm is dominated by the step of evaluating the oracle circuit C using the algorithm A, which can be done in time $\text{poly}(n) + O(T'(n)) < T(n)$. Also, the algorithm relies on $O(\log(1/\delta))$ bits of advice that depend only on the randomness and not on the input. This contradicts our assumption that $f \notin \mathsf{avg}_{1-\delta}\text{-}\mathsf{BPTIME}[T]//(1/\alpha)$. ■

Chapter 13

Superfast Derandomization of Proof Systems

Contents

13.1 Introduction

We study the well-known problem of *eliminating randomness from interactive proof systems*. While we do not expect to be able to eliminate randomness from general proof systems (as that would imply that NP = IP = PSPACE, by [Sha92]), a classical line of works suggests that for proof systems that only use constantly many rounds of interaction, this might be doable. In particular, assuming sufficiently strong circuit lower bounds,[1] we have that AM = NP (for a subset of prominent works in this area, see *e.g.*, [KvM02, IKW02, MV05, SU05, GSTS03, SU07]).

Recently, Doron *et al.* [DMOZ20] raised the fine-grained derandomization question of what is the *minimal time overhead* that we need to pay when eliminating randomness from probabilistic algorithms. In their paper and in two subsequent works [CT21b, CT21a], the following picture emerged: Under sufficiently strong lower bounds for algorithms that use non-uniformity, we can derandomize probabilistic algorithms that run in time T by deterministic algorithms that run in time $n \cdot T^{1+\varepsilon}$, for an arbitrarily small $\varepsilon > 0$ (see [CT21b] for details);[2] and furthermore, if we are willing to settle for derandomization that errs on a negligible fraction of inputs over any

[1] For example, it suffices to assume that for some $\varepsilon > 0$ it holds that $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ is hard for SVN circuits of size $2^{\varepsilon \cdot n}$ on all input lengths (see [MV05]).

[2] This is tight for worst-case derandomization, under the hypothesis #NSETH (see Assumption 13.3.19).

polynomial-time samplable distribution, then we can derandomize probabilistic time T in deterministic time $n^{\varepsilon} \cdot T$ for an arbitrarily small $\varepsilon > 0$ (see [CT21a]).

This chapter studies the question of *superfast derandomization of proof systems*: We ask what is the minimal time overhead that we need to pay when eliminating randomness from proof systems such as MA and AM. While it is well-known that proof systems with constantly many rounds can be simulated by proof systems with two rounds (see [BM88]), in this work we care about *fine-grained time bounds*, and therefore we care about the precise number of rounds in any such system. We denote by $\mathsf{AMTIME}^{[\rightleftharpoons c]}[T]$ the set of problems solved by proof systems that use c turns of interaction, where in each turn the relevant party communicates $T(n)$ bits (recall that the verifier just sends random bits; see Definition 13.3.5 for precise details).

Our contributions, at a bird's eye. First, we construct *essentially optimal* derandomization algorithms for MA, *and for* AM *protocols with constantly many rounds*. The derandomization algorithms are constructed under appealing hardness hypotheses, which compare favorably to hypotheses from previous works (and the claim that they are essentially optimal is under the assumption #NSETH). The results for MA and for AM are presented in Section 13.1.1 and Section 13.1.2, respectively.

Secondly, we introduce the notion of *deterministic doubly efficient argument systems*, which are deterministic doubly efficient proof systems such that no polynomial-time adversary can find, with non-negligible probability, an input $x \notin L$ and a proof π such that the verifier accepts x given proof π. Under strong hardness assumptions, we compile every constant-round doubly efficient proof system into a deterministic doubly efficient argument system with essentially no time overhead.

As a special case of the latter result, under strong hardness assumptions, we construct a verifier that certifies the number of solutions for a given n-bit formula of size $2^{o(n)}$ in time $2^{\varepsilon \cdot n}$, for any $\varepsilon > 0$, such that no $2^{O(n)}$-time algorithm can find a formula and a proof on which this verifier errs. The general result mentioned in the preceding paragraph as well as the special case of #SAT are presented in Section 13.1.3.

Useful properties. Many of our hardness hypotheses will rely on the existence of *useful properties*, in the sense of Razborov and Rudich [RR97]: Loosely speaking, these are algorithms verifying that a given string is the truth-table of a function that is hard for a certain class. Specifically, for

two complexity classes $\mathcal{C}$ and $\mathcal{C}'$, we say that a property Π of truth-tables is $\mathcal{C}'$-constructive and useful against $\mathcal{C}$ if there is a $\mathcal{C}'$-algorithm that decides whether a truth-table is in Π, and if every truth-table in Π is hard for algorithms from $\mathcal{C}$ (see Definition 13.3.3 for precise details).[3]

The reason that we consider useful properties is that *they are necessary for any derandomization of proof systems, even just of* MA. In fact, the useful properties that are necessary are constructive in *quasilinear time* (this follows using the well-known approach of Williams [Wil13a, Wil16b]; see Section 13.8 for a proof). Accordingly, we will consider useful properties in which truth-tables of length $N = 2^n$ can be recognized in nondeterministic polynomial time $N^k = 2^{k \cdot n}$ or even in nondeterministic near-linear time $N^{1+\varepsilon} = 2^{(1+\varepsilon) \cdot n}$, for a small constant $\varepsilon > 0$.

Previous works deduced superfast derandomization from hard functions whose truth-tables can be printed quickly (*i.e.*, they have bounded amortized complexity; see, *e.g.*, [CT21b]). The assumptions above are more relaxed: We only require recognizing the truth-table (rather than printing it), we allow non-determinism (*i.e.*, proof) in recognizing the truth-table, and we allow for a collection of hard truth-tables per input length, rather than just a single hard truth-table per input length.

13.1.1 Superfast Derandomization of MA

Our first result is a derandomization of MA that induces a (near-)quadratic time overhead. This derandomization is (conditionally) tight, and it relies on the existence of constructive properties that are useful against $\mathsf{NP} \cap \mathsf{coNP}$ machines that receive near-maximal non-uniform advice. That is:

Theorem 13.1.1 (quadratic derandomization of MA and useful properties)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists an $\mathsf{NTIME}[N^{2+\varepsilon/4}]$-constructive property useful against $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{(2-\delta) \cdot n}]/2^{(1-\delta) \cdot n}$. Then,*

$$\mathsf{prMATIME}[T] \subseteq \mathsf{prNTIME}[T^{2+\varepsilon}] \,.$$

We stress that, in contrast to derandomization of prBPP, for MA the quadratic time overhead above might be unavoidable. To see this, assume that #NSETH holds; that is, for every $\varepsilon > 0$ there is no nondeterministic algorithm running in time $2^{(1-\varepsilon) \cdot n}$ and *counting* the number of satisfying assignments for a given n-bit formula of size $2^{o(n)}$ (see Assumption 13.3.19). Then, for every $\varepsilon >$

[3] We also assume non-triviality, *i.e.* for every $n \in \mathbb{N}$ the property Π contains functions on n input bits.

0 we have that $\mathsf{MATIME}[T] \not\subseteq \mathsf{NTIME}[T^{2-\varepsilon}]$: The reason is that MA algorithms may simulate a sumcheck protocol to count the number of satisfying assignments of a given formula in time $\tilde{O}(2^{n/2})$ (see [Wil16b]). See Theorem 13.6.1 for a more general statement.

The technical result underlying Theorem 13.1.1 is actually stronger: Under the hypothesis we deduce that $\mathsf{prBPTIME}[T] \subseteq \mathsf{prNTIME}[T^{2+\varepsilon}]$ (see Theorem 13.4.3), and if we assume that a *deterministic* algorithm can quickly print the hard truth-tables, we can deduce that $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[T^{2+\varepsilon}]$ (see Theorem 13.4.4). The latter extension to prBPTIME compares favorably to assumptions needed to get the same conclusion in previous works [DMOZ20, CT21b]; we elaborate on this in Section 13.1.4.

13.1.2 Superfast Derandomization of AM

Our next step is to consider derandomization of AM protocols, and for simplicity of presentation we focus on protocols with perfect completeness.[4] Our second main result is a derandomization of $\mathsf{AMTIME}^{[\rightleftharpoons c]}[T]$, for any constant number $c \in \mathbb{N}$ of turns, that is (conditionally) tight: Loosely speaking, we prove that any such protocol can be simulated in nondeterministic time $\approx T^{c/2}$, assuming the existence of constructive properties that are useful against MAM protocols that receive near-maximal non-uniform advice. That is:

Theorem 13.1.2 (superfast derandomization of AM)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that for every $k \geq 1$ there exists an $\mathsf{NTIME}[N^{k+\varepsilon/3}]$-constructive property useful against $\mathsf{MAMTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$. Then, for every polynomial T it holds that*

$$\mathsf{prAMTIME}[T] \subseteq \mathsf{prNTIME}[n \cdot T^{1+\varepsilon}] \,,$$

and furthermore for every constant $c \in \mathbb{N}$ it holds that

$$\mathsf{prAMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{prNTIME}[n \cdot T^{\lceil c/2\rceil+\varepsilon}] \,.$$

The conclusions in Theorem 13.1.2 are essentially tight, assuming #NSETH; see Theorem 13.6.2 for details. Moreover, the hardness assumption is quite mild compared to what one might expect; let us explain why we claim so, comparing our assumption to ones in previous results. Recall that in classical results, to derandomize AM we assume lower bounds for the non-uniform analogue

[4] Our results extend to the case of protocols with imperfect completeness, at the cost of strengthening the hardness assumptions (see, *e.g.*, Remark 13.5.4 and Remark 13.7.13).

of NP (see, *e.g.*, [SU05]). The assumption above is for a stronger class, namely the non-uniform analogue of MAM. However, interestingly, this assumption still suffices to *optimally* derandomize AM with *any* constant number of turns; that is, the derandomization overhead is optimal for any number of rounds given hardness only for (non-uniform) MAM.

Now, observe that our assumption refers to hardness for time bounds $2^{kn} \gg 2^n$ (*i.e.*, upper bounds of 2^{kn} and lower bounds of $2^{(1-\delta)\cdot kn}$, for every constant $k \geq 1$). This is quantitatively reminiscent of a hardness assumption from [CT21b] that was used to deduce that $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[n \cdot T^{1+\varepsilon}]$. However, the latter result also needed an additional hardness assumption, namely the existence of *one-way functions*; in contrast, in Theorem 13.1.2 *we do not rely on any cryptographic assumption*. In fact, similarly to Section 13.1.1, the techniques underlying Theorem 13.1.2 also extend to the setting of derandomization of prBPP, and allow us to deduce a conclusion as in [CT21b] without cryptographic assumptions. For further details see Section 13.1.4.

A strengthening. We further strengthen Theorem 13.1.2 by relaxing its hypothesis. Specifically, recall that in Theorem 13.1.2 we assumed that for every $k \geq 1$ there is a corresponding useful property, and let us denote it by $\mathcal{L}_k$. We prove a stronger version that relies on the same hardness hypothesis for $\mathcal{L}_1$, but for every $k > 1$ only requires $\mathcal{L}_k$ to be hard for $\mathsf{NTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$ (rather than for MAM protocols with the same complexity). See Section 13.5.3 and Theorem 13.5.20 for details.

Uniform tradeoffs for $\mathsf{AM} \cap \mathsf{coAM}$. The results above relied on hardness for protocols that use a large number of non-uniform advice bits. Classical results were able to deduce derandomization of the more restricted class $\mathsf{AM} \cap \mathsf{coAM}$ relying only on hardness assumptions for *uniform* protocols (see, *e.g.*, [GSTS03, SU07]).

Indeed, we are able to show an "extreme high-end" analogue of these results too. Assuming that there exists a function whose truth-tables can be recognized in near-linear time but that is hard for 7-round protocols running in time $2^{(1-\delta)\cdot n}$, we show that $\mathsf{AM} \cap \mathsf{coAM}$ can be derandomized with only a quadratic time overhead:

Theorem 13.1.3 (uniform tradeoffs for superfast derandomization)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \notin$ i.o.-$(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}_2^{[\rightleftharpoons 7]}[2^{(1-\delta)\cdot n}]$ such that truth-tables of L of length $N = 2^n$ can be recognized in nondeterministic time $N^{1+\varepsilon/3}$.* [5] *Then,*

[5] That is, we have that $\mathsf{tt}(L) \in \mathsf{NTIME}[N^{1+\varepsilon/3}]$, where $\mathsf{tt}(L) = \{f_n \in \{0,1\}^{N=2^n}\}_{n\in\mathbb{N}}$ and f_n is the truth-table of

for every time-computable T it holds that

$$(\mathsf{AM} \cap \mathsf{coAM})\mathsf{TIME}[T] \subseteq (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T^{2+\varepsilon}].$$

Needless to say, hardness for protocols with constantly many rounds did not appear in previous results [GSTS03, SU07], and this is because previous results did not consider fine-grained time bounds (in which case constantly many rounds can be simulated by two rounds [BM88]).

13.1.3 A Lunch That Looks Free: Deterministic Doubly Efficient Arguments

In Chapter 11 and Chapter 12, the way to bypass the #NSETH barrier and obtain faster derandomization results was to consider derandomization *that is allowed to err on a tiny fraction of inputs*, and to construct derandomization algorithms that use new *non-black-box techniques*. In this chapter we follow in the same vein.

Let us first state a striking special case of the general results that we prove. Under a very strong hardness assumption (we will be formal about the assumption when we state the general case), we prove that for every $\varepsilon > 0$ there is a verifier that *counts the number of satisfying assignments for a given n-bit formula in time $2^{\varepsilon \cdot n}$*, with one caveat: The soundness of the verifier is only *computational*, rather than information-theoretic.

Theorem 13.1.4 (effectively certifying #SAT in deterministic time $2^{\varepsilon \cdot n}$; informal)**.** *Assume that a certain lower bound for probabilistic machines with oracle access to proof systems holds (see Theorem 13.7.16). Then, for every $\varepsilon > 0$ there is a deterministic verifier V that gets as input an n-bit formula Φ of size at most $2^{o(n)}$, runs in time $2^{\varepsilon \cdot n}$, and satisfies the following:*

1. **(Completeness.)** *There is an algorithm that, given any input formula Φ as above, runs in time $2^{O(n)}$ and outputs a proof π such that $V(\Phi, \pi) = \#\mathsf{SAT}(\Phi)$.*[6]
2. **(Computational soundness.)** *For every probabilistic algorithm $\tilde{P}$ running in time $2^{O(n)}$, the probability that $\tilde{P}(1^n)$ prints an n-bit formula Φ of size $2^{o(n)}$ and proof π such that $V(\Phi, \pi) \notin \{\bot, \#\mathsf{SAT}(\Phi)\}$ is $2^{-\omega(n)}$.*

We remind the reader that constructing a verifier as above without the relaxation of computational hardness (*i.e.*, a verifier that is sound for all inputs and with any proof) is believed to be

L on n-bit inputs. Also, the subscript "2" in the notation $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}_2^{[\rightleftharpoons 7]}$ refers to $\mathsf{MA} \cap \mathsf{coMA}$ protocols with imperfect completeness (say, $2/3$).

[6] We denote by $\#\mathsf{SAT}(\Phi)$ the number of assignments that satisfy the formula Φ.

impossible; this is known as the #NETH assumption, which is weaker than the "strong" version #NSETH mentioned above. To the best of our knowledge, the existence of a verifier such as the one in Theorem 13.1.4 was not conjectured before.

Deterministic doubly efficient argument systems

More generally, consider the notion of doubly efficient proof systems, as introduced by Goldwasser, Kalai, and Rothblum [GKR15] (for a survey, see [Gol18]). These are interactive proof systems in which the verifier is very fast (say, runs in quasilinear time) and the honest prover is slower, but still efficient (say, runs in polynomial time). We denote by $\mathsf{delP}^{[\rightleftharpoons c]}[T]$ the class of problems solvable by doubly efficient proof systems with c turns of interaction, a verifier that runs in time T, and an honest prover that runs in time $\mathrm{poly}(T)$ (see Definition 13.3.6).[7]

We initiate a study of deterministic doubly efficient argument systems. Recall that argument systems are a standard relaxation of proof systems in which soundness is guaranteed only against computationally bounded provers (see, *e.g.*, [Tha22] for an exposition, and [Gol18, Section 1.5] for a discussion of probabilistic doubly efficient argument systems). In our notion of deterministic doubly efficient argument systems *we require the verifier to be deterministic* (*i.e.*, an NP-type verifier), and the honest prover to be efficient, but we further relax the soundness requirement such that it is guaranteed only against inputs that can be found by computationally bounded adversaries.

Definition 13.1.5 (deterministic doubly efficient argument system)**.** *We say that $L \subseteq \{0,1\}^*$ is in $\mathsf{NARG}[T]$ if there exists a deterministic T-time verifier V such that the following holds:*

1. *There exists a deterministic algorithm P that, when given $x \in L$, runs in time $\mathrm{poly}(T)$ and outputs π such that $V(x,\pi) = 1$.*
2. *For every polynomial p, and every probabilistic algorithm $\tilde{P}$ running in time $p(T)$, and every sufficiently large $n \in \mathbb{N}$, the probability that $\tilde{P}(1^n)$ prints $x \notin L$ of length n and $\pi \in \{0,1\}^{T(n)}$ such that $V(x,\pi) = 1$ is $T(n)^{-\omega(1)}$.*

Intuitively, the meaning of the soundness condition in Definition 13.1.5 is that no efficient adversary can find a false claim $x \notin L$ and then convince the verifier that the false claim is true, except with negligible probability.

One may wonder whether we can relax the condition that the honest prover is efficient. We remark that such a definition would be too relaxed, and potentially allow the resulting class to

[7] The original definition of doubly efficient proof systems uses the fixed time bound $T = \tilde{O}(n)$ (see, *e.g.*, [GKR15, Gol18]). In this work we also consider more general time bounds.

contain essentially all possible problems (even uncomputable ones); we elaborate and further discuss our definitional choices in Section 13.3.2.

Derandomizing doubly efficient proofs into deterministic doubly efficient arguments

Our main result in this context is that, under strong hardness hypotheses, *every constant round doubly efficient proof system* can be simulated by a deterministic doubly efficient argument system *with essentially no time overhead*. We stress that even if we start with a system that has many rounds, after this simulation we still end up with an NP-type verifier that has essentially the same time complexity. This opens the door to first using a protocol with many rounds to solve a problem faster, and then simulating this fast protocol by a deterministic doubly efficient argument system of roughly the same complexity. (This is the approach that underlies Theorem 13.1.4.)

As a first step, consider any $\mathsf{deIP}^{[\rightleftharpoons c]}[T]$ protocol in which the verifier uses only $n^{o(1)}$ random coins. We simulate it by a deterministic doubly efficient argument system, under the following assumption: There exists $f\colon \{0,1\}^{O(T)} \to \{0,1\}^{n^{o(1)}}$ such that each output bit of f can be computed in time $\bar{T}$, but no probabilistic machine with oracle access to $\mathsf{prAMTIME}^{[\rightleftharpoons c]}[n]$ can print (an approximate version) of the entire string $f(x)$ in time slightly larger $\bar{T}$; and this hardness holds with high probability when choosing x from any efficiently samplable distribution.

Assumption 13.1.6 (non-batch-computability assumption; see Assumption 13.7.4)**.** *The* $(N \mapsto K, K', \eta)$-non-batch-computability *assumption for time* $\bar{T}$ *with oracle access to* $\mathcal{O}$ *is the following. There exists a function* $f\colon \{0,1\}^N \to \{0,1\}^K$ *such that:*

1. *There exists a deterministic algorithm that gets input* $(z, i) \in \{0,1\}^N \times [K]$ *and outputs the* i^{th} *bit of* $f(z)$ *in time* $\bar{T}$*.*
2. *For every oracle machine* M *running in time* $\bar{T} \cdot K'$ *with oracle access to* $\mathcal{O}$*, and every distribution* $\mathbf{z}$ *over* $\{0,1\}^N$ *that is samplable in time polynomial in* $\bar{T}$*, with probability at least* $1 - (\bar{T})^{-\omega(1)}$ *over choice of* $z \sim \mathbf{z}$ *it holds that* $\Pr[M^{\mathcal{O}}(z)_i \neq f(z)_i] \geq \eta$*, where the probability is over* $i \in [K]$ *and the random coins of* M*.* [8]

Indeed, the name "non-batch-computability" in Assumption 13.1.6 refers to the fact that each individual output bit of f is computable in time $\bar{T}$, whereas printing (an approximate version) of the entire K-bit string is hard for time $\bar{T} \cdot K' > \bar{T}$ (even with oracle access to $\mathcal{O}$). Our first general result is then the following:

[8] In all our results we will use this assumption with $\bar{T}(N) = \mathrm{poly}(N)$, in which case the error bound $(\bar{T})^{-\omega(1)}$ is just any negligible function in the input length.

Theorem 13.1.7 (simulating doubly efficient proof systems with few coins by deterministic doubly efficient argument systems; informal, see Theorem 13.7.5)**.** *Let $L \in \mathsf{delP}^{[\rightleftharpoons c]}[T]$ such that the verifier for L uses only $R(n) = n^{o(1)}$ random coins. Assume that for some $\alpha, \beta \in (0,1)$ the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T}$ with oracle queries of length $O(T)$ to $\mathsf{prAMTIME}_2^{[\rightleftharpoons c]}[n]$, where*

$$N(n) = n + c \cdot T$$
$$K(n) = \mathrm{poly}(R(n))$$
$$\bar{T}(n) = T \cdot K\,.$$

Then $L \in \mathsf{NARG}[T \cdot n^{o(1)}]$.

The hardness hypothesis in Theorem 13.1.7 is very strong, but the corresponding derandomization conclusion is also exceptionally strong. We discuss the hardness hypothesis in detail in Section 13.7.2, and in particular compare it to known results about batch-computing functions by interactive protocols, by Reingold, Rothblum and Rothblum [RRR21, RRR18]. Indeed, even an assumption considerably stronger than the one in Theorem 13.1.7 still seems reasonable, in particular an assumption in which the oracle is a linear-space machine (rather than a linear-time verifier in an interactive protocol); see Section 13.7.2 for details. We also comment that "non-batch-computability" hardness against probabilistic machines (without oracles) is necessary for derandomization with no overhead in limited special cases (see [CT21a, Section 6.3]).

Theorem 13.1.7 implies Theorem 13.1.4 as a special case, because the underlying probabilistic proof system for #SAT (*i.e.*, a constant-round sumcheck protocol) uses a number of random coins that is logarithmic in the running time. Moreover, in the special case of Theorem 13.1.4 we are able to *considerably relax the hypothesis*, relying on certain properties of the proof system for #SAT; see Theorem 13.3.17, Theorem 13.7.15, and Theorem 13.7.16 for details.

The general case. Turning to the general case of simulating doubly efficient proof systems (without restricting the number of random coins) by deterministic doubly efficient argument systems, we can do so under one additional hypothesis:

Theorem 13.1.8 (simulating general doubly efficient proof systems by deterministic doubly efficient argument systems; informal, see Corollary 13.7.20)**.** *For every $\alpha, \beta, \varepsilon \in (0,1)$ there exists $\eta, \delta > 0$ such that for every polynomial $T(n)$ and constant $c \in \mathbb{N}$ the following holds. Assume that:*

1. *There exists* $L^{\mathsf{hard}} \notin$ *i.o.-*$\mathsf{MATIME}^{[\rightleftharpoons c+1]}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ *such that given* $n \in \mathbb{N}$*, the truth-table of* L^{hard} *of* n*-bit inputs can be printed in time* $2^{(1+\varepsilon/3)\cdot n}$*.*
2. *The* $(N \mapsto K, K^{\beta}, \alpha)$*-non-batch-computability assumption holds for time* $\bar{T}$ *with oracle queries of length* $O(T)$ *to* $\mathsf{prAMTIME}_2^{[\rightleftharpoons c]}[n]$*, where* $N = n + c \cdot T^{1+\varepsilon/2}$ *and* $K = \mathrm{polylog}(T)$ *and* $\bar{T} = T^{1+\varepsilon/2} \cdot K$*.*

Then, $\mathsf{delP}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{NARG}[T^{1+\varepsilon}]$.

The additional hypothesis in Theorem 13.1.8 (*i.e.*, the one in Item (1)) is a strengthening of the one in Theorem 13.1.2 that refers to the value of $k = 1$. The strengthening is because now we consider hardness for protocols with $c + 1$ turns rather than 3 turns, we assume a single hard problem rather than a useful property, and the upper-bound on the complexity of truth-tables is deterministic rather than nondeterministic.

Deterministic argument systems for NP-relations

In Definition 13.1.5 the honest prover runs in time $\bar{T} = \mathrm{poly}(T)$, and therefore such argument systems exist only for problems in $\mathsf{DTIME}[\bar{T}]$. An alternative definition, which would allow constructing deterministic argument systems for $\mathsf{NTIME}[\bar{T}]$-relations, is to give the honest prover a witness for the corresponding relation as auxiliary input (while still insisting that it runs in time $\bar{T}$). Our proofs extend to this setting, allowing to simulate $\mathsf{AMTIME}^{[\rightleftharpoons c]}[T]$ protocols for $\mathsf{NTIME}[\bar{T}]$-relations, in which the honest prover is efficient given a witness for the relation, into deterministic argument systems running in time $T^{1+\varepsilon}$. For further details see Remark 13.7.14.

13.1.4 Implications for Derandomization of prBPP

Our proof techniques also extend to the setting of derandomization of prBPP (rather than just of proof systems), and in this setting they yield results that compare favorably to previous works. For derandomization in this setting we replace the assumptions about useful properties with assumptions that truth-tables of hard functions can be efficiently printed. (This is since our derandomization algorithm cannot guess-and-verify a hard truth-table, but needs to efficiently print it.) It is useful to think of an algorithm that prints a truth-table of a function as "batch-computing" the function.

The techniques underlying Theorem 13.1.1 extend to show a quadratic-time derandomization of prBPP, assuming that a function whose truth-tables can be printed in time $2^{(2+\varepsilon/3)\cdot n}$ is hard for

$\mathsf{NTIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ (see Theorem 13.4.4).[9] This compares favorably to two relevant results in the previous works [DMOZ20, CT21b], as follows. (In the table below, the upper-bound is for a deterministic algorithm that prints the truth-table of the hard function on n bits.)

Upper bound	**Lower bound**	**Time overhead**	
$2^{(2+\Theta(\varepsilon))\cdot n}$	$\mathsf{NTIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$	$T^{2+\varepsilon}$	Theorem 13.4.4
$2^{(2+\Theta(\varepsilon))\cdot n}$	$\mathsf{MATIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$	$T^{2+\varepsilon}$	[DMOZ20]
$2^{(3/2)\cdot n}$	$\mathsf{NTIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$	$T^{3+\varepsilon}$	[CT21b, Thm 1.8]

While the three results above are formally incomparable, in this chapter we are able to obtain derandomization with quadratic time overhead from hardness for NTIME machines with advice, whereas previous works either assumed hardness for MATIME machines with advice, or paid a cubic time overhead.

Moreover, the techniques underlying Theorem 13.1.2 extend to yield the conclusion $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[n \cdot T^{1+\varepsilon}]$, which is optimal under #NSETH, under an assumption that compares favorably to the one used to obtain derandomization with such overhead in Chapter 11. In Chapter 11, we obtained this conclusion using a cryptographic assumption (the existence of one-way functions) as well as a hardness assumption that is necessary for obtaining the conclusion using PRGs. In this chapter we are able to replace the cryptographic assumption with a hardness assumption for MA protocols with advice (see Theorem 13.5.5 for precise details).

13.2 Technical Overview

Most of the proofs in this work sequentially build on the ideas of each other, and therefore we encourage readers read the current section sequentially.

In Section 13.2.1 we describe the simplest proof, which is of Theorem 13.1.1. Then in Section 13.2.2 we build on this proof to prove Theorem 13.1.2. In Section 13.2.3 we describe the more involved proofs in this chapter, which are for the results Section 13.1.3; these are inherently different from other proofs, and use non-black-box techniques while also relying on the previous proofs. We conclude by explaining the proof of Theorem 13.1.3 in Section 13.2.4 (this proof does not rely on Section 13.2.3).

[9] For simplicity, in this section we assert lower bounds for NTIME and MATIME. The results that we mention only require hardness for $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}$ and $\mathsf{MA}\cap\mathsf{coMATIME}$, respectively.

Basic building-block: A simple and efficient PRG. The starting point for our results is a very simple PRG, which was recently used for superfast derandomization in [DMOZ20, CT21b] (its ideas date back to [Sip88] and variations on them have been used for derandomization of proof systems, *e.g.* [MV05]). Recall that a reconstructive PRG is a pair of efficient algorithms: A generator Gen maps a truth-table f to a list G^f of strings; and a reconstruction Rec maps every distinguisher D for G^f to an efficient procedure Rec^D that computes f. [10] Indeed, if f is hard for algorithms with complexity as that of Rec^D, then it follows that G^f is pseudorandom.

The simple reconstructive PRG works as follows. Given f, the generator Gen encodes it to $\bar{f} = \mathsf{Enc}(f)$ by an error-correcting code that is encodable in near-linear time and locally list-decodable, partitions $\bar{f}$ into consecutive substrings of size $N = |f|^{.99}$, and outputs the list of $|\bar{f}|/N \approx N^{.01}$ substrings. Indeed, this generator Gen runs in near-linear time, and is thus particularly suitable for constructing superfast derandomization algorithms. In contrast, the reconstruction Rec is inefficient, and in addition only works when the distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ is extremely biased; for example, when D accepts all but $2^{N^{.99}}$ strings and yet rejects 1% of the strings in G^f. [11]

Note that for any such D, a random pairwise-independent hash function $h\colon \{0,1\}^N \to \{0,1\}^{N^{.999}}$ will not have collisions in the set $D^{-1}(0)$. The reconstruction Rec gets as advice a description of such h and the hash values of the N-bit substrings of $\bar{f}$ that G^f outputs. Given input $z \in [|f|]$, it runs the local list-decoder for Enc, and whenever the decoder queries $q \in [|\bar{f}|]$, the reconstruction *nondeterministically de-hashes* the corresponding substring in $\bar{f}$ to obtain $\bar{f}_q$. That is, denoting by $\bar{f}^{(i)}$ the substring of $\bar{f}$ that contains the index q, the reconstruction Rec guesses $z \in \{0,1\}^N$, verifies that $h(z) = h(\bar{f}^{(i)})$ using the stored hash value, verifies that $z \in D^{-1}(0)$ by querying D, and if the verifications passed then $z = \bar{f}^{(i)}$ (since there are no collisions in $D^{-1}(0)$) and Rec outputs the bit in z corresponding to index q.

We refer the reader to Proposition 13.4.1 for a clean statement that outlines the foregoing basic version of this reconstructive PRG.

Preliminary observations: Using this approach for derandomizing proof systems. Note that the reconstruction Rec is nondeterministic, and thus when using (Gen, Rec) above we need to

[10] When we say that D is a distinguisher for a list $G^f \subseteq \{0,1\}^n$, we mean that D distinguishes the uniform distribution over the list from a truly uniform n-bit string.

[11] Thus, the reconstructive PRG (Gen, Rec) is particularly suitable for the task of *quantified derandomization*; see [GW14, DMOZ20, CT21b, Tel21] for further details about this application.

assume hardness for nondeterministic procedures. Such assumptions are natural in the current context of derandomizing proof systems. In addition, the natural way to use Gen for derandomizing proof systems, which is indeed the one that we will use for many of our results, is to receive a truth-table f from a prover, verify that f is indeed hard (using an assumption that there is a constructive and useful property), and then instantiate the PRG Gen^f.

13.2.1 Warm-up: Proof of Theorem 13.1.1

The main bottleneck in the previous proofs [DMOZ20, CT21b] that used $(\mathsf{Gen}, \mathsf{Rec})$ came from the fact that Gen only fools extremely biased distinguishers, whereas our goal is to construct a PRG that fools all distinguishers. In previous works this was bridged by straightforward error-reduction: They considered a procedure $\bar{D}(z)$ that uses a randomness-efficient sampler Samp and verifies that z satisfies

$$\Pr_{i\in[N^{1.01}]}[D(\mathsf{Samp}(z,i))=1] \approx \Pr_{r\in\{0,1\}^N}[D(r)=1]\,,$$

and noted that if Gen^f fools $\bar{D}$, then $\mathsf{S}\circ\mathsf{Gen}^f = \left\{\mathsf{Samp}(\mathsf{Gen}^f,s)\right\}_{s\in[N^{1.01}]}$ fools D. (See [CT21b, Section 5.2] for a detailed explanation.)

To see why this is a bottleneck, note that the resulting $\bar{D}$ is of size $O(N^{2.01})$ (because Samp uses $N^{1.01}$ seeds s). Thus, if we use f that is hard for nondeterministic circuits of such size, we need $|f| > N^{2.01}$ and the number of pseudorandom strings is $(\bar{f}/N)\cdot N^{1.01} > N^{2.01}$. Evaluating D on each of the strings, this yields derandomization in near-cubic time.[12]

The observation leading the way to Theorem 13.1.1 is simple: We do not really need f to be hard for nondeterministic circuits of size $N^{2.01}$, but only for *nondeterministic algorithms* that run in time $N^{2.01}$ and use $|f|^{.99}+N$ bits of advice. There can indeed be such truth-tables of size $|f|=N^{1.01}$, as in the hypothesis of Theorem 13.1.1.

To elaborate, let us sketch the proof, and in fact let us show how to derandomize prBPP in near-quadratic nondeterministic time. We are given $D\colon\{0,1\}^N\to\{0,1\}$ of linear size, and we want to approximate its acceptance probability up to a small additive error. We guess f of size $|f|\approx N^{1.01}$ and verify in time $N^{2.02}$ that f is hard for nondeterministic algorithms running in time $N^{2.01}$ and using $O(N)$ bits of advice. Our generator is $\mathsf{S}\circ\mathsf{Gen}^f$, yielding $N^{1.01}$ pseudorandom strings and

[12] An alternative approach from these works is to allow $\bar{D}$ to use randomness, in which case it is only of size $O(N)$ and the derandomization runs in quadratic time. However, this requires assuming that f is hard for non-uniform MA circuits, an assumption we are trying to avoid.

derandomization in quadratic time. The reconstruction Rec gets D and the hash values as advice; whenever the list-decoder queries $\bar{D}$, then Rec computes $\bar{D}$ with queries to D in time $N^{2.01}$. See Section 13.4 for further details.

13.2.2 Proof of Theorem 13.1.2

Turning to AM protocols, let V be an $\mathsf{AMTIME}[T]$ verifier, and recall that we want to derandomize V in $\mathsf{NTIME}[n \cdot T^{1.01}]$. The approach above can still be used, but it yields derandomization in quadratic time rather than in time $n \cdot T^{1.01}$.

The main idea in the proof is to *compose the generator* $\mathsf{S} \circ \mathsf{Gen}$ *with itself*, using different hard truth-tables, to get a PRG with only $n^{1.01}$ seeds rather than $T^{1.01}$ seeds. In high-level, we first use $\mathsf{S} \circ \mathsf{Gen}^{f_1}$ with a long truth-table (of size $|f_1| = T^{1.01}$) to transform V into a verifier V' with running time $T^{1.01}$ that uses only $O(\log T)$ random coins, and then use $\mathsf{S} \circ \mathsf{Gen}^{f_2}$ with a short truth-table (of size $|f_2| = n^{1.01}$) to fully derandomize V', using a seed of length $(1.01) \cdot \log(n)$. [13] Details follow.

As a first step, we will need a refined version of $(\mathsf{Gen}, \mathsf{Rec})$. Recall that in Theorem 13.1.2 we are assuming hardness for probabilistic protocols. Following an idea from [DMOZ20], the reconstruction can now compute $\bar{D}$ probabilistically rather than deterministically, and thus use only a small number queries to D rather than $N^{1.01}$; this reduces the running time to be close to T rather than to T^2. Furthermore, we observe that *the reconstruction* Rec *does not actually use the full power of its oracle access to* D: Loosely speaking, there is $\alpha \in (0,1)$ such that for a good nondeterministic guess, it suffices to "prove" to Rec that an α-fraction of Rec's queries are accepted by D; and simultaneously, for a bad nondeterministic guess, less than α-fraction of Rec's queries will be accepted by D. See Proposition 13.5.2 for precise details and a proof.

Now, when derandomizing AM with perfect completeness, the distinguisher D is a nondeterministic procedure testing whether there exists a satisfying witness for a given input and random coins.[14] A naive instantiation of Rec would thus yield an $\mathsf{MA}^{\mathsf{NP}}$ machine, whereas we want to assume hardness only for MAM. To do so we rely on the observations about Rec above: Our reconstruction algorithm will first receive a witness for Rec, then it will use randomness to choose

[13] This approach follows an idea from [CT21b], wherein superfast derandomization was achieved by composing two PRGs in a similar way. However, in [CT21b] one of the PRGs relied on cryptographic assumptions, and the other was the Nisan-Wigderson [NW94] PRG. In contrast, in this work there are no cryptographic assumptions, and we simply compose two instantiations of $(\mathsf{Gen}, \mathsf{Rec})$.

[14] In works concerning derandomization of AM the output of D is often negated, and it is then thought of as co-nondeterministic circuit. This is done when considering hitting-set generators for D, for derandomizing protocols with perfect completeness. We avoid doing so, and as mentioned in Section 13.1 our results easily generalize to derandomization of protocols with imperfect completeness (see Remark 13.5.4).

a small number of queries (thereby reducing the running time to near-linear), and finally it will send the queries to the prover to obtain nondeterministic witnesses for D on these queries. This yields an MAM protocol, and we show that it indeed suffices for the reconstruction to work (for details see Proposition 13.5.2 and the proof of Proposition 13.5.3).

Now, denote the running time of the protocol V that we want to derandomize by $T(n) = n^k$. Recall that our derandomization will compose $\mathsf{S} \circ \mathsf{Gen}$ using a truth-table f_1 of length $T^{1.01}$ and another truth-table f_2 of length $n^{1.01}$. We assume that f_1 can be verified in near-linear time $T^{1.01}$ and is hard for $\mathsf{MAMTIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$, and that f_2 can be verified in time $n^{k+.01}$ and is hard for $\mathsf{MAMTIME}[2^{(1-\delta)k\cdot n}/2^{(1-\delta)\cdot n}]$. Thus, we can verify both in time $T^{1.01}$, and using the reconstruction argument above, both of them are pseudorandom for V, which runs in time T. Our derandomization enumerates over the output strings of $\mathsf{S} \circ \mathsf{Gen}^{f_2}$, and uses each string as a seed for $\mathsf{S} \circ \mathsf{Gen}^{f_1}$.[15] The resulting pseudorandom set has $n^{1.01}$ strings and can be computed in time $n^{1.01} \cdot T^{1.01} < n \cdot T^{1.02}$, which suffices for our derandomization of V.

To extend this result to AM protocols with constantly many rounds, we first simulate any constant-round protocol by a two-round protocol, and then apply the result above as a black-box. Indeed, the key observation is that the simulation overhead when using the classical result of [BM88] allows us to obtain tight results (under #NSETH), while only assuming hardness for MAM with advice; see Section 13.5.2 for details.

A relaxation of the hypothesis. As mentioned after the statement of Theorem 13.1.2, we further relax its hypothesis, by requiring that for $k > 1$, the property $\mathcal{L}_k$ will be useful only against NTIME machines with the specified time and advice complexity (rather than against MAM protocols of this complexity). Our approach for doing so is to replace the inner PRG $\mathsf{S} \circ \mathsf{Gen}^{f_2}$ with the *Nisan-Wigderson* generator [NW94]. This can be useful for us, because the NW generator is suitable for our superfast parameter setting when it is instantiated for a small output length (*i.e.*, $|f|^\eta$ where $|f|$ is the truth-table length and $\eta \ll \varepsilon$ is a sufficiently small constant; see Theorem 13.5.18), and $\mathsf{S} \circ \mathsf{Gen}^{f_1}$ reduces the number of random coins to $O(\log(T)) = O(\log(n))$.

The main obstacle towards applying the NW generator in this setting is that it requires hardness against machines with advice and (non-adaptive) *oracle access* to NTIME (*i.e.*, to the distinguisher, which in this case is an NTIME machine), whereas we only assume hardness against NTIME *machines* with advice. To bridge this gap, we use an idea of Shaltiel and Umans [SU06] (fol-

[15] Indeed, the length of output strings of $\mathsf{S} \circ \mathsf{Gen}^{f_2}$ is an "overkill", since they are of length close to n but we only use the first $O(\log(n))$ bits in them as a seed for $\mathsf{S} \circ \mathsf{Gen}^{f_1}$.

lowing [FF93, SU05]), which allows to transform truth-tables that are hard for NTIME machines with advice into truth-tables that are hard for machines with advice and non-adaptive oracle access to NTIME. We use their transformation while analyzing it in a more careful way, which allows us to bound the overheads in the hardness of the truth-table incurred by the transformation. (See Section 13.5.3 for further details.)

13.2.3 Proofs of the Results from Section 13.1.3

We simulate probabilistic doubly efficient proof systems by deterministic doubly efficient argument systems using *non-black-box derandomization algorithms*, rather than PRGs. Specifically, we will use a targeted pseudorandom generator, which takes an input z and prints a list of strings that looks pseudorandom to efficient algorithms that also have access to the same z.

Our targeted PRG will be reconstructive (*i.e.*, based on a hard function), and will thus yield an "instance-wise" hardness-vs-randomness tradeoff: If the underlying function is hard to compute on a set S of inputs of density $1-\mu$, then the targeted PRG is pseudorandom on each and every input in S.[16] The specific generator that we will use, from [CT21a], is denoted G_f and relies on a function $f\colon \{0,1\}^* \to \{0,1\}^*$ with *multiple output bits* such that:

1. **(Upper bound.)** Individual output bits of the string $f(z)$ can be computed in time T'.
2. **(Lower bound.)** The entire string $f(z)$ cannot be approximately printed in time $T' \cdot |f(x)|^\beta$, for a small constant $\beta > 0$.[17]

Note that the obvious algorithm prints $f(z)$ in time $T' \cdot |f(z)|$, but f is *"non-batch-computable"*, in the sense that printing (an approximate version of) the entire string cannot be done in time close to that of computing a single bit (*i.e.*, in time $T' \cdot |f(x)|^\beta$).

As a first step we will consider derandomizing verifiers that use a small number of coins, say $n^{o(1)}$. We instantiate G_f accordingly with $n^{o(1)}$ output bits, and with f and a time bound T' that is slightly larger than the running time of the verifier. In this setting G_f runs in time $T' \cdot n^{o(1)}$ and prints $n^{o(1)}$ random strings, and thus reduces the number of random coins (to $o(\log(n))$) without increasing the time complexity. To deduce that G_f is pseudorandom for an algorithm from a class $\mathcal{C}$ on input z, it suffices to assume that f is hard to approximately print by algorithms running in

[16] To be more accurate, for every potential distinguisher M there exists a "reconstruction" algorithm F_M such that the following holds: On every z on which F_M fails to compute the hard function, the targeted PRG with input z is pseudorandom for M on input z.

[17] The meaning of "approximately print" here is as in Assumption 13.1.6: A probabilistic algorithm M approximately prints $f(x)$ with error δ if $\Pr[M(x)_i = f(x_i)] \geq 1-\delta$, where the probability is over the random coins of M and an output index. For simplicity, we think of δ as a small constant for now.

time $T' \cdot n^{o(1)}$ with oracle access to $\mathcal{C}$. (See Theorem 13.7.1 for a precise statement.)

Our goal will be to use G_f in order to replace the random coins of the verifier by pseudorandom coins. If we are able to reduce the number of coins to (say) $o(\log(n))$, then the verifier can just ask the prover in advance to send all possible $n^{o(1)}$ transcripts of interaction, and then check for consistency and compute the probability that it would have accepted (see, *e.g.*, the proof of Theorem 13.7.5 for details). However, a naive application of G_f to the common input x to the protocol does not seem to suffice for this purpose: This is because in any round of interaction, when we replace the verifier's random coins by pseudorandom coins, the verifier's behavior depends not only on x but also on the messages sent by the prover. [18]

The main idea: Using transcripts as a source of hardness. The main idea that underlies our constructions is using the *transcript of the interaction in each round as a source of hardness* for G_f. That is, in each round of interaction we apply the targeted generator G_f with the current *transcript as input*, and obtain coins that the verifier can use in that specific round, given the previous interaction.

Recall, however, that we do not assume that f is hard on all inputs, and thus an all-powerful prover could potentially find transcripts on which f is easy (in which case G_f is not guaranteed to be pseudorandom). This is where our relaxation of the soundness condition comes in: Since we are only interested in soundness with respect to polynomial-time provers and polynomial-time samplable inputs, *we can think of the transcript as an input (to G_f) sampled from a polynomial-time samplable distribution*. By our assumption f cannot be computed with non-neglibile probability over inputs chosen from *any* polynomial-time samplable distribution, and thus for all but a negligible fraction of transcripts G_f will be pseudorandom.

We stress that the function f is hard for algorithms running in fixed polynomial time $T' \cdot n^{o(1)}$ (this models the verifier), but its hardness holds over inputs (*i.e.*, transcripts) chosen according to any polynomial-time samplable distribution, where the latter polynomial may be arbitrarily large (this models an input chosen from a polynomial-time samplable distribution and the prover's corresponding messages).

Materializing this approach turns out to be considerably more subtle than it might seem. We thus include a self-contained proof for an easy "warm-up" case, namely that of derandomizing doubly efficient proof systems with one prover message (similar to MA). In this setting, if we are

[18] Another way to see this is to think of a worst-case verifier that *tries* to distinguish the pseudorandom strings from random ones. In this case, the prover's messages can be thought of as supplying this worst-case verifier with non-uniform advice.

willing to assume that one-way functions exist, then we can reduce the number of random coins to be n^{ε} (for an arbitrarily small $\varepsilon > 0$) and carry out the strategy above quite easily. See Section 13.7.1 for details.

The key step: Proof of Theorem 13.1.7. Let us start with Theorem 13.1.7, in which we derandomize protocols with c rounds in which the verifier uses $n^{o(1)}$ random coins, under the assumption that f is hard to approximately print for algorithms with oracle access to AM protocols with c rounds. Recall that we are interested in algorithms with perfect completeness, and thus when replacing random coins by pseudorandom ones we only need to argue that soundness is maintained. (For protocols with few random coins this can be assumed without loss of generality; see Remark 13.7.13.)

In each round we feed the current transcript to G_f and thus reduce the number of coins to $o(\log(n))$. Naturally, we want to use a hybrid argument, and claim that if in any round the residual acceptance probability of the verifier (when considering the continuation of the interaction after this round) significantly increased,[19] then we can compute the hard function f with the transcript at that round as input. Note that our "distinguisher" for G_f in this case is the function that computes the acceptance probability of the residual verifier after fixing the current transcript.

When this interaction happens with an all-powerful prover, the distinguisher for G_f can be computed by an AM protocol with at most c turns, and we can indeed use our hardness hypothesis. However, when we consider the acceptance probability with respect to efficient provers, then the distinguisher will be an argument system (rather than an AM protocol). This is a challenge for us (rather than an advantage), because we want to use this distinguisher to contradict the hardness of f – but the class of argument systems is broader than AM, so this will necessitate using a stronger hardness hypothesis (*i.e.*, against algorithms with oracle access to argument systems).

To handle this challenge we use a careful hybrid argument: In each round we replace not only random coins by pseudorandom ones, but also gradually replace all-powerful prover strategies by efficient prover strategies; that is, in the i^{th} hybrid we replace the random coins in the i^{th} round by pseudorandom coins, and replace the all-powerful prover strategy in the i^{th} round by an efficient strategy. Assuming that the common input x is a NO instance, the first hybrid (with random coins and an all-powerful prover) has low acceptance probability, and we are interested in analyzing

[19] By "acceptance probability" here we mean the maximal probability that the verifier accepts, when considering interaction with a prover (in the relevant class of provers) that maximizes this probability. In some sources this is referred to as the value of the game/protocol.

the case where the last hybrid (with pseudorandom coins and an efficient prover) has higher acceptance probability, in which case there is a noticeable increase in the acceptance probability between a pair of hybrids, say in the i^{th} hybrid.

In our analysis, we now further replace the efficient prover strategy in the i^{th} round by an all-powerful prover strategy; crucially, this only causes an *increase* in the acceptance probability gap.[20] At this point the "residual protocol" in both distributions that were obtained in the i^{th} hybrid uses an all-powerful prover, and (with some work) we can show that there exists a distinguisher for G_f (with the transcript as input) that is an AM protocol with c rounds. (For further details see the part titled "Obtaining an $\mathsf{AMTIME}^{[\rightleftharpoons c]}$ distinguisher" in the proof of Claim 13.7.7 for details.)

The general case: Proof of Theorem 13.1.8. The argument above works under the assumption that the protocol uses $n^{o(1)}$ coins, and we now want to extend it to general protocols. To do so we use an additional assumption, namely that there exist functions whose truth-tables can be verified in near-linear time, but that are hard for AM protocols with $c+1$ rounds that run in time $2^{(1-\delta)\cdot n}$ and use $2^{(1-\delta)\cdot n}$ bits of non-uniform advice. We will use this additional assumption to reduce the number of random coins in each round from $\mathrm{poly}(n)$ to $O(\log(n))$, using the reconstructive PRG $(\mathsf{Gen}, \mathsf{Rec})$, and then invoke Theorem 13.1.7 as a black-box.

The argument here follows in the same spirit as the one above, first replacing the coins in all rounds simultaneously and then using a careful hybrid argument (and a reconstruction argument) to obtain a contradiction. To support the setting that is obtained via the hybrid argument, we refine the reconstructive PRG $(\mathsf{Gen}, \mathsf{Rec})$ yet again, this time showing that it works when the distinguisher is any function that agrees with a certain promise problem, where the advice to the reconstruction algorithm depends only on the promise problem rather than on the function. (See Section 13.7.3.)

13.2.4 Proof of Theorem 13.1.3

Finally, we briefly explain the ideas in the proof of Theorem 13.1.3. The most important change, compared to the proofs of Theorem 13.1.1 and Theorem 13.1.2, is that since now we are only assuming hardness against *uniform* protocols (our hardness assumption is that $L \notin$ i.o.-$(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}^{[\rightleftharpoons 7]}[2^{(1-\delta)\cdot n}]$), we need to make the reconstruction algorithm uniform as well.

[20] This is the place in the proof where we use the fact that the protocol has perfect completeness (*i.e.*, we are only arguing that soundness is maintained given pseudorandom coins).

We will use yet another refinement of $(\mathsf{Gen}, \mathsf{Rec})$ above. Recall that Rec (after our last refinement) gets the "right" advice $\mathtt{adv}$, and can then compute the function f if the distinguisher "proves" to the verifier that sufficiently many queries are 1-instances. Following [MV05, GSTS03, SU07], we strengthen Rec so that it meets a resiliency condition: Namely, using a small number of rounds of interaction, Rec is able to get any prover to send advice $\mathtt{adv}$ and *commit* to a single truth-table $f_{\mathtt{adv}}$ specified by the advice.[21] That is, given $\mathtt{adv}$ and a few rounds of interaction, there is a single $f_{\mathtt{adv}}$ such that Rec answers according to it. Working carefully, we are able to make Rec resilient without significant time overhead (see Definition 13.5.1 and Proposition 13.5.2).

If our reconstruction could be convinced that $f_{\mathtt{adv}}$ to which the prover committed is the "right" truth-table (*i.e.*, $f_{\mathtt{adv}} = f$), then on query x it could simply output $f_{\mathtt{adv}}(x)$. But what if the prover committed to a truth-table different than f? To resolve this issue we initially encode f using a *highly efficient PCP of proximity* (*i.e.*, the one of [BGH+05]), which then allows us to locally test the encoded string without incurring significant time overheads and deduce that $f_{\mathtt{adv}}$ is close to f, otherwise we reject (see Theorem 13.3.16); we combine this with local error correction, to compute f given any truth-table that is close to f. The price that we pay for using the PCPP is that the truth-table (which is now a PCPP witness of the original truth-table) is necessarily of size $T^{1.01}$ (*i.e.*, slightly larger than the time complexity of the function), and thus our PRG uses $(1.01) \cdot \log(T)$ seeds and we get derandomization in quadratic time $T^{2.01}$ rather than in time $n \cdot T^{1.01}$.

13.3 Preliminaries

Throughout this chapter, we will typically denote random variables by boldface, and will denote the uniform distribution over $\{0,1\}^n$ by $\mathbf{u}_n$ and the uniform distribution over a set $[n]$ by $\mathbf{u}_{[n]}$. Recall the following standard definition of a distinguisher for a distribution $\mathbf{w}$, by which we (implicitly) mean a distinguisher between $\mathbf{w}$ and the uniform distribution.

Definition 13.3.1 (distinguisher). *We say that a function $D\colon \{0,1\}^n \to \{0,1\}$ is an ε-*distinguisher *for a distribution $\mathbf{w}$ over $\{0,1\}^n$ if $\Pr[D(\mathbf{w}) = 1] \notin \Pr[D(\mathbf{u}_n) = 1] \pm \varepsilon$. We say that D is an (α, β)-*distinguisher *if $\Pr[D(\mathbf{w}) = 1] \geq \alpha$ and $\Pr[D(\mathbf{u}_n) = 1] \leq \beta$.*

We also fix a standard notion of "nice" time bounds for complexity classes, where we are only concerned of time bounds that are not sub-linear.

[21] To see the challenge, assume that $\mathtt{adv}$ is a nondeterministic circuit that supposedly computes the truth-table. In this case, different nondeterministic guesses could yield different truth-tables.

Definition 13.3.2 (time bound)**.** *We say that $T\colon \mathbb{N} \to \mathbb{N}$ is a* time bound *if T is time-computable and non-decreasing, and for every $n \in \mathbb{N}$ we have that $T(n) \geq n$.*

Recall that $\mathsf{prMATIME}[T]$ denotes the class of *promise problems* (rather than languages) that can be solved by MA protocols with a verifier running in time T. Derandomization of $\mathsf{prMATIME}[T]$ as in the conclusions of Theorem 13.1.1 and Theorem 13.1.2 is stronger than derandomization of $\mathsf{MATIME}[T]$.

13.3.1 Useful Properties

Following Razborov and Rudich [RR97], we now define useful properties, which are sets of truth-tables that can be efficiently recognized and that describe functions hard to compute in a certain class $\mathcal{C}$. Our definition is a bit more careful than usual, since we are interested in the case where $\mathcal{C}$ is a class decidable by uniform machines that gets non-uniform advice (rather than a class decidable by non-uniform circuits).

Definition 13.3.3 (useful property)**.** *Let $\mathcal{L} \subseteq \{0,1\}^*$ be a collection of strings such that every $f \in \mathcal{L}$ is of length that is a power of two, and let $\mathcal{C}$ be a class of languages. We say that $\mathcal{L}$ is a $\mathcal{C}'$-*constructive property useful against *$\mathcal{C}$ if the following two conditions hold:*

1. *(Non-triviality.) For every $N = 2^n$ it holds that $\mathcal{L}_n = \mathcal{L} \cap \{0,1\}^N \neq \emptyset$.*
2. *(Constructivity.) $\mathcal{L} \in \mathcal{C}'$.*
3. *(Usefulness.) For every $L \in \mathcal{C}$ and every sufficiently large $n \in \mathbb{N}$ it holds that $L_n \notin \mathcal{L}_n$, where $L_n \in \{0,1\}^{2^n}$ is the truth-table of L on n-bit inputs.*

To clarify the meaning of the "usefulness" condition above, let us consider the case where $\mathcal{C}$ is a class of Turing machines with advice of bounded length. In this case, the condition asserts that for every fixed machine M and infinite sequence adv of advice strings, and every sufficiently large $n \in \mathbb{N}$, the machine M with advice adv fails to compute any truth-table in $\mathcal{L}_n$.

Since each string in $\mathcal{L}$ is of length 2^n for some $n \in \mathbb{N}$, and we think of it as a truth-table of a function over n bits, we will usually denote the input length to $\mathcal{L}$ as $N = 2^n$. For example, when we refer to an $\mathsf{NTIME}[N^2]$-constructive property useful against $\mathsf{NTIME}[2^{1.99 \cdot n}]$ we mean that 2^n-length truth-tables in $\mathcal{L}$ can be recognized in nondeterministic time 2^{2n}, but that the corresponding n-bit functions cannot be computed in nondeterministic time $2^{1.99 \cdot n}$.

13.3.2 Proof Systems

The following definition refers to nondeterministic unambiguous computation, which captures nondeterministic computation of both the language and its complement:

Definition 13.3.4 (nondeterministic unambiguous computation)**.** *We say that a machine M is* nondeterministic and unambiguous *if for every $x \in \{0,1\}^*$ there exists a value $L(x) \in \{0,1\}$ such that the following holds:*

1. *There exists a nondeterministic guess π such that $M(x,\pi) = L(x)$.*
2. *For every nondeterministic guess π' it holds that $M(x,\pi') \in \{L(x), \perp\}$.*

Arthur-Merlin proof systems

Let us now recall the definition of Arthur-Merlin proof systems (*i.e.*, of AM). Since we will be concerned with precise time bounds, we also specify the precise structure of the interaction, as follows.

Definition 13.3.5 (Arthur-Merlin proof systems)**.** *We say that $L \in \mathsf{AMTIME}^{[\rightleftharpoons c]}[T]$ if there is a proof system in which on a shared input $x \in \{0,1\}^*$, a verifier interacts with a prover, taking* turns *in sending each other information, such that the following holds.*

- **Public coins:** *Whenever the verifier sends information to the prover, that information is just uniformly chosen bits.*
- **Structure of the interaction:** *The number of turns is c, and we always assume that the first turn is the verifier sending random bits to the prover.*
- **Running time:** *The number of bits that are sent in each turn is exactly $T(|x|)$, and in the end the verifier performs a deterministic linear-time computation on the transcript (which is of length $c \cdot T(|x|) = O(T(|x|))$ and outputs a single bit.*
- **Completeness and soundness:** *For every $x \in L$ there exists a prover such that the verifier accepts with probability 1; for every $x \notin L$ and every prover, the verifier rejects, with probability at least $2/3$.*

Furthermore, if the verifier sends at most $R(n)$ random coins in each round, then we say that $L \in \mathsf{AMTIME}^{[\rightleftharpoons c]}[T,R]$. When L can be decided by an interaction as above in which the prover *takes the first turn, we say that $L \in \mathsf{MATIME}^{[\rightleftharpoons c]}[T]$. In all the definitions above, when omitting the number of messages c, we mean that $c = 2$.*

Following standard conventions, we will sometimes refer to the verifier as Arthur and to the prover as Merlin. Note that when c is odd (meaning that the last turn is the verifier's), in the last

turn the verifier does not need to send any random bits to the verifier, but may run a randomized linear-time computation on the transcript rather than a deterministic one.[22]

When we want to refer to proof systems with *imperfect completeness* (*i.e.*, for any $x \in L$ there is a prover such that the verifier accepts with probability at least $2/3$), we explicitly add a subscript "$_2$", for example $\mathsf{AMTIME}_2^{[\rightleftharpoons c]}[T]$ or $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}_2^{[\rightleftharpoons c]}[T]$.

Doubly efficient proof systems and deterministic argument systems

As mentioned in Section 13.1.3, our derandomization results will apply to proof systems in which the honest prover is efficient. We define this notion as follows:

Definition 13.3.6 (doubly efficient proof systems)**.** *We say that $L \in \mathsf{delP}^{[\rightleftharpoons c]}[T]$ if it meets all the conditions in Definition 13.3.5, and in addition meets the following condition: There exists a deterministic algorithm P running in time polynomial in T such that for every $x \in L$, when the verifier interacts with P on common input x, it accepts with probability 1. We define $\mathsf{delP}^{[\rightleftharpoons c]}[T, R]$ analogously.*

Note that Definition 13.3.6 focuses on doubly efficient proof systems with *public coins*; the known constructions of doubly efficient proof systems all use public coins (see [GKR15, RRR21, GR18, Gol18]). In addition, one could use a broader definition that allows the honest prover to run in time $\bar{T} \gg T$ that is not necessarily polynomial in T; in this work the narrower definition will suffice for us.

We will also be interested in a natural subclass of doubly efficient proof systems, which we now define. Loosely speaking, we say that a system has an efficient universal prover if for any partial transcript, the maximum acceptance probability of the residual protocol across all provers is (approximately) attained by an *efficient* prover. That is:

Definition 13.3.7 (universal provers)**.** *Let $L \in \{0,1\}^*$ and let V be a verifier in a proof system for L. For $\mu \colon \mathbb{N} \to [0,1)$, we say that the proof system has a μ-*approximate universal prover *with running time $\bar{T}$ if there exists an algorithm P that on any input x and π, where π is a partial transcript for the proof system, runs in time $\bar{T}(|x|)$, and satisfies that*

$$\Pr\left[\langle V, P\rangle(x,\pi) = 1\right] > \max_{\bar{P}}\left\{\Pr\left[\langle V, \bar{P}\rangle(x,\pi) = 1\right]\right\} - \mu(|x|)\,,$$

[22] This is equivalent to the definition above, in which the verifier sends random coins in the last turn then runs a deterministic linear-time computation on the transcript (which includes these random coins).

where the notation $\langle V, P\rangle(x, \pi)$ denotes the outcome of interaction of V and P on input x and when the first part of the transcript is fixed to π, and the maximum on the RHS is over all prover strategies (regardless of their efficiency).

Note that given $x \in L$, the universal prover acts as an honest prover that convinces the verifier to accept with probability at least $2/3 - \mu$ (or $1 - \mu$, if the protocol admits perfect completeness). In particular, a proof system with an efficient universal prover is a doubly efficient proof system. However, we do not think of the universal prover as an honest prover, since given $x \notin L$ or a dishonest partial transcript, the universal prover still tries to maximize the acceptance probability of the verifier.

A well-known doubly efficient proof system that has a universal prover is the *sumcheck protocol*; see Theorem 13.3.17 for details.

Let us now recall Definition 13.1.5 of deterministic doubly efficient argument systems and discuss a few definitional issues.

Definition 13.3.8 (deterministic doubly efficient argument system; Definition 13.1.5, restated)**.** *We say that $L \subseteq \{0,1\}^*$ is in $\mathsf{NARG}[T]$ if there exists a deterministic T-time verifier V such that the following holds:*

1. *There exists a deterministic algorithm P that, when given $x \in L$, runs in time $\mathrm{poly}(T)$ and outputs π such that $V(x, \pi) = 1$.*
2. *For every polynomial p, and every probabilistic algorithm $\tilde{P}$ running in time $p(T)$, and every sufficiently large $n \in \mathbb{N}$, the probability that $\tilde{P}(1^n)$ prints $x \notin L$ and $\pi \in \{0,1\}^T$ such that $V(x, \pi) = 1$ is $T(n)^{-\omega(1)}$.*

Observe that $\mathsf{NARG}[T] \subseteq \mathsf{DTIME}[\mathrm{poly}(T)]$, since the honest prover runs in time $\mathrm{poly}(T)$. Removing the efficiency restriction on the honest prover makes the definition too broad to be meaningful: Under a plausible hardness assumption, *every* language (regardless of is complexity) has a proof system as above in which the honest prover P runs in time larger than that of the adversaries $\tilde{P}$. [23]

Thus, for the definition to be meaningful we need to have $T = T_V < T_P < T_{\tilde{P}}$, where the three latter notations represent the running times of the verifier V, of the honest prover P, and

[23] To see this, assume that there exists a relation $R = \{(x, \pi)\}$ that can be decided in time T, but every probabilistic algorithm $\tilde{P}$ getting input x and running in time $\mathrm{poly}(T)$ fails to find π such that $(x, \pi) \in R$, except with negligible probability (over x and over the random coins of $\tilde{P}$). Then, for every L, define V that accepts (x, π) iff $(x, \pi) \in R$, and observe that this verifier meets the relaxed definition of NARG.

of the adversaries $\tilde{P}$, respectively. While Definition 13.3.8 couples these bounds so that they are all polynomially related, a broader definition in which the gaps are super-polynomial still makes sense. We use the narrower definition only because it suffices for our purposes in this chapter.

13.3.3 Error-correcting Codes

We recall the definition of locally list-decodable codes, and state a standard construction that we will use.

Definition 13.3.9 (locally list decodable codes)**.** *We say that* $\mathsf{Enc}\colon \Sigma^N \to \Sigma^M$ *is* locally list-decodable from agreement ρ in time t and with output-list size L *if there exists a randomized oracle machine* $\mathsf{Dec}\colon [N] \times [L] \to \Sigma$ *running in time* t *such that the following holds. For every* $z \in \Sigma^M$ *that satisfies* $\Pr_{i\in[M]}[z_i = \mathsf{Enc}(x)_i] \geq \rho$ *for some* $x \in \Sigma^N$ *there exists* $a \in [L]$ *such that for every* $i \in [N]$ *we have that* $\Pr[\mathsf{Dec}^z(i,a) = x_i] \geq 2/3$*, where the probability is over the internal randomness of* Dec.

Theorem 13.3.10 (a locally list-decodable code, see [STV01])**.** *For every constant* $\eta > 0$ *there exists a constant* $\eta' > 0$ *such that the following holds. For every* $m \in \mathbb{N}$ *and* $\rho = \rho(m)$ *there exists a code* $\mathsf{Enc}\colon \{0,1\}^m \to \Sigma^{\bar{m}}$*, where* $|\Sigma| = O(m^{\eta'}/\rho^2)$ *and* $\bar{m} = O_{\eta'}(m/\rho^{2/\eta'})$*, such that:*

1. *The code is computable in time* $\tilde{O}(\bar{m} \cdot \log(|\Sigma|)) = \tilde{O}(m/\rho^{2/\eta'})$.
2. *The code is locally list-decodable from agreement* ρ *in time* $m^{\eta} \cdot (1/\rho)^{1/\eta'}$ *and with output list size* $O(1/\rho)$*. Furthermore, the local decoder issues its queries in parallel, as a function of the randomness and the input.*

We also need the following two uniquely decodable codes:

Theorem 13.3.11 (uniquely decodable code, see *e.g.*, [GLR+91] and [AB09, Section 19.4])**.** *For every constant* $\eta > 0$ *the following holds. For every* $m \in \mathbb{N}$ *there exists a code* $\mathsf{Enc}\colon \{0,1\}^m \to \{0,1\}^{\bar{m}}$*, where* $\bar{m} = \tilde{O}(m)$*, such that:*

1. *The code is computable in time* $\tilde{O}(\bar{m}) = \tilde{O}(m)$.
2. *The code is locally decodable from agreement* 0.9 *with decoding circuit size* m^{η}*. Furthermore, the decoding circuit issues its queries in parallel and outputs correctly with probability at least* $1 - 1/m$.

Lemma 13.3.12 (unique decoding for low-degree univariate polynomials; see [WB86])**.** *Let* q *be a prime power. Given* t *pairs* (x_i, y_i) *of elements of* $\mathbb{F}_q$*, there is at most one polynomial* $g\colon \mathbb{F}_q \to \mathbb{F}_q$ *of degree at most* u *for which* $g(x_i) = y_i$ *for more than* $(t+u)/2$ *pairs. Furthermore, there is a polynomial time algorithm that finds* g *or report that* g *does not exist.*

13.3.4 Near-linear-time Constructions: Extractors, Hash Functions, Cryptographic PRGs

In this section we state several known algorithms that we will use in our proofs and that run in near-linear time. The first is a pairwise-independent hash function based on convolution hashing [MNT93].

Theorem 13.3.13 (a quasilinear-time computable pairwise independent hash function; for proof see, *e.g.*, [CT21b, Theorem 3.12]). *For every $m, m' \in \mathbb{N}$ there exists a family $\mathcal{H} \subseteq \left\{ \{0,1\}^m \to \{0,1\}^{m'} \right\}$ of quasilinear-time computable functions such that for every distinct $x, x' \in \{0,1\}^m$ it holds that $\Pr_{h \in \mathcal{H}}[h(x) = h(x')] \leq 2^{-m'}$.*

The second construction is of a seeded randomness extractor that runs in linear time, which was presented by Doron *et al.* [DMOZ20] following [TSZS06, Theorem 5].

Theorem 13.3.14 (a linear-time computable extractor, see [DMOZ20]). *There exists $c \geq 1$ such that for every $\gamma < 1/2$ the following holds. There exists a strong oblivious (δ, ε)-sampler $\mathsf{Samp}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ for $\delta = 2^{n^{1-\gamma}-n}$ and $\varepsilon \geq c \cdot n^{-1/2+\gamma}$ and $d \leq (1 + c \cdot \gamma) \cdot \log(n) + c \cdot \log(1/\varepsilon)$ and $m = \frac{1}{c} \cdot n^{1-2\gamma}$ that is computable in linear time.*

The third construction is that of a cryptographic PRG that works in near-linear time. Such a PRG can be obtained assuming standard one-way functions, by starting with a fast PRG that has small stretch and then applying standard techniques for extending the output length.

Theorem 13.3.15 (OWFs yield PRGs with near-linear running time; see [CT21a, Theorem 3.4]). *If there exists a polynomial-time computable one-way function secure against polynomial-time algorithms, then for every $\varepsilon > 0$ there exists a PRG that has seed length $\ell(n) = n^\varepsilon$, is computable in time $n^{1+\varepsilon}$, and fools every polynomial-time algorithm with negligible error. Moreover, if the one-way function is secure against polynomial-sized circuits, then the PRG fools every polynomial-sized circuit.*

13.3.5 Pair Languages and PCPPs

A pair language L is a subset of $\{0,1\}^* \times \{0,1\}^*$. We say that $L \in \mathsf{NTIME}[T]$ if $L(x,y)$ can be computed by a nondeterministic algorithm in time $T(|x| + |y|)$. We also say L has stretch K for a function $K\colon \mathbb{N} \to \mathbb{N}$, if for all $(x,y) \in L$, it holds that $|y| = K(|x|)$. We will use the following PCP of proximity for pair languages by Ben-Sasson *et al.* [BGH+05]:

Theorem 13.3.16 (PCPP with short witnesses [BGH+05]). *Let $K\colon \mathbb{N} \to \mathbb{N}$ such that $K(n) \geq n$ for all $n \in \mathbb{N}$. Suppose that L is a pair language in $\mathsf{DTIME}[T]$ for some non-decreasing function $T\colon \mathbb{N} \to \mathbb{N}$ such that L has stretch K. There is a verifier V and an algorithm A such that for every $x \in \{0,1\}^n$, denoting $T = T(n + K(n))$ and $K = K(n)$:*

1. ***(Efficiency.)*** *When V is given input x and oracle access to $y \in \{0,1\}^K$ and to a proof $\pi \in \{0,1\}^{2^r}$, where $r \leq \log(T) + O(\log\log(T))$, the verifier V uses r bits of randomness, makes at most $\mathrm{polylog}(T)$ non-adaptive queries to both y and π, and runs in time $\mathrm{poly}(n, \log(K), \log(T))$.*
2. ***(Completeness.)*** *Let M be an $O(T)$-time nondeterministic machine that decides L. That is, $(x,y) \in L$ if and only if there exists $w \in \{0,1\}^{O(T)}$ such that $M((x,y),w) = 1$. For every $y \in \{0,1\}^K$ and $w \in \{0,1\}^{O(T)}$ such that $M((x,y),w) = 1$, the algorithm $A(x,y,w)$ runs in time $\tilde{O}(T)$ and outputs a proof $\pi \in \{0,1\}^{2^r}$ such that*

$$\Pr\left[V^{y,\pi}(x, \mathbf{u}_r) = 1\right] = 1 \,.$$

3. ***(Soundness.)*** *Let $Z = \left\{z \in \{0,1\}^K : (x,z) \in L\right\}$. Then, for every $y \in \{0,1\}^K$ that has Hamming distance at least $K/\log(T)$ from every $z \in Z$ and every $\pi \in \{0,1\}^{2^r}$,*

$$\Pr\left[V^{y,\pi}(x, \mathbf{u}_r) = 1\right] \leq 1/3 \,.$$

13.3.6 Constant-round Sumcheck and #NSETH

The following result is an "asymmetric" version of Williams' [Wil16b] adaptation of the sumcheck protocol [LFKN92] into a constant-round protocol for counting the number of satisfying assignments of a given formula. Specifically, while in [Wil16b] the n variables are partitioned into symmetric subsets and each sumcheck round is a summation over one of the subsets, here we partition the variables such that the first set is smaller, and in the corresponding sumcheck protocl the first round is shorter.

Theorem 13.3.17 (a constant-round protocol for counting satisfying assignments of a formula). *Let $k \in \mathbb{N}$, $\delta \in (0,1)$, and $\gamma = \frac{1-\delta}{k}$ be constants. For any $s(n) \leq 2^{o(n)}$, there is an $\mathsf{MATIME}^{[\rightleftharpoons 2k]}[2^{\max(\delta,\gamma)\cdot n + o(n)}]$ protocol Π that gets as input a formula $C\colon \{0,1\}^n \to \{0,1\}$ of size $s(n)$ and (with probability 1) outputs the number of satisfying assignments of C, such that the first prover message has length at most $2^{\delta n + o(n)}$, and the rest $k-1$ prover messages have length at most $2^{\gamma n + o(n)}$.*

Moreover, the protocol has the following two properties:

1. *After the prover sends its first message, the maximum acceptance probability of the subsequent protocol is either* 1 *or at most* $1/3$.
2. *There is a* $1/n$*-approximate universal prover running in* $2^{O(n)}$ *for* Π.[24]

Proof. The protocol is essentially identical to that of [Wil16b, Theorem 3.4]. The only difference is that in [Wil16b, Theorem 3.4] the n input variables are partitioned into $k+1$ blocks, each of length $n/(k+1)$, whereas we will partition them into a single block of length $\delta \cdot n$ and k other blocks of length $\gamma \cdot n$. Due to this fact, and since we are also claiming additional properties in the "moreover" part that were not stated in [Wil16b], we include a complete proof.

Let $s(n) \leq 2^{o(n)}$, and let $C\colon \{0,1\}^n \to \{0,1\}$ be a formula of size $s(n)$. The prover and verifier will work with a prime $p \in (2^n, 2^{n+1}]$.[25] Let $P\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be the arithmetic circuit constructed by the standard arithmetization of the Boolean circuit C (see the proof of [Wil16b, Theorem 3.3] for details) such that P has size $\mathrm{poly}(s(n)) \leq 2^{o(n)}$ and degree at most $\mathrm{poly}(s(n)) \leq 2^{o(n)}$.

For simplicity, we assume that $\delta \cdot n$ and $\gamma \cdot n$ are integers. We partition the n variables $x_1, \ldots, x_n$ into $k+1$ blocks $S_1, \ldots, S_{k+1}$, such that $|S_1| = \delta \cdot n$ and $|S_i| = \gamma \cdot n$ for every $i \geq 2$. For each $i \in [k+1]$, via interpolation similar to the proof of [Wil16b, Theorem 3.4], we define a degree-$2^{|S_i|}$ polynomial $\Phi_i\colon \mathbb{F}_q \to \mathbb{F}_q^{|S_i|}$ such that for every $j \in \{0,1,\ldots,2^{|S_i|}-1\}$, $(\Phi_i(j))_\ell$ is the ℓ-th bit in the $|S_i|$-bit binary representation of j. Using a fast interpolation algorithm (see [Wil16b, Theorem 2.2]), the polynomial Φ_i (*i.e.*, the list of the coefficients of $|S_i|$ univariate polynomials, each corresponding to one of Φ_i's output values) can be constructed in $2^{|S_i|} \cdot \mathrm{poly}(n)$ time.

The protocol Π is specified as follows:

- In the first round, the honest prover computes the coefficients of the polynomial

$$Q_1(y) = \sum_{\substack{j_2,\ldots,j_{k+1} \\ \in \{0,1,\ldots,2^{\gamma\cdot n}-1\}}} P(\Phi_1(y), \Phi_2(j_2), \ldots, \Phi_{k+1}(j_k)) \tag{13.3.1}$$

  via interpolation, and sends Q_1 to the verifier (Note that Q_1 has degree $2^{o(n)+\delta\cdot n}$, so this message has size $2^{o(n)+\delta\cdot n}$). The verifier then chooses $r_1 \in \mathbb{F}_q$ uniformly at random and sends it to the prover.
- In the t-th round for $t \in \{2,3,\ldots,k\}$, the honest prover sends the coefficients of the degree-

[24] For the definition of a universal prover, see Definition 13.3.7.

[25] The honest prover can find the smallest prime p that is larger than 2^n, which is at most 2^{n+1} by Bertrands postulate, in time $2^{O(n)}$ and send p to the verifier. The verifier checks that the received number lies in $(2^n, 2^{n+1}]$ and is a prime, in deterministic time $\mathrm{poly}(n)$ (*e.g.*, using [AKS04, AKS19]).

$2^{o(n)+\gamma\cdot n}$ polynomial

$$Q_t(y) = \sum_{\substack{j_{t+1},\ldots,j_{k+1} \\ \in\{0,1,\ldots,2^{\gamma\cdot n}-1\}}} P(\Phi_1(r_1),\ldots,\Phi_{t-1}(r_{t-1}),\Phi_t(y),\Phi_{t+1}(j_{t+1}),\ldots,\Phi_{k+1}(j_{k+1})) \quad (13.3.2)$$

to the verifier. The verifier first checks if

$$\sum_{j_t\in\{0,1,\ldots,2^{\gamma\cdot n}-1\}} Q_t(j_t) = Q_{t-1}(r_{t-1}), \quad (13.3.3)$$

and rejects immediately if the equality does not hold. The verifier then picks $r_t \in \mathbb{F}_q$ uniformly at random. The verifier then picks $r_\tau \in \mathbb{F}_q$ uniformly at random, and sends it to the prover if $\tau < k$

- Finally, at the end of the k-th round, the verifier checks if

$$Q_k(r_k) = \sum_{j_{k+1}\in\{0,1,\ldots,2^{\gamma\cdot n}-1\}} P((\Phi_1(r_1),\ldots,\Phi_k(r_k),\Phi_{k+1}(j_{k+1})),$$

and rejects immediately if the equality does not hold. Otherwise, it outputs

$$\sum_{j_1\in\{0,1,\ldots,2^{\delta\cdot n}-1\}} Q_1(j_1)\,.$$

The upper bound on message lengths and the running time of the verifier can be verified directly from the protocol above. The analysis establishing the completeness and soundness of this protocol, as well as the upper bound on the complexity of the honest prover, follow a standard analysis of the sumcheck protocol; see [Wil16b, Theorem 3.4]. We therefore focus on establishing the moreover part.

To see the "moreover" part, for every $i \in [k]$, let D_i be the degree of the polynomial Q_i. Fix a partial transcript π for all the interaction before the t-th prover message, and without loss of generality assume that $\pi = (\tilde{Q}_1, r_1, \ldots, \tilde{Q}_{t-1}, r_{t-1})$ (if $t = 1$ then π is empty). We prove the following claim.

Claim 13.3.18. *Given a partial transcript $\pi = (\tilde{Q}_1, r_1, \ldots, \tilde{Q}_{t-1}, r_{t-1})$, if $t = 1$ or $\tilde{Q}_{t-1}(r_{t-1}) = Q_{t-1}(r_{t-1})$, then the maximum acceptance probability of the subsequent protocol is 1, and can be achieved by a $2^{O(n)}$-time prover. Otherwise, it is at most $(k+1-t)/n^2$.*

Proof. We prove the claim by an induction on t, starting from $k+1$ and moving downward to 1. In the base case $t = k+1$ all messages have been sent, and the verifier accepts if and only $\tilde{Q}_{t-1}(r_{t-1}) = Q_{t-1}(r_{t-1})$. Hence the claim holds immediately.

Now, for $t \in [k]$, assuming the claim holds for $t+1$. If $t = 1$ or $\tilde{Q}_{t-1}(r_{t-1}) = Q_{t-1}(r_{t-1})$, then the prover simply sends the correct polynomial Q_t defined by (13.3.1) or (13.3.2) and proceeds as the honest prover (note that the check (13.3.3) will pass).

Otherwise, we have that $t \geq 2$ and $\tilde{Q}_{t-1}(r_{t-1}) \neq Q_{t-1}(r_{t-1})$. In this case, in order to not be rejected immediately, the prover has to send a polynomial $\tilde{Q}_t$ such that $\sum_{j_t \in \{0,1,\ldots,2^{\gamma \cdot n}-1\}} \tilde{Q}_t(j_t) = \tilde{Q}_{t-1}(r_{t-1})$. In particular, it means that $\tilde{Q}_t \neq Q_t$, where Q_t is defined by (13.3.2). Therefore, with probability at most $D_t/q \leq 1/n^2$ over the choice of r_t, it holds that $\tilde{Q}_t(r_t) = Q_t(r_t)$. By the induction hypothesis, we know that the maximum acceptance probability is at most $1/n^2 + (k-t)/n^2 \leq (k+1-t)/n^2$. □

The existence of a $1/n$-approximate universal prover with running $2^{O(n)}$ (the second item of the "moreover" part) follows immediately from Claim 13.3.18.

To see the first item, we note that if $\tilde{Q}_1 = Q_1$, where Q_1 is defined by (13.3.1), then the maximum acceptance probability is 1, by Claim 13.3.18. If $\tilde{Q}_1 \neq Q_1$, then since both of them have degree at most D_1, the probability of $\tilde{Q}_1(r_1) = Q_1(r_1)$ is at most $D_1/q \leq 1/n^2$, hence the overall acceptance probability is at most $1/n$, using the same argument as in the proof of Claim 13.3.18.

■

The "savings" in running time above (*i.e.*, the improvements over the brute-force algorithm that runs in time $2^{(1+o(1))\cdot n}$) were achieved via a probabilistic interactive protocol. The following assumption, denoted #NSETH, asserts that without randomness it is impossible to achieve running time $2^{(1-\varepsilon)\cdot n}$, for any constant $\varepsilon > 0$.

Assumption 13.3.19 (#NSETH)**.** *There does not exist a constant $\varepsilon > 0$ and a nondeterministic machine M that gets as input a formula Φ over n variables of size $2^{o(n)}$, runs in time $2^{(1-\varepsilon)\cdot n}$, and satisfies the following:*

1. *There exists nondeterministic choices such that M outputs the number of satisfying assignments for Φ.*
2. *For all nondeterministic choices, M either outputs the number of satisfying assignments for Φ or outputs $\perp$.*

Recall that the standard strong exponential-time hypotheses SETH asserts that for every $\varepsilon > 0$ it is hard to solve k-SAT with n variables in time $2^{(1-\varepsilon)\cdot n}$, where $k = k_\varepsilon$ is sufficiently large. The assumption #NSETH is incomparable to SETH: On the one hand, in #NSETH we assume hardness with respect to a larger class of formulas (*i.e.*, n-bit formulas of size $2^{o(n)}$), and also assume hardness of the *counting* problem; but on the other hand, the hardness in #NSETH is for nondeterministic machines rather than just for deterministic algorithms.

13.4 Superfast Derandomization of MA

The following is the "basic version" of the highly efficient reconstructive PRG, which was mentioned in the beginning of Section 13.2. We will further refine this version later on in Proposition 13.5.2 and Proposition 13.7.17.

Proposition 13.4.1 (a reconstructive PRG with unambiguous nondeterministic reconstruction)**.** *For every $\varepsilon_0 > 0$ there exists $\delta_0 > 0$ and a pair of algorithms that for any $N \in \mathbb{N}$ and $f \in \{0,1\}^{N^{1+\varepsilon_0/3}}$ satisfy the following:*

1. **(Generator.)** *The generator G gets input 1^N, oracle access to f, and a random seed of length $(1+\varepsilon_0)\cdot\log(N)$, and outputs an N-bit string in time $N^{1+\varepsilon_0}$.*
2. **(Reconstruction.)** *For any $D\colon \{0,1\}^N \to \{0,1\}$ such that $\Pr_{s\in[N^{1+\varepsilon_0}]}[D(G^f(1^N,s)) = 1] > \Pr_{r\in\{0,1\}^N}[D(r) = 1] + 1/10$ there exists a string $\mathtt{adv}$ of length $|f|^{1-\delta_0}$ such that the following holds. When the reconstruction R gets input $x \in [|f|]$ and oracle access to D and non-uniform advice $\mathtt{adv}$, it runs in nondeterministic time $|f|^{1-\delta_0}$, issues queries in parallel, and unambiguously computes f_x.*

Proof. For a sufficiently small $\gamma = \gamma(\varepsilon_0)$ to be determined later, let $\mathsf{Samp}\colon \{0,1\}^{\bar{N}} \times [\bar{L}] \to \{0,1\}^N$ be the sampler from Theorem 13.3.14, instantiated with parameter γ, with a a sufficiently small constant error, and with $\bar{N} = N^{1+O(\gamma)}$ and $\bar{L} = \bar{N}^{1+O(\gamma)}$.

The generator G. The generator encodes f to $\bar{f} = \mathsf{Enc}(f)$ using the code Enc in Theorem 13.3.10, instantiated with parameters $m = N$ and ρ, η that are sufficiently small constants. We think of $\bar{f}$ as a binary string (by naively encoding each symbol using $\log(|\Sigma|)$ bits, in which case $|\bar{f}| = \bar{m}\cdot\log(|\Sigma|) = \tilde{O}(N^{1+\varepsilon_0/3})$. The generator then partitions $\bar{f}$ into $L = |\bar{f}|/\bar{N}$ consecutive substrings $\bar{f}_1, \ldots, \bar{f}_L$, and given seed $(i,j) \in [L] \times [\bar{L}]$ it outputs the N-bit string $\mathsf{Samp}(\bar{f}_i, j)$. The number of

strings in the set is

$$L \cdot \bar{L} = (|\bar{f}|/\bar{N}) \cdot (\bar{N}^{1+O(\gamma)}) = N^{1+\varepsilon_0/3+O(\gamma)} < N^{1+\varepsilon_0} ,$$

and the running time of the generator is $\tilde{O}(N^{1+\varepsilon_0/3}) < N^{1+\varepsilon_0}$.

The reconstruction R**.** Fix a $(1/10)$-distinguisher $D\colon \{0,1\}^N \to \{0,1\}$. Denoting the uniform distribution over the output-set of G^f by $\mathbf{G}$, our assumption is that $\Pr[D(\mathbf{G}) = 1] > \Pr_{r\in\{0,1\}^N}[D(r) = 1] + 1/10$.

Let $\bar{D}\colon \{0,1\}^{\bar{N}} \to \{0,1\}$ be the function

$$\bar{D}(z) = 1 \iff \Pr_{j\in[\bar{L}]}[D(\mathsf{Samp}(z,j)) = 1] \leq \Pr_{r\in\{0,1\}^N}[D(r) = 1] + .01 , \tag{13.4.1}$$

and let $S = \bar{D}^{-1}(1)$ and $T = \bar{S} = \bar{D}^{-1}(0)$. [26] Note that $|\bar{S}| \leq 2^{\bar{N}^{1-\gamma}}$, by the properties of Samp. Then, by the definition of T we have that

$$\begin{aligned}\Pr[D(\mathbf{G})) = 1] &= \Pr_{i\in[L],j\in[\bar{L}]}[D(\mathsf{Samp}(\bar{f}_i, j))] \\ &\leq \Pr_i[\bar{f}_i \in T] + \Pr_{i,j}[D(\mathsf{Samp}(\bar{f}_i, j)) = 1 | \bar{f}_i \notin T] \\ &\leq \Pr_i[\bar{f}_i \in T] + \left(\Pr_{r\in\{0,1\}^N}[D(r) = 1] + .01\right) . \end{aligned} \tag{13.4.2}$$

Since $\Pr[D(\mathbf{G}) = 1] - \Pr_r[D(r) = 1] > (1/10)$, we deduce that $\Pr_i[\bar{f}_i \in T] > (1/10) - .01 > \rho$, where the last inequality is by a sufficiently small choice of ρ.

Computing a "corrupted" version of $\bar{f}$. We first construct a machine M that computes a "corrupted" version of $\bar{f}$, as follows. Let $\mathcal{H}$ be the hash family in Theorem 13.3.13, using parameters $m = \bar{N}$ and $m' = \bar{N}^{1-\gamma/2}$. We argue that:

Fact 13.4.2. *With probability at least* $1 - 2^{-\bar{N}^{1-\gamma}}$ *over* $h \sim \mathcal{H}$*, for every distinct* $z, z' \in \bar{S}$ *it holds that* $h(z) \neq h(z')$.

Proof. For every distinct $z, z' \in \bar{S}$, the probability over $h \sim \mathcal{H}$ that $h(z) = h(z')$ is at most $2^{-\bar{N}^{1-\gamma/2}}$. By a union bound over $|\bar{S}|^2 \leq 2^{2\bar{N}^{1-\gamma}}$ pairs, with probability at least $1 - 2^{-\bar{N}^{1-\gamma/2}} \cdot 2^{2\bar{N}^{1-\gamma}} > 1 - 2^{-\bar{N}^{1-\gamma}}$ there does not exist a colliding pair in $\bar{S}$. □

[26] Denoting the complement of S both by $\bar{S}$ and by T might seem unnecessarily cumbersome. However, this formulation will generalize more easily later on when we prove the "furthermore" statement.

Let $I = \{i \in [L] : \bar{f}_i \in T\}$. The machine M gets as advice the foregoing h, the set $\{(i, h(\bar{f}_i)) : i \in I\}$, and the value $\Pr_{r \in \{0,1\}^N}[D(r) = 1]$. Given $x \in [\bar{f}]$, the machine computes $i \in [L]$ such that the index x belongs to the i^{th} substring of $\bar{f}$, and if $i \notin I$ it outputs zero. Otherwise, the machine:

1. Nondeterministically guesses a preimage $z \in \{0,1\}^{\bar{N}}$ for $\bar{f}_i$ under h.
2. Verifies that $h(z) = \bar{f}_i$ using the advice value $(i, h(\bar{f}_i))$.
3. Verifies that $z \notin S$, using the oracle access to D, the sampler Samp, and the acceptance probability of D (that is given as advice).
4. If either of the two verifications failed, the machine aborts. Otherwise, it outputs the bit in z corresponding to index x.

Note that the foregoing machine computes, in an unambiguous nondeterministic manner, a string $\tilde{f}$ such that $\Pr_x[\bar{f}_x = \tilde{f}_x] \geq \rho$. (This is because for every x belonging to a substring indexed by $i \notin I$ it holds that $\tilde{f}_x = 0$, and for every other x it holds that $\tilde{f}_x = \bar{f}_x$.) The number of advice bits that M uses is at most $\tilde{O}(\bar{N} + L \cdot \bar{N}^{1-\gamma/2} + N) \leq N^{1+O(\gamma)}$, and its running time is dominated by computing the hash function once (which takes time $\tilde{O}(\bar{N})$) and computing $\bar{D}$ once.

Computing $\bar{f}$, and thus also f. Consider the execution of the local list-decoder Dec from Theorem 13.3.10 with agreement ρ, when it is given oracle access to the "corrupted" version of $\bar{f}$ computed by M (*i.e.*, the version in which a block $\bar{f}_i$ indexed by $i \in I$ has the correct values of $\bar{f}$, and all other blocks are filled with zeroes). Fixing the "right" index $\eta \in [O(1/\rho)]$ of f in the corresponding list of codewords, we reduce the error probability of Dec to less than $1/N^2$ (by $O(\log(N))$ repetitions) and now consider its execution with a fixed random string.

The reconstruction procedure R gets as advice the non-uniform advice for M, the index η, and the fixed random string for Dec (we will see that the latter string is of length $N^{O(\gamma)}$). Given $x \in [|\bar{f}|]$, it runs Dec and answers its queries using M. If any of the queries to M was answered by $\perp$, we abort and output $\perp$, and otherwise we output the result of the list-decoder's computation. Note that Dec can be described by an oracle circuit of size $\text{poly}(1/\rho) \cdot N^{\eta} + \log(1/\rho) + O(1) \leq N^{2\eta} \leq N^{O(\gamma)}$, where we relied on a sufficiently small choice of η. It issues its queries in parallel, since both Dec and the machine M issue their queries in parallel.

We thus obtained a procedure for $\bar{f}$ that is nondeterministic and unambiguous, and uses at most $N^{1+O(\gamma)} < |f|^{1-\delta_0}$ advice bits, where the inequality relies on sufficiently small choices of γ and of δ_0. Denoting the time for verifying that $z \in T$ by K, the running time of the procedure for

$\bar{f}$ is at most

$$N^{O(\gamma)} \cdot \left(\tilde{O}(\bar{N}) + K\right) = N^{1+O(\gamma)} + N^{O(\gamma)} \cdot K\,,$$

and using the naive algorithm for $\bar{D}$ (which enumerates over $i \in [\bar{L}]$) this is at most $N^{1+O(\gamma)} < |f|^{1-\delta_0}$. ■

Given the reconstructive PRG in Proposition 13.4.1, we can now prove Theorem 13.1.1:

Theorem 13.4.3 (derandomization with quadratic overhead from useful properties against SVN circuits)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists an $\mathsf{NTIME}[N^{2+\varepsilon/4}]$-constructive property $\mathcal{L}$ useful against $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, for any time bound T we have that $\mathsf{prBPTIME}[T] \subseteq \mathsf{pr}(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T^{2+\varepsilon}]$.*

Proof. Let A be a $\mathsf{prBPTIME}[T]$ machine, fix a sufficiently large input length $|x|$, and let $N = T(|x|)$. Given input x, our derandomization algorithm guesses a string f_n of length $2^n = N^{1+\varepsilon/3}$ and verifies that $f_n \in \mathcal{L}$ (if the verification fails, it aborts).[27] It then enumerates over the seeds of the generator from Proposition 13.4.1, when the latter is instantiated with $\varepsilon_0 = \varepsilon/2$ and with f_n as the oracle, to obtain $N^{1+\varepsilon/2}$ pseudorandom strings $w_1, \ldots, w_{N^{1+\varepsilon/2}} \in \{0,1\}^N$; and it outputs $\mathtt{MAJ}\{A(x, w_i)\}_{i\in[N^{1+\varepsilon/2}]}$. This derandomization algorithm runs in time $O(N^{2+\varepsilon}) = O(T(n)^{2+\varepsilon})$.

Let $\delta \leq \delta_0/2$ be sufficiently small. For any $n \in \mathbb{N}$ and $N = 2^{n/(1+\varepsilon/3)}$, assume that there is $x \in \{0,1\}^{T^{-1}(N)}$ and $f_n \in (\mathcal{L} \cap \{0,1\}^{2^n})$ such that $D_x(r) = A(x, r)$ is a $(1/10)$-distinguisher for the generator above, when the latter guesses the truth-table f_n. Let $\bar{D}_x$ be either the function computed by D_x (if D_x accepts a pseudorandom input with higher probability than it does a random input) or the negation of that function (otherwise). By Proposition 13.4.1, there is a reconstruction algorithm R for f_n that runs in time $|f_n|^{1-2\delta}$ and uses oracle access to $\bar{D}_x$ and $|f_n|^{1-2\delta}$ bits of advice. To simulate the oracle for R, we supply R with additional advice x of length $|x| = T^{-1}(2^{n/(1+\varepsilon/3)}) \leq |f_n|^{1-2\delta}$ and with an advice bit indicating whether or not to flip the output of D_x. Plugging in the time complexity of D_x as $N = |f_n|^{1/(1+\varepsilon/3)}$, the running time of R is $|f_n|^{(1-2\delta)+1/(1+\varepsilon/3)} < |f_n|^{2-\delta}$, and it nondeterministically and unambiguously computes the function whose truth-table is f_n.

Now, assume towards a contradiction that there are infinitely many $x \in \{0,1\}^*$ and f_n such that D_x is a $(1/10)$-distinguisher for the generator with f_n, and fix corresponding advice strings for R as above. (On input lengths for which there are no suitable x and f_n, the advice string indicates

[27] For simplicity we assume that $N^{1+\varepsilon/3}$ is a power of two, since rounding issues do not meaningfully affect the proof.

that R should compute the all-zero function.) This yields $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ whose truth-tables are included, infinitely often, in $\mathcal{L}$, contradicting the usefulness of $\mathcal{L}$. ■

The proof of Theorem 13.4.3 also yields the following result, in which both the hypothesis and the conclusion are stronger:

Theorem 13.4.4 (derandomization with quadratic overhead from batch-computable truth-tables)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists*

$$L \notin \textit{i.o.-}(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$$

such that there is an algorithm that gets input 1^n and prints the truth-table of L on n-bit inputs in time $2^{(2+\varepsilon/4)\cdot n}$. Then, for any time bound T we have that $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[T^{2+\varepsilon}]$.

We think of the algorithm in the hypothesis of Theorem 13.4.4 as "batch-computing" the hard function: It prints the entire truth-table in time slightly larger than 2^{2n}, whereas computing individual entries in the truth-table cannot be done in time slightly less than 2^{2n} (even using unambiguous non-determinism). The only difference between the proof of Theorem 13.4.3 and in the proof of Theorem 13.4.4 is that in the latter, instead of guessing and verifying the truth-table of a hard function, we explicitly compute the truth-table using the hypothesized algorithm.

13.5 Superfast Derandomization of AM

In this section we prove Theorem 13.1.2 and Theorem 13.1.3. Towards this purpose, in Section 13.5.1 we refine the reconstructive PRG from Proposition 13.4.1. Then, in Section 13.5.2, we conditionally construct two PRGs that rely on different hardness hypotheses and have different parameters, both of which use the foregoing refined reconstructive PRG. In Section 13.5.2 we compose the two PRGs in order to prove Theorem 13.1.2. Lastly, in Section 13.5.4 we use the reconstructive PRG in a different way to prove Theorem 13.1.3.

13.5.1 Refining the Reconstructive PRG from Proposition 13.4.1

We now extend Proposition 13.4.1 in two ways. First, we argue that by allowing randomness in the reconstruction, we can reduce its query complexity (this is along the lines of ideas from [DMOZ20, CT21b]). Secondly, we claim that the reconstruction does not need "full oracle access" to D; loosely

speaking, the functionality of the reconstruction is maintained as long as we can guarantee that D answers zero on a sufficiently large fraction of the queries.

Definition 13.5.1 (α-indicative sequences)**.** *Let $t, s \in \mathbb{N}$ such that $s|t$, let $\alpha \in (0,1)$, and let $d \in \{0,1\}^t$. We say that d is (s,α)-*valid *if for every $i \in [t/s]$ it holds that $\sum_{j\in[s]} d_{(i-1)+j} \geq \alpha \cdot s$. We say that a sequence $q \in \{0,1\}^t$ is (s,α)-*indicative of *d if for every $i \in [t/s]$ it holds that $\sum_{j\in[s]} (q_{(i-1)+j} \wedge d_{(i-1)+j}) \geq \alpha \cdot s$. We say that a sequence $q \in \{0,1\}^t$ is (s,α)-*deficient *if there exists $i \in [t/s]$ such that $\sum_{j\in[s]} q_{(i-1)+j} < \alpha \cdot s$.*

Proposition 13.5.2 (an extension of the PRG from Proposition 13.4.1)**.** *In the reconstruction of Proposition 13.4.1, for any D satisfying the hypothesis and any input x and witness w for the reconstruction procedure, denote by $\bar{R}^D(x,w) \in \{f_x, \perp\}$ the output of R with oracle access to D and advice $\mathtt{adv}$. If we allow R to use randomness, then we may assume that it makes only $\bar{t} = N^{\varepsilon_0/10}$ parallel queries to D and satisfies $\Pr[R^D(x,w) = \bar{R}^D(x,w)] \geq 2/3$.*

Furthermore, denoting $\bar{N} = N^{1+\varepsilon_0/3}$, there exists $s \in \mathbb{N}$ satisfying $s|\bar{t}$ and $\alpha \in (0,1)$ and a random variable $\mathbf{h}$ over $\{0,1\}^{\bar{N}^{1-2\delta_0}}$ that can be sampled in quasilinear time, such that for any $D\colon \{0,1\}^N \to \{0,1\}$ satisfying $\Pr_{r\in\{0,1\}^N}[D(r) = 1] \leq 1/3$, with probability at least $1 - 1/N$ over $h \sim \mathbf{h}$ the following holds. For any input $x \in [\bar{N}]$ and witness w and random coins γ, denote by $d_{x,w,\gamma} \in \{0,1\}^{\bar{t}}$ the evaluations of D on the queries made by R, and denote by $a_{x,w,\gamma} \in \{0,1\}^{\bar{t}}$ the answers that R received to these queries. Then,

1. **(Honest oracle.)** *For any $f \in \{0,1\}^{\bar{N}}$, assume that $\Pr_{s\in[N^{1+\varepsilon_0}]}[D(G^f(s)) = 1] \geq 1/2$. Then, there exists $\mathtt{adv} \in \{0,1\}^{\bar{N}^{1-2\delta_0}}$ such that when R gets advice $(h, \mathtt{adv})$ the following holds.*
   (a) *Completeness: For every x there exists w such that with probability $1 - 1/N$ over γ it holds that $d_{x,w,\gamma}$ is (s,α)-valid, and if $a_{x,w,\gamma}$ is (s,α)-indicative of $d_{x,w,\gamma}$ then $R(x,w)$ outputs f_x.*
   (b) *Soundness: For every (x,w), with probability at least $1 - 1/N$ over γ, if $a_{x,w,\gamma}$ is (s,α)-indicative of $d_{x,w,\gamma}$, then $R(x,w)$ outputs either f_x or $\perp$.*
2. **(Dishonest oracles.)** *For every $\mathtt{adv} \in \{0,1\}^{\bar{N}^{1-2\delta_0}}$ there exists $g \in \{0,1\}^{\bar{N}}$ such that when R gets advice $(h, \mathtt{adv})$ the following holds. For every (x,w), with probability at least $1 - 1/N$ over γ, if $a_{x,w,\gamma}$ is (s,α)-indicative of $d_{x,w,\gamma}$ then $R(x,w)$ outputs either g_x or $\perp$.*
3. **(Deficient oracles.)** *For every advice $(h, \mathtt{adv})$ to R and any (x,w) and any choice of γ, if $a_{x,w,\gamma}$ is (s,α)-deficient then R outputs $\perp$.*

We stress that the "dishonest oracles" claim and the "deficient oracles" claim do not depend on any particular choice of f for the generator G. That is, the two claims refer only to the behavior of the reconstruction R given advice $(h, \mathtt{adv})$ and oracle access to D such that $\Pr_{r\in\{0,1\}^N}[D(r) =$

$1] \leq 1/3$.

Proof of Proposition 13.5.2. We follow the same proof as for Proposition 13.4.1 and explain the necessary changes. Let us start with proving the bound of $N^{\varepsilon/10}$ on the number of queries when R is allowed to use randomness.

We modify the definition of $\bar{D}$ in Eq. (13.4.1). Specifically, let $\nu = \Pr_{r\in\{0,1\}^N}[D(r) = 1]$, and let $\bar{D}$ be a probabilistic procedure that gets $z \in \{0,1\}^{\bar{N}}$, uniformly samples $s = O(\log(N))$ values $i_1, \ldots, i_s \in [\bar{L}]$ and outputs 1 if and only if $\Pr_{j\in[s]}[D(\mathsf{Samp}(z, i_j)) = 1] < \nu + .005$. We also modify the definitions of S and of T, as follows:

$$S = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j\in[\bar{L}]}[D(\mathsf{Samp}(z,j)) = 1] \leq \nu + .001 \right\} ,$$

and

$$T = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j\in[\bar{L}]}[D(\mathsf{Samp}(z,j)) = 1] > \nu + .01 \right\} .$$

Note that $T \subseteq \bar{S}$, that $|\bar{S}| \leq 2^{\bar{N}^{1-\gamma}}$ (by the properties of Samp, assuming we instantiate it with a sufficiently small error), and that $\Pr_i[\bar{f}_i \in T] > \rho$ (using the exact same calculation as in Eq. (13.4.2)). Also, $\bar{D}$ accepts every $z \in S$, with probability at least $1 - 1/N^2$, and rejects every $z \in T$, with probability at least $1 - 1/N^2$.

Given these properties, the rest of the proof continues with only one change. Specifically, in the third step of the execution of M, we compute $\bar{D}(z)$ (using randomness), and continue only if the output is zero. Since any $z \in S$ will be accepted with high probability, if we continue then we are confident that $z \in \bar{S}$, and thus (given that we verified $h(z) = h(\bar{f}_i)$) we are certain that $z = \bar{f}_i$. The final procedure that uses Dec and M uses at most $\bar{t} = s \cdot N^{O(\gamma)} < N^{\varepsilon_0/10}$ queries to D.

The "furthermore" part. We partition the queries of R into $\bar{t}/s$ sets of size s in the natural way (*i.e.*, each subset corresponds to a set of s queried made by one of the executions of M).[28] The variable $\mathbf{h}$ is the choice of hash function, and as proved in Fact 13.4.2, with probability at least $1 - 2^{-\bar{N}^{1-\gamma}} > 1 - 1/N$, there are no distinct $z, z' \in \bar{S}$ such that $h(z) = h(z')$. We also modify the definition of $\bar{D}$ and S and T above to use the value $\nu = 1/3$ instead of $\nu = \Pr_{r\in\{0,1\}^N}[D(r) = 1]$, and let $\alpha = 1/3 + .005$.

[28] Recall that the queries that Dec makes to M are non-adaptive and that the queries that M makes to D are also non-adaptive.

Deficient oracles. Assume, for a moment, that all the queries that R makes to M during its execution are to indices in substrings $\bar{f}_i$ such that $i \in I$. Then, the soundness condition for deficient oracles follows immediately by the definitions of M and R: If in any execution of M less than α of its queries are answered by 1 then $\bar{D}(z) = 0$ and M aborts (in which case R outputs $\perp$).

The only gap is that some queries to M might be to indices in substrings $\bar{f}_i$ where $i \notin I$. To handle this gap we change the definition of M as follows. Whenever $i \notin I$, the original machine just outputs 0, whereas our new modified machine will nondeterministically guess $z \in \{0,1\}^{\tilde{N}}$ and verify that $z \notin S$, and only if the verification passes it outputs 0 (otherwise it aborts). This does not affect the original proof in any way, but now the soundness condition for deficient oracles holds even without the assumption that queries to M are such that $i \in I$.

Honest oracle. We now assume that D accepts a pseudorandom string of G^f with probability at least $1/2$. Observe that $\Pr_r[\bar{f}_i \in T] \geq \rho$ as in the proof of Proposition 13.4.1; to see this, use the same calculation as in Eq. (13.4.2) only replacing the original value $\Pr_{r \in \{0,1\}^N}[D(r) = 1]$ with the new value $\nu = 1/3$.

The string adv consists of the hash values $\{(i, h(\bar{f}_i) : i \in I\}$ where $I = \{i \in [L] : \bar{f}_i \in T\}$ and of the local list decoding circuit Dec. (The circuit Dec is just as in the proof of Proposition 13.4.1: We consider the execution of Dec on the corrupted codeword defined by D and h and I, use naive error-reduction, and hard-wire a good random string. This circuit is given to R as part of the advice adv.) We can assume that the advice string is of length $|f|^{1-2\delta_0}$ (rather than δ_0) by choosing the parameter δ_0 to be smaller than in Proposition 13.4.1.

Now, recall that the non-determinism w for R yields nondeterministic strings for each of the execution of M. For every execution of M on query q and with non-determinism z, with probability at least $1 - 1/N^2$, if at least $\alpha \cdot t$ of M's oracle queries are answered by 1, and D indeed evaluates to 1 on these queries, then M outputs the value $\sigma \in \{0, 1, \perp\}$ such that $\Pr[M^D(q, z) = \sigma] \geq 2/3$. By a union-bound, with probability $1 - 1/O(N)$ this happens for all queries to M.

In this case, we can think of the oracle that Dec gets access to as a fixed string y in $\{0, 1, \perp\}$. Given any input, if Dec sees gets an answer of $\perp$ from M, then it outputs $\perp$; and otherwise it outputs the answer corresponding to the unique codeword determined by y and by the list-decoding index.[29]

To prove completeness, note that for every x there exists w for which $\sigma \in \{0, 1\}$ for every pos-

[29] To be accurate, in the latter case Dec returns the answer corresponding to the string y' in which $\perp$'s are replaced with 0's.

sible query q to M. Also, for such w, all the corresponding z's will be in $\bar{S}$. Thus, with probability $1 - 1/O(N)$ over the coins of M, at least an α-fraction of the queries of M to the oracle will be to 1-instances of D. In this case, when at least an α-fraction of the received answers are 1, then Dec (and hence also R) outputs f_x.

Dishonest oracles. Fix any advice adv to R. We can assume without loss of generality that adv consists of hash values $\{(i, h_i) : i \in I\}$ where $I \subseteq [L]$ such that $|I| \geq \rho$ and of the index $\eta \in [O(1/\rho)]$ of a codeword for the local decoder Dec and a random string for Dec (otherwise R can always output $\perp$, regardless of x and w).

We partition $[L]$ into $\bar{I} = [L] \setminus I$, and $I_0 = \left\{i \in I : \exists z \in \bar{S}, h(z) = h_i\right\}$, and $I_1 = I \setminus I_0$. For every $i \in I_0$ let $z(i)$ be the unique string in $\bar{S}$ such that $h(z(i)) = h_i$. Denote by $\bar{g} \in \{0,1\}^{|\bar{f}|}$ the following string: For every $q \in [|\bar{g}|]$, let $i(q)$ be the $i \in [L]$ such that q indexes a location in $\bar{g}_i$, and let $\bar{g}(q) = \begin{cases} 0 & i(q) \in \bar{I} \cup I_1 \\ z(i)_q & i(q) \in I_0 \end{cases}$, where $z(i)_q$ is the bit in $z(i)$ corresponding to the location indexed by q. Note that $\bar{g}$ along with the index η define together a unique codeword g (*i.e.*, g is the η^{th} codeword in the list of codewords that agree with $\bar{g}$ on at least ρ indices).

Now fix any (x, w). For any query q to M with non-determinism z', if $i(q) \notin I$ then by definition M can only output either $\bar{g}(q) = 0$ or $\perp$. Also, with probability at least $1 - 1/N^2$ over γ, if at least $\alpha \cdot s$ of M's oracle queries are answered by 1 and D evaluates to 1 on these queries:

1. If $i(q) \in I_0$ then M outputs $\begin{cases} \bar{g}(q) & z' = z(i) \\ \perp & o.w. \end{cases}$.
2. If $i(q) \in I_1$ then M outputs $\perp$. (Because the oracle answers guarantee that $z' \notin \bar{S}$, and hence $h(z') \neq h_i$.)

By a union bound, with probability $1 - 1/O(N)$ the above holds for all queries that Dec makes to M. Now, consider an execution of Dec on input $x \in [|f|]$ with oracle access to $\bar{g}$ and with the index η and the randomness that R provides it. Denote by g the string such that g_x is the answer of Dec on input $x \in [|f|]$ in such an execution. Note that g depends only on D, on h and on the advice adv.

The last step is to analyze the behavior of Dec when its gets oracle access to M rather than to $\bar{g}$, and under the condition on M's random choices above and the assumption that $a_{x,w,\gamma}$ is α-indicative of $d_{x,w,\gamma}$. Recall that Dec issues its queries in parallel, and therefore it makes the same queries to M and to $\bar{g}$. We now consider two cases: If at least one query of Dec to M is answered by $\perp$, then R aborts and outputs $\perp$ (by the definition of R). Otherwise, all of the queries of Dec to

M are answered according to $\bar{g}$; in this case Dec outputs g_x. ■

13.5.2 The Superfast Derandomization Result: Basic Version

In this section we prove Theorem 13.1.2. First, in Section 13.5.2 we present two PRGs that use the refined construction from Proposition 13.5.2. Then, in Section 13.5.2 we show how to compose these PRGs to obtain our hardness vs randomness results for AM protocol.

Two PRGs that will be composed later

The first PRG, presented next, uses a truth-table that can be recognized in near-linear time, and is hard for MAM protocols running in time $2^{(1-\delta)\cdot n}$ with $2^{(1-\delta)\cdot n}$ bits of non-uniform advice, for a small constant $\delta > 0$. This PRG allows to reduce the number of coins the verifier uses in any T-time protocol to be approximately $\log(T)$. (Recall that in our final result we want the number of coins to be approximately $\log(n)$.)

Proposition 13.5.3 (the "outer PRG" – radically reducing the number of random coins)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists an $\mathsf{NTIME}[N^{1+\varepsilon/3}]$-constructive property $\mathcal{L}$ useful against $\mathsf{MAMTIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, for every time bound T it holds that $\mathsf{prAMTIME}[T] \subseteq \mathsf{AMTIME}[T^{1+\varepsilon}, (1+\varepsilon)\cdot\log(T)]$.*

Proof. Fix a problem $\Pi = (\mathsf{Y}, \mathsf{N}) \in \mathsf{prAMTIME}[T]$, and let V be a T-time verifier for Π. Given input $x \in \{0,1\}^n$, let $N = T(n)$. We guess $f_n \in \{0,1\}^{N^{1+\varepsilon/3}}$ and verify that $f_n \in \mathcal{L}$ (otherwise we abort). Consider the generator G from Proposition 13.5.2 with $\varepsilon_0 = \varepsilon$ and input 1^N oracle access to f_n, and denote its set of outputs by $s_1, \ldots, s_{N^{1+\varepsilon}} \in \{0,1\}^N$. The new verifier V' chooses a random $i \in [N^{1+\varepsilon}]$ and simulates V at input x with random coins s_i. Note that this verifier indeed runs in time $O(N^{1+\varepsilon})$.

The reconstruction argument. Assume towards a contradiction that there exists an infinite set S of pairs (x, f_n) such that $x \in \mathsf{N}$ and $\Pr_{i\in[N^{1+\varepsilon}]}[\exists w : V(x, s_i, w) = 1] \geq .5$, where the s_i's are the outputs of G on nondeterministic guess f_n. For every such pair, denote by $D_x \colon \{0,1\}^N \to \{0,1\}$ the function $D_x(z) = 1 \iff \exists w : V(x, z, w) = 1$, and note that $\Pr_{r\in\{0,1\}^N}[D_x(r) = 1] \leq 1/3$ (because $x \in \mathsf{N}$).

We design an MAM protocol for f_n as follows. Consider the reconstruction algorithm R from Proposition 13.5.2 with oracle access to D_x and with the corresponding advice string. Given input $z \in [|f_n|]$, our protocol acts as follows:

1. The prover sends non-determinism w for R.
2. The verifier simulates R using random coins, computes a set $q_1, .., .q_t$ of queries to D_x, and sends them back to the prover.
3. The prover sends t responses $w_1, \ldots, w_t$, to be used as non-determinism for D_x with queries $q_1, \ldots, q_t$.
4. (Deterministic step.) The verifier computes the values $\{d_i = V(q_i, z, w_i)\}_{i\in[t]}$. Then it simulates R with witness w and with the query answers $\{d_i\}_{i\in[t]}$, and accepts iff R accepts.

We now use the "furthermore" part of Proposition 13.5.2. By the "honest oracle" part, and our assumption about D_x, there exists $\mathtt{adv}$ and $s \in \mathbb{N}$ and $\alpha \in (0,1)$ such that for every x there exists w for which with high probability over R's random coins, the oracle queries can be answered in a manner that will be (s,α)-indicative of the (s,α)-valid sequence of answers by D_x, and whenever that happens R outputs f_x; and whenever the sequence of answers to oracle queries are (s,α)-indicative of the sequence of answers by D_x, then R outputs a value in $\{\perp, f_x\}$.

We hard-wire $\mathtt{adv}, s, \alpha$ as non-uniform advice to the MAM protocol. For every x the prover can send the "correct" w in the first step, and witnesses $w_1, \ldots, w_t$ in the third step such that $d_i = D_x(q_i)$ for all $i \in [t]$, in which case the output of R is f_x, with high probability. To establish the soundness of the protocol, observe that we always have $d_i \leq D_x(q_i)$ (*i.e.*, it is impossible that $d_i = 1$ whereas $D_x(q_i) = 0$). Then, by Proposition 13.5.2, with high probability the following holds: If the sequence of answers to the verifier's queries is (α, s)-deficient, then R outputs $\perp$; and otherwise, it means that the answers are (s,α)-indicative of the sequence of answers by D_x, in which case we are in the soundness case and the output is either f_x or $\perp$.

By hard-wiring appropriate advice for every input length n such that $(x, f_n) \in S$, the protocol above computes a language whose truth-tables are included in $\mathcal{L}$ infinitely often. (As in the proof of Theorem 13.4.3, on input lengths for which there are no suitable x and f_n, the advice string indicates that the protocol should compute the all-zero function.) The time complexity of the protocol is at most

$$O(\underbrace{|f_n|^{1-\delta_0}}_{\text{complexity of } R} + \underbrace{N^{\varepsilon/10}}_{\text{number of queries (\textit{i.e.} } t)} \cdot \underbrace{N}_{\text{complexity of } V}) < |f_n|^{1-\delta}, \tag{13.5.1}$$

where the inequality relies on a sufficiently small choice of δ; and similarly, the advice needed for this protocol is of length $|x| + |f_n|^{1-\delta_0} + O(1) < |f_n|^{1-\delta}$. This contradicts the usefulness of $\mathcal{L}$. ■

Remark 13.5.4 (handling AM with imperfect completeness)**.** Observe that the proof above can be easily adapted to yield derandomization of AM protocols that do not have perfect completeness, at the cost of strengthening the hardness assumption. Specifically, assume that f_n is hard even for MA protocols (with the specific running time and advice) that can issue *oracle queries* to $\mathsf{AMTIME}[O(n)]$. Then, given $x \in Y$ such that $\Pr[\exists w : V(x, s_i, w) = 1]$ is too low, we can define $\bar{D}_x = 1$ to be the negation of the function D_x defined in the proof above, and run the reconstruction argument with oracle access to $\bar{D}_x$, contradicting the hardness of f_n. We chose to present our result as above merely since AM with perfect completeness is a natural class to consider, and we preferred using the weaker hypothesis.

A tangent: Derandomization prBPP**.** Using the proof approach of Proposition 13.5.3, we can obtain a derandomization of prBPP (rather than of AM) in time that is optimal under #NSETH, using hypotheses that compare favorably to previous results. Specifically, we show that:

Theorem 13.5.5 (optimal derandomization of prBPP without OWFs)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that for any polynomial $T(n)$ the following holds. Assume that:*

1. *There exists $L \notin$ i.o.-$\mathsf{MATIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ and a deterministic algorithm that gets input 1^n, runs in time $2^{(1+\varepsilon/3)\cdot n}$, and prints the truth-table of L on n-bit inputs.*
2. *For a constant $k = k_T \geq 1$ there exists $L \notin$ i.o.-$\mathsf{DTIME}[2^{(k-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ and a deterministic algorithm that gets input 1^n, runs in time $2^{(k+1)\cdot n}$, and prints the truth-table of L on n-bit inputs.*

Then, $\mathsf{prBPTIME}[T] \subseteq \mathsf{prDTIME}[n \cdot T^{1+\varepsilon}]$.

The conclusion in Theorem 13.5.5 is identical to that in [CT21b], and so is the hypothesis in Item (2). The new part is that the hypothesis in Item (1) of Theorem 13.5.5 replaces the cryptographic hypothesis that one-way functions exist in [CT21b].

Proof sketch for Theorem 13.5.5. Using the hypothesis in Item (1), we mimic the proof of Proposition 13.5.3 to argue that $\mathsf{prBPTIME}[T] \subseteq \mathsf{prBPTIME}[T^{1+\varepsilon/2}, (1+\varepsilon/2)\cdot\log(T)]$, where the latter class is that of problems that can be decided in time T with $(1+\varepsilon/2)\cdot\log(T)$ random coins. (This follows since the generator in Proposition 13.5.3 can now deterministically print f_n, which is the truth-table of L on $(1+\varepsilon/6)\cdot\log(N)$ input bits; and since the reconstruction protocol does not need an additional round, because the distinguisher is now just a deterministic function.)

We then use the hypothesis in Item (2) with the Nisan-Wigderson generator, when the latter is instantiated for small output length (see Theorem 13.5.18 for a statement of the generator's

parameters). Instantiating this generator with the truth-table of L on inputs of length $(1+\Theta(\varepsilon))\cdot\log(n)$ as in [CT21b, Section 4.2] or in Proposition 13.5.19, we deduce that $\mathsf{prBPTIME}[T^{1+\varepsilon/2},(1+\varepsilon/2)\cdot\log(T)]\subseteq\mathsf{prDTIME}[n\cdot T^{1+\varepsilon}]$ ■

The inner PRG. The next PRG that we present assumes that there is a function whose truth-tables can be recognized in time approximately n^k and that is hard for MAM protocols running in time $2^{(1-\delta)\cdot k\cdot n}$ with $2^{(1-\delta)\cdot n}$ bits of non-uniform advice (again $\delta>0$ here is a small constant). Under this assumption, any protocol that uses at most linearly-many random coins can be derandomized with time overhead that is multiplicative in the input length.

Proposition 13.5.6 (the "inner PRG" – derandomizing AM protocols with few random coins)**.** *For every $\varepsilon>0$ there exists $\delta>0$ such that the following holds. Fix $k\geq 1$, and assume that there exists an $\mathsf{NTIME}[N^{k+\varepsilon}]$-constructive property $\mathcal{L}$ useful against $\mathsf{MAMTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$. Then,*

$$\mathsf{AMTIME}[n^k,n]\subseteq\mathsf{NTIME}[n^{1+(1+\varepsilon)\cdot k}]\,.$$

Proof. Fix $\Pi\in\mathsf{AMTIME}[n^k,n]$, and let V be the n^k-time verifier that uses at most n random coins. Given input $x\in\{0,1\}^n$ and witness $w\in\{0,1\}^{n^k}$, we guess $f_n\in\{0,1\}^{n^{1+\varepsilon/3}}$ and verify that $f_n\in\mathcal{L}$. Consider G from Proposition 13.5.2 with $\varepsilon_0=\varepsilon$ and input 1^n oracle access to f_n. We enumerate over the output-set $s_1,\ldots,s_{n^{1+\varepsilon}}\in\{0,1\}^n$ of G and output $\mathtt{MAJ}_i\{V(x,s_i,w)\}$. Note that this verifier indeed runs in time $O(n^{(1+\varepsilon/3)\cdot(k+\varepsilon)}+n^{1+\varepsilon}\cdot n^k)<n^{1+(1+\varepsilon)\cdot k}$.

The reconstruction argument is essentially identical to the one in the proof of Proposition 13.5.3, the only difference being its time complexity. Specifically, using a calculation analogous to Eq. (13.5.1), the time complexity of the reconstruction in our parameter setting is

$$O(\underbrace{|f_n|^{1-\delta_0}}_{\text{complexity of }R}+\underbrace{n^{\varepsilon/10}}_{\text{number of queries}}\cdot\underbrace{n^k}_{\text{complexity of }V})<|f_n|^{(1-\delta)\cdot k}\,,$$

where the inequality relies on the fact that $k+\varepsilon/10<(1+\varepsilon/3)(1-\delta)\cdot k$, using a sufficiently small choice of $\delta>0$. ■

Composing the two PRGs

By a straightforward combination of Proposition 13.5.3 and Proposition 13.5.6, we obtain the following result, which is the first conclusion in Theorem 13.1.2:

Corollary 13.5.7 (superfast derandomization of AM; the first part of Theorem 13.1.2)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that for every $k \geq 1$ there exists an* $\mathsf{NTIME}[N^{k+\varepsilon/3}]$*-constructive property useful against* $\mathsf{MAMTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$*. Then, for every polynomial T it holds that* $\mathsf{prAMTIME}[T] \subseteq \mathsf{prNTIME}[n \cdot T^{1+\varepsilon}]$*.*

Proof. Let $T(n) = n^c$. By Proposition 13.5.3 with parameter value $\varepsilon/3$ (using our hypothesis with $k = 1$), we have that $\mathsf{prAMTIME}[T] \subseteq \mathsf{prAMTIME}[T^{1+\varepsilon/3}, (1+\varepsilon/3)\cdot\log(T)]$. By Proposition 13.5.6 instantiated with parameter values $\varepsilon/3$ and $k = (1+\varepsilon/3)\cdot c$, such that $T^{1+\varepsilon/3} = n^k$, we have that $\mathsf{prAMTIME}[n^k, O(\log(n))] \subseteq \mathsf{prAMTIME}[n^{1+(1+\varepsilon/3)\cdot k}]$, and note that $n^{1+(1+\varepsilon/3)\cdot k} \leq n \cdot T^{1+\varepsilon}$. ■

To prove the "furthermore" part of Theorem 13.1.2 we combine the foregoing result with a careful application of the standard round-reduction procedure for AM protocols by Babai and Moran [BM88].

Corollary 13.5.8 (superfast derandomization of $\mathsf{AMTIME}^{[\rightleftharpoons c]}$; the "furthermore" part of Theorem 13.1.2, restated)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that for every $k \geq 1$ there exists an* $\mathsf{NTIME}[N^{k+\varepsilon/3}]$*-constructive property useful against* $\mathsf{MAMTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$*. Then, for every polynomial T and constant $c \in \mathbb{N}$,*

$$\mathsf{prAMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{prNTIME}[n \cdot T^{\lceil c/2\rceil+\varepsilon}] \,.$$

and

$$\mathsf{prMATIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{prNTIME}[T^{\lfloor c/2\rfloor+1+\varepsilon}] \,.$$

Proof. Let us first prove the claim about AM, and note that the case of $c = 2$ was proved in Corollary 13.5.7. For $c \geq 3$, we first use the standard round-reduction of Babai and Moran [BM88] to unconditionally simulate the class in AM, while carefully tracking the simulation overheads, and then we apply Corollary 13.5.7.

Specifically, denote by $\mathsf{prAMTIME}^{[\rightleftharpoons c,T']}[T]$ (resp., $\mathsf{prMATIME}^{[\rightleftharpoons c,T']}[T]$) a protocol with c turns in which the verifier runs in time T in each turn and the prover sends T' bits in each turn. We use the following result:

Proposition 13.5.9 (the round-reduction of [BM88] for AM; see, *e.g.*, [Gol08, Appendix F.2.2.1, Extension])**.** *For every constant $c \geq 3$ we have that* $\mathsf{prAMTIME}^{[\rightleftharpoons c,T']}[T] \subseteq \mathsf{prAMTIME}^{[\rightleftharpoons \min\{c-2,2\},O(T')]}[T \cdot T']$.

We can apply Proposition 13.5.9 for $c' = \lceil c/2 \rceil - 1$ times to deduce that

$$\mathsf{prAMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{AMTIME}[O(T^{\lceil c/2 \rceil})] \,,$$

and the claim follows from Corollary 13.5.7.

The case of $\mathsf{MATIME}^{[\rightleftharpoons c]}$. Let $\Pi = (\mathsf{Y}, \mathsf{N}) \in \mathsf{prMATIME}^{[\rightleftharpoons c]}[T]$ and let V be a corresponding T-time protocol. Let $V_{>1}$ be the sub-protocol of V after the first turn, which takes an input $x \in \{0,1\}^n$ to Π and a first prover message $w_1 \in \{0,1\}^{T(n)}$ as input. Note that $V_{>1}$ is an $\mathsf{AMTIME}^{[\rightleftharpoons c-1]}[O(n)]$ protocol on $n + T(n)$ bits of input.

From the definition of $\mathsf{MATIME}^{[\rightleftharpoons c]}[T]$, we have

$$\begin{cases} \exists\, w \in \{0,1\}^{T(n)} \text{ s.t. } V_{>1}(x,w) \text{ accepts with probability 1} & \text{for } x \in \mathsf{Y} \,, \\ \forall\, w \in \{0,1\}^{T(n)}\ V_{>1}(x,w) \text{ accepts with probability at most } 1/3 & \text{for } x \in \mathsf{N} \,. \end{cases}$$

Let $\bar{\Pi} = (\bar{\mathsf{Y}}, \bar{\mathsf{N}})$ such that $(x,w) \in \bar{\mathsf{Y}}$ if $V_{>1}(x,w)$ accepts with probability at least $2/3$, and $(x,w) \in \bar{\mathsf{N}}$ if $V_{>1}(x,w)$ accepts with probability at most $1/3$. Note that $\bar{\Pi} \in \mathsf{prAMTIME}^{[\rightleftharpoons c-1]}[O(n)]$, and hence (by the first part of the proof), we have that $\bar{\Pi} \in \mathsf{prNTIME}[n^{1+\lceil (c-1)/2 \rceil + \varepsilon}] = \mathsf{prNTIME}[n^{1+\lfloor c/2 \rfloor + \varepsilon}]$.

Let $M(x,w)$ be the corresponding nondeterministic machine that decides $\bar{\Pi}$. We can decide Π as follows: Given input $x \in \{0,1\}^n$, guess $w \in \{0,1\}^{T(n)}$, simulate $M(x,w)$ and accept iff M accepts. Note that for $x \in \mathsf{Y}$, there exists $w \in \{0,1\}^{T(n)}$ such that $(x,w) \in \bar{\mathsf{Y}}$, and hence $M(x,w)$ accepts on this particular w. And when $x \in \mathsf{N}$, for all $w \in \{0,1\}^{T(n)}$ we have $(x,w) \in \bar{\mathsf{N}}$, and thus $M(x,w)$ rejects on all possible w. Therefore, we have that $\Pi \in \mathsf{prNTIME}[T^{1+\lfloor c/2 \rfloor + \varepsilon}]$. ■

13.5.3 The Superfast Derandomization Result: Stronger Version

In this section we prove the stronger version of Theorem 13.1.2, which was mentioned after the original theorem's statement. Our main goal will be to relax the hypothesis that is needed for values of $k > 1$, by requiring hardness only against NTIME machines with advice, rather than against MAM protocols with advice.

To do so, we replace the "inner" generator from Proposition 13.5.6 with the Nisan-Wigderson generator, which is similar to a proof in [CT21b]. The key challenge is that the latter generator requires hardness against algorithms (with advice) that have oracle access to NTIME, whereas we only want to assume hardness against NTIME machines (with advice). To bridge this gap, we

first show, in Section 13.5.3, how to transform a truth-table that is hard for NTIME machines with advice into a truth-table that is hard for DTIME machines with advice that have oracle access to NTIME. This proof follows an argument from [SU06], but uses a more careful analysis to obtain tighter bounds on the running time. Then, in Section 13.5.3 we combine this transformation with an application of the Nisan-Wigderson generator as above to prove the stronger version of Theorem 13.1.2 (for the result statement, see Theorem 13.5.18).

A refined analysis of the "random curve reduction"

In this section it will be more convenient to work with the notion of non-uniform programs, rather than with Turing machines that take advice. A non-uniform program A on n-bit inputs with advice complexity α and running time T is a RAM program Π of description size α (*i.e.*, $|\Pi| \leq \alpha$). Given an input $x \in \{0,1\}^n$, $A(x)$ is defined to be the output of running the program Π on input x for at most T steps (if, Π does not stop, we define the output to be 0). The notation "A on n-bit inputs" means that we only care about A's outputs on n-bit inputs.[30]

We will need the notions of non-uniform single-valued programs (which are non-uniform programs analogous of NP $\cap$ coNP) and of non-uniform non-adaptive SAT-oracle program (which are non-uniform programs analogous to P with oracle access to NP), defined as follows.

Definition 13.5.10 (non-uniform SVN programs)**.** *A single-valued nondeterministic program A on n-bit inputs with advice complexity α and running time T is a nonuniform program with the same advice complexity and running time which receives two inputs: an input $x \in \{0,1\}^n$ and a second input y of length at most T, and outputs two bits: the **value** bit and the **flag** bit. A computes the function $f\colon \{0,1\}^n \to \{0,1\}$ if the following hold:*

- *For every $x \in \{0,1\}^n$ and y, if the **flag** bit of $A(x,y)$ equals 1, then the **value** bit of $A(x,y)$ equals $f(x)$.*
- *For every x, there is y such that the **flag** bit of $A(x,y)$ equals 1.*

We note that if $\alpha = T$, then a single-valued nondeterministic program is essentially a single-valued nondeterministic circuit (see [SU06]).

Definition 13.5.11 (non-uniform programs)**.** *A non-adaptive non-uniform* SAT*-oracle program A on n-bit inputs is a pair of non-uniform programs A_{pre} and A_{post}. The program A_{pre} has n-bit inputs, and an*

[30] This is the reason why we use *program* instead of *algorithm*. We want to emphasize the fact that a non-uniform program is a *non-asymptotic* object and we only care about its behaviors on a fixed-input length n; this is similar to a circuit with n-bit inputs.

input $x \in \{0,1\}^n$, it outputs queries $q_1, \ldots, q_k$. [31] *The program A_{post} receives $x \in \{0,1\}^n$ together with k bits $a_1, \ldots, a_k$ where $a_i = 1$ if and only if $q_i \in \mathsf{SAT}$, and outputs a single answer bit.*

The running time (resp. advice complexity) of A is defined as the sum of the running time (resp. advice complexity) of A_{pre} and A_{post}. We also call k the query complexity of A.

For convenience, we will assume that the SAT oracle takes the description of a formula ϕ as input, and the number of variables in ϕ is at most the description length of ϕ. In other words, to prove that a formula ϕ with m-bit description length is satisfiable, one only needs to provide a satisfying assignment with at most m bits.

The transformation. The main result that we prove in this section is a transformation of truth-tables of functions that are hard for nondeterministic programs to truth-tables of functions that are hard for SAT-oracle programs, as follows:

Theorem 13.5.12 (the "random curve reduction" with a careful analysis of overheads)**.** *There is a universal constant $c > 1$ and an algorithm $A_{\mathsf{low\text{-}d}}$ which takes an input function $f\colon \{0,1\}^n \to \{0,1\}$, a real $\varepsilon \in (0,1)$, and an integer $k \leq 2^{\varepsilon n/100}$ as input, and outputs the truth-table of another function $g\colon \{0,1\}^{(1+\varepsilon)n} \to \{0,1\}$ such that for all sufficiently large $n \in \mathbb{N}$:*

1. *$A_{\mathsf{low\text{-}d}}(f, \varepsilon, k)$ runs in $\tilde{O}(2^{(1+\varepsilon)n})$ time.*
2. *If f does not have a single-valued nondeterministic program with running time T and advice complexity α, then g does not have a non-adaptive non-uniform SAT-oracle program with running time $T \cdot (n^{1/\varepsilon} \cdot k)^{-c}$, advice complexity $\alpha - O(n^2 \cdot k)$, and query complexity k.*

To prove Theorem 13.5.12 we will need the following technical tools. First, we need the standard low-degree extension of Boolean functions, and we use the precise definition from [SU06].

Definition 13.5.13 (low-degree extension)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function, h, q be powers of 2 such that $h \leq q$ and $d \in \mathbb{N}$ such that $h^d \geq 2^n$. Let H be the first h elements from $\mathbb{F}_q$, and I be an efficiently computable injective mapping from $\{0,1\}^n \setminus H^d$. The low-degree extension of f with respect to q, h, d is the unique d-variable polynomial $\hat{f}\colon \mathbb{F}_q^d \to \mathbb{F}_q$ with degree $h-1$ in each variable, such that $\hat{f}(I(x)) = f(x)$ for all $x \in \{0,1\}^n$ and $\hat{f}(v) = 0$ for all $v \in (H^d \setminus \mathit{Im}(I))$.*

We will also consider the Boolean function of $\hat{f}$, denoted as $f_{\mathsf{bool}}\colon \{0,1\}^{d \log q + \log\log q} \to \{0,1\}$ and defined by $f_{\mathsf{bool}}(x, i) = \hat{f}(x)_i$, where $\hat{f}(x)_i$ denotes the i-th bit of the binary representation of $\hat{f}(x)$.

[31] On all inputs from $\{0,1\}^n$ the program A_{pre} outputs the same number of queries.

Since in Definition 13.5.13 we always consider finite fields $\mathbb{F}_q$ whose size is a power of 2, we can naturally encode each element in $\mathbb{F}_q$ by exactly $\log q$ bits. Hence, the mapping I can be constructed as follows: given $x \in \{0,1\}^n$, partition it into d consecutive blocks $x^{(1)}, \ldots, x^{(d)}$ each of size $\log h$ and ignore the remaining bits (note that $n \geq d \cdot \log h$), and then interpret each $x^{(i)}$ as an element of H via a natural bijection (first interpret $x^{(i)}$ as an integer from $[h]$, and then interpret the obtained integer u as the u-th element from H). Also, by standard interpolation, the truth-table of $\hat{f}$ or f_{bool} can be computed in $\tilde{O}(q^d)$ time from the truth-table of f. For simplicity, we will chose the representation of $\mathbb{F}_q$ in such that a way that for every $u \in \{0,1\}^n$, $f(u) = f_{\mathsf{bool}}(I(u), 1)$.

We also need the notion of parametric curves, and a concentration bound for functions on random curves that was proved in [SU06].

Definition 13.5.14 (parametric curves)**.** *Let q be a prime power and $f_1, \ldots, f_q$ be an enumeration of elements from $\mathbb{F}_q$. For convenience we will assume $f_1 = 0$. Given r elements $v_1, \ldots, v_r \in \mathbb{F}_q^d$ for $r \leq q$, we define the curve passing through $v_1, \ldots, v_r$ to be the unique degree $r-1$ polynomial function $c \colon \mathbb{F}_q \to \mathbb{F}_q^d$ such that $c(f_i) = v_i$ for every $i \in [r]$. We say that a curve c is* one to one *if $c(f_i) \neq c(f_j)$ for every distinct pair i, j from $[q]$.*

The following technical lemma from [SU06] asserts that any function (or set of functions) satisfy good concentration bounds on random curves. To state it, denote by $c(x, c_1, \ldots, c_r)$ the unique curve passing through $x, c_1, \ldots, c_r$, and note that $c(x, c_1, \ldots, c_r)(0) = x$ by its definition. We also use $\mathbb{F}_q^*$ to denote $\mathbb{F}_q \setminus \{0\}$. For two finite sets W, Z such that $W \subseteq Z$ and $h \colon Z \to [0,1]$, we define

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

Then, the technical lemma from [SU06] is as follows:

Lemma 13.5.15 (curves are good samplers; see [SU06])**.** *Let q be a prime power and r be an integer such that $2 \leq r < q$. For every point $x \in \mathbb{F}_q^d$, a list of k functions $h_1, \ldots, h_k \colon \mathbb{F}_q^d \to [0,1]$, and $\delta \in (0,1)$, the probability over a random choice of points $v_1, \ldots, v_r \in \mathbb{F}_q^d$ that $c(x, c_1, \ldots, c_r)$ is one to one and*[32]

$$\left| \mu_{c(x,c_1,\ldots,c_r)(\mathbb{F}_q^*)}(h_i) - \mu_{\mathbb{F}_q^d}(h_i) \right| < \delta$$

[32] We use $c(x, c_1, \ldots, c_r)(\mathbb{F}_q^*)$ to denote the set $\{c(x, c_1, \ldots, c_r)(u) : u \in \mathbb{F}_q^*\}$.

for every $i \in [k]$ is at least

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

We now turn to the actual proof of Theorem 13.5.12.

Proof of Theorem 13.5.12. We mimic the proof of [SU06, Theorem 3.2] with a more careful choice of parameters. We also keep track of the running time and non-uniformity separately, instead of a single measure of non-uniform circuit size as in [SU06].

We define $\mathsf{pw}(x) = 2^{\lceil \log x \rceil}$. That is, $\mathsf{pw}(x)$ is the smallest power of 2 that is at least x. Let $c_0 \in \mathbb{N}$ be a large enough constant to be chosen later. We first define the following parameters:

1. $r = c_0 \cdot n$.
2. $h = \mathsf{pw}(\tilde{h})$ where $\tilde{h} = \max\left(c_0 \cdot r^2 \cdot (kn)^4, (c_0 \cdot (n+3) \cdot r)^{3/\varepsilon}\right)$.
3. $d = \lceil n/\log h \rceil + 3$.
4. $q = \mathsf{pw}(\tilde{q}/2)$, where $\tilde{q} = c_0 \cdot h \cdot d \cdot r$.

Construction of the function g. Now, we define $\hat{f}\colon \mathbb{F}_q^d \to \mathbb{F}_q$ and $f_{\mathsf{bool}}\colon \{0,1\}^{d\cdot\log q + \log\log q} \to \{0,1\}$ as the low-degree extension of f with respect to q, h, d, according to Definition 13.5.13. And we define our output function g so that given an input $x \in \{0,1\}^{(1+\varepsilon)\cdot n}$, $g(x)$ computes f_{bool} on the length-$(d \cdot \log q + \log\log q)$ prefix of x.

Fact 13.5.16. *By our parameter choices we have that $(1+\varepsilon)\cdot n \geq d \cdot \log q + \log\log q$, and hence g is well-defined.*

Proof. By the definition of q, we have that $q \leq \tilde{q} = h \cdot (c_0 \cdot d \cdot r)$. By the definition of h, we have that

$$h \geq \tilde{h} \geq (c_0 \cdot (n+3) \cdot r)^{3/\varepsilon} \geq (c_0 \cdot d \cdot r)^{3/\varepsilon} \ .$$

The above further implies that $q \leq \tilde{q} \leq h^{1+\varepsilon/3}$. Hence, for a sufficiently large $n \in \mathbb{N}$, we have that

$$\begin{aligned} d \cdot \log q + \log\log q &= (\lceil n/\log h \rceil + 3) \cdot \log q + \log\log q \\ &\leq n \cdot \frac{\log q}{\log h} + 5 \cdot \log q \leq (1 + \varepsilon/3) \cdot n + 5 \cdot \log q \ . \end{aligned}$$

Next, from the definition of q and the assumption that $k \leq 2^{\varepsilon n/100}$, we have

$$\begin{aligned}\log q &\leq O(\log n) + \log h \leq O(\log n) + 4 \cdot \log k \\ &\leq O(\log n) + 4 \cdot \varepsilon/100 \cdot n \leq \varepsilon/20 \cdot n \,.\end{aligned}$$

Putting the above together, we have

$$d \cdot \log q + \log\log q \leq (1 + \varepsilon/3 + \varepsilon/4) \cdot n < (1+\varepsilon) \cdot n \,.$$

□

As mentioned after Definition 13.5.13, the truth-table of g can be computed in $\tilde{O}(2^{(1+\varepsilon)n})$ time. This proves Item (1).

Construction of a probabilistic program B' for f_{bool}. To prove Item (2), it suffices to show the following

- If f_{bool} has a non-adaptive non-uniform SAT-oracle program $A = (A_{\mathsf{pre}}, A_{\mathsf{post}})$ with running time $T \geq 1$, advice complexity $\alpha \geq 0$, and query complexity k,
- then f_{bool} has a single-valued nondeterministic program B with advice complexity $\alpha + O(n^2 \cdot k)$ and running time $O(T \cdot q + \mathrm{poly}(q))$.[33]

We will first construct a probabilistic single-valued nondeterministic program B' that computes f_{bool} and then fix its randomness to obtain the desired program that computes f (the definition of such a probabilistic program will be clear later in the proof). Using our assumption about f_{bool}, we can construct a non-adaptive non-uniform SAT-oracle program $\hat{A} = (\hat{A}_{\mathsf{pre}}, \hat{A}_{\mathsf{post}})$ with query complexity $m = k \cdot \log q$, advice complexity α, and running time $T \cdot \log q$ that computes $\hat{f} \colon \mathbb{F}_q^d \to \mathbb{F}_q$.

For $x \in \mathbb{F}_q^d$, let $Q_1(x), \ldots, Q_m(x)$ and $A_1(x), \ldots, A_m(x)$ be the queries and answers associated with $\hat{A}$, respectively, on input x. We then define $p_i = \mu_{\mathbb{F}_q^d}(A_i)$ for every $i \in [m]$ and $\delta = 1/(9m)$. The probabilistic program B' takes the α-bit advice of $\hat{A}$ together with all the $p_1, \ldots, p_m$ as advice.[34]

On an input $(x, b) \in \{0,1\}^{d \log q} \times [\log q]$, B' works as follows:

[33] By the discussion after Definition 13.5.13, we have $f(u) = f_{\mathsf{bool}}(I(u), 1)$. Hence B can be used to compute f as well with a minor overhead in the running time.

[34] Note that each p_i can be described by a single integer between 0 and q^d. Hence these m reals take $m \cdot O(d \cdot \log q) = O(m \cdot n)$ bits to store.

1. Pick $v_1, \ldots, v_r$ uniformly at random, and set $x_a = c_{x,v_1,\ldots,v_r}(a)$ for every $a \in \mathbb{F}_q$. Simulate $\hat{A}_{\mathsf{pre}}$ to compute queries $Q_i(x_a)$ for every $i \in [m]$ and $a \in \mathbb{F}_q^*$.
2. Set $n_i = \lfloor (p_i - \delta) \cdot (q-1) \rfloor$. For every $i \in [m]$, guess $z_i \in \{0,1\}^{\mathbb{F}_q^*}$ with exactly n_i ones, and T-bit strings $\{w_{i,a}\}_{a \in \mathbb{F}_q^*}$.
3. For every $i \in [m]$ and $a \in \mathbb{F}_q^*$, check that $(z_i)_a = 1$ implies that $w_{i,a}$ is a witness that query $Q_i(x_a)$ is answered positively (recall that the query is to SAT); otherwise, set the **flag** bit in the output to be 0 and halt.
4. For every $a \in \mathbb{F}_q^*$, compute $y_a = \hat{A}_{\mathsf{post}}(x_a, (z_1)_a, (z_2)_a, \ldots, (z_m)_a)$.
5. Run the algorithm from Lemma 13.3.12 on the $q-1$ pairs (f_a, y_a) with degree $u = hdr$ to obtain a polynomial $\tau \colon \mathbb{F}_q \to \mathbb{F}_q$ of degree u. Set the **flag** bit in the output to be 1. If no such τ exists, set the **value** bit in the output to be 0, and otherwise set the **value** bit to be the b-th bit of $\tau(0)$.

Analysis of the program B'. We need the following claim.

Claim 13.5.17. *For every $(x, b) \in \mathbb{F}_q^d \times [\log q]$, with probability more than $1 - \frac{2^{-n}}{2 \log q}$ over the choice of $v_1, \ldots, v_r$, the following two conditions hold:*

1. *For all guesses z_i and $w_{i,a}$ such that the **flag** bit of the output is set to 1, then the **value** bit of the output is $f_{\mathsf{bool}}(x, b)$.*
2. *There exist guesses z_i and $w_{i,a}$ such that the **flag** bit of the output is set to 1.*

Proof. Fix $x \in \mathbb{F}_q^d$. We apply Lemma 13.5.15 to show that over a random choice of points $v_1, \ldots, v_r \in \mathbb{F}_q^d$,

$$c_{(x,v_1,\ldots,v_r)} \text{ is one to one and for all } i \in [m], \left| \mu_{c(x,c_1,\ldots,c_r)(\mathbb{F}_q^*)}(A_i) - \mu_{\mathbb{F}_q^d}(A_i) \right| < \delta \tag{13.1}$$

holds with probability at least

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

We need to show the above is lower bounded by $1 - \frac{2^{-n}}{2 \log q}$. First, by our choices of q, h, d, we have

$$\frac{1}{q^{d-2}} = \frac{1}{q^{\lceil n / \log h \rceil + 1}} \leq \frac{1}{h^{\lceil n / \log h \rceil}} \cdot \frac{1}{q} \leq \frac{2^{-n}}{4 \log q}.$$

Next, note that $\frac{2r}{(q-1) \cdot \delta^2} \leq \frac{4 \cdot 9^2 \cdot rm^2}{q} < 1/2$ by our choice of q (note that $q > h > c_0 \cdot r^2 \cdot (kn)^4 >$

$c_0 \cdot r \cdot m^2$, and c_0 is sufficiently large). We then have

$$8m \cdot \left(\frac{2r}{(q-1)\cdot\delta^2}\right)^{r/2} < 8m \cdot 2^{-r/2} \leq \frac{2^{-n}}{4\log q},$$

the last inequality follows from $r = c_0 \cdot n$ for a sufficiently large $c_0 \in \mathbb{N}$, and $m \leq \log q \cdot k \leq \log q \cdot 2^{\varepsilon n/100}$. Putting everything together, we have that (13.1) holds with probability more than $1 - \frac{2^{-n}}{2\log q}$.

Next we show whenever (13.1) holds, the two items in the claim hold. For the second item, note that since (13.1) holds, for every $i \in [m]$ we know that $A_i(x_a) = 1$ for at least n_i distinct elements $a \in \mathbb{F}_q^*$. Therefore, if for every $i \in [m]$, the guess z_i is the string with exactly n_i ones in entries indexed by those a with $A_i(x_a) = 1$, then there are witnesses $w_{i,a_{a\in\mathbb{F}_q^*}}$ that pass the check together with all the z_i.

For the first item, for all guesses z_i and $w_{i,a}$ such that the **flag** bit is 1, we know that for all $i \in [m]$ and $a \in \mathbb{F}_q^*$, $(z_i)_a = 1$ implies $A_i(x_a) = 1$, and there are exactly n_i ones in z_i. On the other hand, by (13.1), for every $i \in [m]$, the number of $a \in \mathbb{F}_q^*$ such that $A_i(x_a) = 1$ is at most $\lceil (p_i + \delta)\cdot(q-1)\rceil$. Hence, we can bound the number of errors associated with query i as follows:

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

The above in particular means that for all but at most $2\delta q \cdot m$ many $a \in \mathbb{F}_q^*$, we have $(z_i)_a = A_i(x_a)$ for all $i \in [m]$, and consequently $y_a = \hat{f}(x_a)$. Letting p be the restriction $\hat{f} \circ c_{(x,v_1,\ldots,v_r)}$, from the discussions above, it follows that for at least $(q-1) - 2\delta q m = (1-2\delta m)q - 1 = \frac{7}{9}\cdot q - 1$ of the pairs (a, y_a) we have $y_a = p(a)$. Note that the degree of p is at most hdr (since $\hat{f}$ a d-variate polynomial of individual degree at most $h-1$, and we compose it with a curve of degree r). and $q \geq \tilde{q}/2 \geq c_0/2 \cdot hdr$. Since c_0 is a sufficiently large constant, we can apply Lemma 13.3.12 to compute $p(0) = \hat{f}(x)$, and output the b-th bit of $\hat{f}(x)$ as desired, which completes the proof. □

Finally, let $I\colon \{0,1\}^n \to \mathbb{F}_q^d$ be the injective function associated the low-degree extension $\hat{f}$. Let $S = \{(x,b) : x \in I(\{0,1\}^n), b \in [\log q]\}$. Since $|S| \leq 2^n \cdot \log q$, by a union bound, there exists fixed $\hat{v}_1, \ldots, \hat{v}_r$ such that B' with randomness fixed to $\hat{v}_1, \ldots, \hat{v}_r$ correctly computes f_{bool} on all inputs from S. We then define the single-valued nondeterministic program B by fixing the randomness of B' to $\hat{v}_1, \ldots, \hat{v}_r$.

Verifying the complexity of B. We have already established that B computes f_{bool}. It remains to verify the running time and advice complexity of B. First, B takes the following advice:

1. Advice for $\hat{A}$, of length α.
2. Description of the reals $p_1, \ldots, p_m \in [0,1]$, of total length $O(m \cdot n)$.
3. Fixed randomness $\hat{v}_1, \ldots, \hat{v}_r$, which takes $d \log q \cdot r \leq O(n^2)$ bits.

Hence, the number of advice bits of B is $\alpha + O(mn) + O(n^2) \leq \alpha + O(n^2 \cdot k)$.

Finally, the running time of B is dominated by the simulation of $(q-1)$ calls to $\hat{A}$, the final decoding algorithm, and the time to compute the curve $c(x, \hat{v}_1, \ldots, \hat{v}_r)$ by direct interpolation (which takes $\mathrm{poly}(r,q) = \mathrm{poly}(q)$ time). The total running time of B can thus be bounded by

$$O(q \cdot T) + \mathrm{poly}(q) \leq T \cdot \mathrm{poly}(q) \; .$$

and the final bound follows since $q \leq n^{8/\varepsilon} \cdot k^4$. ■

An inner PRG relying on weaker hypotheses

We will use the following version of the Nisan-Wigderson [NW94] generator, when it is combined with the locally decodable code of Sudan, Trevisan, and Vadhan [STV01] and with the weak designs of Raz, Reingold, and Vadhan [RRV02]. This version of the generator is instantiated for a sufficiently small output length, and we carefully bound its running time, the running time of the reconstruction argument, and the number of advice bits that the reconstruction argument needs.

Theorem 13.5.18 (NW generator for small output length)**.** *There exists a universal constant $c_{\mathsf{NW}} > 1$ such that for all $\varepsilon_{\mathsf{NW}} > 0$ and $\mu_{\mathsf{NW}} > c_{\mathsf{NW}} \cdot \sqrt{\varepsilon_{\mathsf{NW}}}$ there exist two algorithms that for any $N \in \mathbb{N}$ and $f \in \{0,1\}^N$ satisfy the following:*

1. **(Generator.)** *When given input 1^N and oracle access to f, the generator G runs in time $N^{1+2c_{\mathsf{NW}}^2 \cdot \mu_{\mathsf{NW}}}$ and outputs a set of strings in $\{0,1\}^{N^{\varepsilon_{\mathsf{NW}}}}$.*
2. **(Reconstruction.)** *For any $(1/N^{\varepsilon_{\mathsf{NW}}})$-distinguisher $D\colon \{0,1\}^{N^{\varepsilon_{\mathsf{NW}}}} \to \{0,1\}$ for $G(1^N)^f$ there exists a string $\mathtt{adv}$ of length $N^{1-\mu_{\mathsf{NW}}}$ such that the following holds. When the reconstruction R gets input $x \in [|f|]$ and oracle access to D and non-uniform advice $\mathtt{adv}$, it runs in time $N^{c_{\mathsf{NW}} \cdot \sqrt{\varepsilon_{\mathsf{NW}}}}$, makes non-adaptive queries to D, and outputs f_x.*

Proof Sketch. The proof is the standard analysis of the NW generator as in [STV01, RRV02]; specifically, we follow the proof in [CT21a, Appendix A.2] and explain the necessary changes.

(These changes are needed since in the current statement we decouple μ_{NW} and $\varepsilon_{\mathsf{NW}}$, instead of choosing a fixed value for μ_{NW}.)

Denote $\varepsilon = \varepsilon_{\mathsf{NW}}$ and $\mu = \mu_{\mathsf{NW}}$ and $M = N^{\varepsilon}$. Instead of using designs with $\log(\rho) = (1-3\beta)\cdot\ell$ we use designs with $\log(\rho) = (1-2\mu)\cdot\ell$. Then, the seed length of the generator becomes $(1+O(\beta+\mu))\cdot\log(N) = (1+O(\mu))\cdot\log(N)$. In the reconstruction, the oracle machine P computing a corrupted version of f runs in time $\mathrm{poly}(M) = N^{O(\varepsilon)}$ and uses $M\cdot N^{(1+\beta)\cdot(1-2\mu)} = N^{1+2\beta-2\mu}$ bits of non-uniform advice. Thus, the final reconstruction uses $N^{1-\mu}$ bits of advice and runs in time $N^{O(\sqrt{\varepsilon})}$. ■

We now use the NW generator above along with Theorem 13.5.12 to replace the "inner" PRG from Proposition 13.5.6 with a PRG that relies only on lower bounds against NTIME machines with advice, rather than against MAM protocols with advice.

Proposition 13.5.19 (the "inner PRG" – derandomizing AM protocols with few random coins)**.** *For every $\varepsilon > 0$ there exist $\delta, \eta > 0$ such that the following holds. Fix $k \geq 1$, and assume that there exists an $\mathsf{NTIME}[N^{k+\varepsilon/3}]$-constructive property $\mathcal{L}$ useful against $\mathsf{NTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$. Then,*

$$\mathsf{prAMTIME}[n^k, n^\eta] \subseteq \mathsf{prNTIME}[n^{1+(1+\varepsilon)\cdot k}] \,.$$

Proof. Let $\eta > 0$ be a sufficiently small constant to be determined later, let $\Pi = (\mathsf{Y},\mathsf{N}) \in \mathsf{prAMTIME}[n^k, n^\eta]$, and let V be an n^k-time verifier for Π that uses at most n^η random coins. Given $x \in \{0,1\}^n$, the deterministic verifier guesses a witness $w \in \{0,1\}^{n^k}$ and $f_n \in \{0,1\}^{n^{1+\varepsilon/7}}$ and checks that $f_n \in \mathcal{L}$. It then uses the algorithm from Theorem 13.5.12 with parameter $\varepsilon/7$ and with a bound of $Q = n^{(1+\varepsilon/3)\cdot c_{\mathsf{NW}}\cdot\sqrt{\eta}}$ on the number of queries to obtain a truth-table $g_n \in \{0,1\}^{n^{1+\varepsilon/3}}$.[35]

The verifier uses G from Theorem 13.5.18 with parameters $\varepsilon_{\mathsf{NW}} = \eta$ and $N = n^{1+\varepsilon/3}$ and $\mu = C\cdot\sqrt{\eta}$ for a sufficiently large universal constant $C > 0$, giving G oracle access to g_n. Denoting the output strings of $G^{g_n}(1^N)$ by $s_1,\ldots,s_L$, the verifier outputs $\mathtt{MAJ}_i\{V(x,s_i,w)\}$. Note that the running time of G and its number of output strings L are bounded by $n^{(1+\varepsilon/3)\cdot(1+O(\sqrt{\eta}))} < n^{1+\varepsilon}$, assuming that $\eta > 0$ is sufficiently small. Thus, the deterministic verifier runs in time $n^{1+(1+\varepsilon)\cdot k}$.

[35] The truth-table that Theorem 13.5.12 outputs is of length $n^{(1+\varepsilon/7)^2}$, and by padding it with zeroes we can assume that it is of length $n^{1+\varepsilon/3}$.

The reconstruction argument. For any $x \in \{0,1\}^n$, let $D_x \colon \{0,1\}^{n^\eta} \to \{0,1\}$ be the function

$$D_x(r) = 1 \iff \exists w : V(x,r,w) = 1 \,.$$

Assume towards a contradiction that for infinitely many pairs (x, f_n) such that $x \in \mathsf{N}$ and $f_n \in \{0,1\}^{|x|^{1+\varepsilon/7}}$ it holds that $\Pr_{i\in[L]}[D_x(s_i) = 1] \geq 1/2$, where the s_i's are the result of running G with oracle access to g_n and with the parameters above.

For every such pair, there is an advice string $\mathtt{adv}$ of length $|g_n|^{1-C\cdot\sqrt{\eta}}$ such that the reconstruction R from Theorem 13.5.18 computes the function whose truth-table is g_n in time $|g_n|^{c_{\mathsf{NW}}\cdot\sqrt{\eta}}$ when given $\mathtt{adv}$ as non-uniform advice and oracle access to D_x. By adding x to the advice (we can assume that $|x| < |\mathtt{adv}|$ by choosing η to be sufficiently small), this is an algorithm that runs in time $|g_n|^{c_{\mathsf{NW}}\cdot\sqrt{\eta}}$, uses $|g_n|^{1-C\cdot\sqrt{\eta}}$ bits of advice, and makes Q non-adaptive oracle queries to $\mathsf{NTIME}[n^k]$. [36] Using an efficient reduction of $\mathsf{NTIME}[n^k]$ to SAT (see [Tou01, FLvMV05]), we obtain an algorithm that runs in time $T' = \tilde{O}(n^k)$, uses $a' = |g_n|^{1-C\cdot\sqrt{\eta}}$ bits of advice, and makes Q non-adaptive queries to a SAT oracle.

The algorithm R above yields a non-adaptive non-uniform SAT-oracle program with running time T' and advice complexity a' and query complexity Q, in the sense of Definition 13.5.11. By Theorem 13.5.12, it holds that g_n can be computed by a nondeterministic non-uniform program with running time $T = T' \cdot \gamma$ and advice complexity $a = a' \cdot \gamma$, where $\gamma = (Q, \log(|f_n|)^{1/\varepsilon})^{c_{\mathsf{SU}}} = n^{c_{\mathsf{SU}}\cdot c_{\mathsf{NW}}\cdot(1+\varepsilon/3)\cdot\sqrt{\eta}} = |g_n|^{c_{\mathsf{SU}}\cdot c_{\mathsf{NW}}\cdot\sqrt{\eta}}$ and c_{SU} is the constant from Theorem 13.5.12 that depends on $\varepsilon > 0$.

By the straightforward simulation of the infinite sequence of non-uniform programs to a Turing machine with advice,[37] there exists a nondeterministic machine M that runs in time $\tilde{O}(T)$ and an advice sequence of length a and infinitely many $g_n \in \mathcal{L}$ such that on input length $\log(|g_n|)$, when M is given the appropriate advice it computes g_n. Plugging in the parameters, we have that

$$\begin{aligned} \tilde{O}(T) &= \tilde{O}(n^k \cdot |g_n|^{c_{\mathsf{SU}}\cdot c_{\mathsf{NW}}\cdot\sqrt{\eta}}) = \tilde{O}(|g_n|^{k/(1+\varepsilon/3)+c_{\mathsf{SU}}\cdot c_{\mathsf{NW}}\cdot\sqrt{\eta}}) < |g_n|^{k/(1+\varepsilon/4)} \\ a &= |g_n|^{1-C\cdot\sqrt{\eta}} \cdot |g_n|^{c_{\mathsf{SU}}\cdot c_{\mathsf{NW}}\cdot\sqrt{\eta}} \leq |g_n|^{1-\sqrt{\eta}} \end{aligned}$$

where the upper-bound on T follows by choosing a sufficiently small $\eta = \eta(\varepsilon) > 0$. This contra-

[36] In more detail, queries to the oracle are of the form $(x, r) \in \{0,1\}^n \times \{0,1\}^{n^\eta}$, and the oracle answers "yes" iff there exists w such that $V(x,r,w) = 1$.

[37] That is, the machine gets as advice the description of the program and simulates it.

dicts the usefulness of $\mathcal{L}$ if we choose $\delta > 0$ to be sufficiently small. ■

By composing the "outer" PRG from Proposition 13.5.3 with the PRG from Proposition 13.5.19, in the exact same way as in Section 13.5.2, we obtain the following stronger version of Theorem 13.1.2:

Theorem 13.5.20 (stronger version of Theorem 13.1.2)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that:*

1. *There exists an $\mathsf{NTIME}[N^{k+\varepsilon/3}]$-constructive property useful against $\mathsf{MAMTIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$.*
2. *For every $k \geq 1$ there exists an $\mathsf{NTIME}[N^{k+\varepsilon/3}]$-constructive property useful against $\mathsf{NTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$.*

Then, for every constant $c \in \mathbb{N}$ it holds that

$$\mathsf{prAMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{prNTIME}[n \cdot T^{\lceil c/2 \rceil + \varepsilon}] \,.$$

13.5.4 Uniform Trade-offs for $\mathsf{AM} \cap \mathsf{coAM}$

In this section we prove Theorem 13.1.3. The main claim in the proof is the following, which shows that under the hardness assumption of Theorem 13.1.3, one can reduce the randomness complexity of any $(\mathsf{AM} \cap \mathsf{coAM})\mathsf{TIME}[T]$ protocol to roughly $\log T(n)$. That is:

Proposition 13.5.21 (radically reducing the number of random coins of $\mathsf{AM} \cap \mathsf{coAM}$)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \notin$ i.o.-$(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}_2^{[\rightleftharpoons 7]}[2^{(1-\delta)\cdot n}]$ such that truth-tables of L of length $N = 2^n$ can be recognized in nondeterministic time $N^{1+\varepsilon/3}$. Then, for every time-computable T it holds that*

$$(\mathsf{AM} \cap \mathsf{coAM})\mathsf{TIME}[T] \subseteq (\mathsf{AM} \cap \mathsf{coAM})\mathsf{TIME}[T^{1+\varepsilon}, (1+\varepsilon) \cdot \log(T(n))].$$

Note that Theorem 13.1.3 follows immediately from Proposition 13.5.21 by enumerating all possible random choices; that is, the deterministic verifier asks the prover to send its responses to all possible $T^{1+\varepsilon}$ random challenges, checks the responses for consistency, and computes the probability that the original random verifier would have accepted.

Proof of Proposition 13.5.21. Fixing any $L \in (\mathsf{AM} \cap \mathsf{coAM})\mathsf{TIME}[T]$, we prove that $L \in \mathsf{AMTIME}[T^{1+\varepsilon}, (1+\varepsilon) \cdot \log T(n)]$; since the same argument can also be applied to the complement of L, it follows that $L \in \mathsf{coAMTIME}[T^{1+\varepsilon}, (1+\varepsilon) \cdot \log T(n)]$.

By definition, there are two T-time verifiers $V_1(x,y,z)$ and $V_0(x,y,z)$ with $|y| = |z| = T(|x|)$, such that the following holds for every $x \in \{0,1\}^n$ and $\sigma = L(x)$:

$$\Pr_{y \sim \mathbf{u}_{T(n)}}[\exists\, z \text{ s.t. } V_\sigma(x,y,z) = 1] = 1\,,$$
$$\Pr_{y \sim \mathbf{u}_{T(n)}}[\exists\, z \text{ s.t. } V_{1-\sigma}(x,y,z) = 1] \leq 1/3\,.$$

Let δ_0 be the corresponding constant from Proposition 13.5.2 when setting $\varepsilon_0 = \varepsilon$, and without loss of generality assume that $\delta_0 < \varepsilon_0$.

Construction of the new verifier V'. Given input $x \in \{0,1\}^n$, let $N = T(n)$. Let $\ell = (1 + \varepsilon/3 - \delta_0/10) \cdot \log N$. V' guesses $f_\ell \in \{0,1\}^{N^{1+\varepsilon/3-\delta_0/10}}$ and verifies that $f_\ell \in \mathtt{tt}(L)$ by guessing the witness for the nondeterministic algorithm that recognizes $\mathtt{tt}(L)$ (otherwise V' rejects). V' then computes $\mathsf{Enc}(f_\ell)$ using the encoder in Theorem 13.3.11 with parameters $m = |f_\ell|$ and $\eta > 0$ that is a sufficiently small constant.

Next, we consider the following pair language

$$L_{\mathsf{pair}} = \{(1^\ell, \mathsf{Enc}(f_\ell))) : f_\ell \text{ is the truth-table of } L_{\mathsf{hard}} \text{ on } \ell\text{-bit inputs}\}.$$

Note that L_{pair} has stretch $K(\ell) = |\mathsf{Enc}(f_\ell)| = \tilde{O}(2^\ell)$, and is decidable in $\mathsf{NTIME}[\tilde{T}(\ell))]$ for some $\tilde{T}(\ell) = \tilde{O}(2^\ell)$. Let M be a $\tilde{T}(\ell)$-time nondeterministic machine such that $(x,y) \in L_{\mathsf{pair}}$ if and only if there exists $w \in \{0,1\}^{\tilde{T}(|x|)}$ such that $M((x,y),w) = 1$.

We apply Theorem 13.3.16 to L_{pair} to obtain a $\mathrm{poly}(\ell)$-time verifier V_{pair} and a $\tilde{O}(2^\ell)$ time algorithm A_{pair}. By Theorem 13.3.16, for some $r(\ell) \leq \ell + O(\log \ell)$, and for every sufficiently large $\ell \in \mathbb{N}$, the followings hold:

1. For every $w \in \{0,1\}^{\tilde{T}(\ell)}$ such that $M((1^\ell, \mathtt{Enc}(f_\ell)), w) = 1$, $A_{\mathsf{pair}}(1^\ell, \mathtt{Enc}(f_\ell), w)$ outputs a proof $\pi \in \{0,1\}^{2^{r(\ell)}}$ such that

$$\Pr[V_{\mathsf{pair}}^{\mathsf{Enc}(f_\ell),\pi}(1^\ell, \mathbf{u}_r) = 1] = 1.$$

2. For every $y \in \{0,1\}^{K(\ell)}$ that has hamming distance at least $K(\ell)/20$ from $\mathsf{Enc}(f_\ell)$, for every $\pi \in \{0,1\}^{2^r}$ it holds that

$$\Pr[V_{\mathsf{pair}}^{\mathsf{Enc}(f_\ell),\pi}(1^\ell, \mathbf{u}_r) = 1] \leq 1/3.$$

Then, V' guesses $w \in \{0,1\}^{\tilde{T}(\ell)}$ such that $M((1^\ell, \mathtt{Enc}(f_\ell)), w) = 1$ (V' reject immediately if this does not hold), computes $\pi_\ell^w = A_{\mathsf{pair}}(1^\ell, \mathtt{Enc}(f_\ell), w)$, which has length $2^{r(\ell)}$. Now we define

$$\Lambda_\ell^w = |\mathtt{Enc}(f_\ell)| \circ \pi_\ell \circ 0^{N^{1+\varepsilon/3} - |\mathtt{Enc}(f_\ell)| - |\pi_\ell^w|}.$$

Note that $|\Lambda_\ell^w| = N^{1+\varepsilon/3}$. Consider the generator G from Proposition 13.5.2 with $\varepsilon_0 = \varepsilon$, input 1^N, and oracle access to Λ_ℓ, and denote its list of outputs by $s_1^w, \ldots, s_{N^{1+\varepsilon}}^w \in \{0,1\}^N$. The new verifier V' chooses a random $i \in [N^{1+\varepsilon}]$ and simulates V_1 at input x with random coins s_i^w. Note that this verifier indeed runs in time $O(N^{1+\varepsilon})$. For notational convenience, we now denoted the set of all accepted w by

$$\mathcal{W} = \{w : w \in \{0,1\}^{\tilde{T}(\ell)} \wedge M((1^\ell, \mathtt{Enc}(f_\ell)), w) = 1\}.$$

We note that $|\mathcal{W}| \geq 1$ since M decides L_{pair}.

The reconstruction argument. Assume towards a contradiction that V' fails to solve L. Since V_1 has perfect completeness and $|\mathcal{W}| \geq 1$, V' can only make mistakes when $x \notin L$. In particular, there exist $x \in \{0,1\}^n$ and $w^* \in \mathcal{W}$:

$$x \notin L \wedge \Pr_{i \in [N^{1+\varepsilon}]}[\exists \omega : V_1(x, s_i^{w^*}, \omega) = 1] > .5\,. \tag{13.5.2}$$

Denote by $D_x \colon \{0,1\}^N \to \{0,1\}$ the function $D_x(z) = 1 \iff \exists \omega : V_1(x, z, \omega) = 1$. From now on, we will use Λ_ℓ to denote $\Lambda_\ell^{w^*}$ for simplicity.

We first design an $\mathsf{MATIME}^{[=7]}$ protocol Π_1 for f_ℓ. Let $\tau = c_0 \cdot \log N$ for a big enough constant $c_0 > 1$. Given an input $\mu \in [|f_\ell|]$, our protocol acts as follows. (See Figure 13-1 for a visual diagram of the protocol.)

1. The prover sends a (supposedly bad) input $x \in \{0,1\}^n$. From now on, we consider the reconstruction algorithm R from Proposition 13.5.2 with oracle access to D_x. Let $\bar{t} = N^{\varepsilon_0/10}$ be the number of parallel queries R makes, and let $s, a \in \mathbb{N}$ be the parameters from Proposition 13.5.2.
2. The verifier draws $h \sim \mathbf{h}$ ($\mathbf{h}$ is defined in Proposition 13.5.2) and τ queries $z_1, z_2, \ldots, z_\tau \in \{0,1\}^N$ uniformly at random, and sends them to the prover. Note that $h \in \{0,1\}^{|f_\ell|^{1-2\delta_0}}$.
3. The prover sends an advice $\mathtt{adv} \in \{0,1\}^{|f_\ell|^{1-2\delta_0}}$, together with witnesses $\omega_1, \omega_2, \ldots, \omega_\tau \in$

$\{0,1\}^N$.

4. The verifier then performs the following:
   (a) It rejects immediately if
   $$\Pr_{i\in[\tau]}[V_0(x,z_i,\omega_i)=1]\leq 1/2.$$
   (b) Otherwise, it draws $\alpha_1,\alpha_2,\ldots,\alpha_\tau\in\{0,1\}^{r(\ell)}$, and then simulates $V_{\mathsf{pair}}(1^\ell,\alpha_i)$ for each $i\in[\tau]$ to obtain a list of $\mathrm{polylog}(\ell)$ queries to the input oracle (which it hopes will be $\mathsf{Enc}(f_\ell)$) and to the proof oracle (which it hopes will be π). We can view those queries as queries $q\in[N^{1+\varepsilon/3}]$ to some values of Λ_ℓ.[38]
   (c) The verifier also simulates the local decoder for Enc with input μ, which issues $|f_\ell|^\eta$ many queries $\beta_1,\beta_2,\ldots,\beta_{|f_\ell|^\eta}\in[|\mathtt{Enc}(f_\ell)|]$. Again, we view all these $|f_\ell|^\eta$ queries as queries $q\in\left[N^{1+\varepsilon/3}\right]$ to some values of Λ_ℓ.
   (d) To summarize, in this turn, the verifier obtained $N_{\mathsf{q}}=O\left(|f_\ell|^\eta\right)$ many queries $q_1,\ldots,q_{N_{\mathsf{q}}}\in[N^{1+\varepsilon/3}]$ and sent them to the prover.
5. For every $i\in[N_{\mathsf{q}}]$, the prover sends a witness $w_i\in\{0,1\}^{|f^\ell|^{1-\delta_0}}$, which is supposed to be the witness for the execution of R on q_i.
6. For every $i\in[N_{\mathsf{q}}]$, the verifier simulates R on input q_i with witness w_i with fresh random coins γ_i, obtains queries $z_{i,1},\ldots,z_{i,\bar{t}}\in[N]$, and sends them to the prover.
7. For every $i\in[N_{\mathsf{q}}]$ and $j\in[\bar{t}]$, the prover sends a witness $\omega_{i,j}\in\{0,1\}^N$, which is supposed to be the witness for $D_x(z_{i,j})$.
8. (Deterministic step.) Finally, the verifier performs the following verifications:
   (a) For every $i\in[N_{\mathsf{q}}]$, it constructs the sequence $\rho_i\in\{0,1\}^{\bar{t}}$ such that $(\rho_i)_j=V_1(x,z_{i,j},\omega_{i,j})$ for every $j\in[\bar{t}]$. If for any $i\in[N_{\mathsf{q}}]$, ρ_i is (s,α)-deficient, then it immediately rejects.
   (b) Otherwise, for every $i\in[N_{\mathsf{q}}]$, let $d_i\in\{0,1\}^{\bar{t}}$ be such that $(d_i)_j=D_x(z_{i,j})$ for every $j\in[\bar{t}]$. We know that ρ_i is (s,α)-indicative of d_i.[39] The verifier then finishes the executions of R on all the q_i's, and rejects immediately if it gets $\perp$ form any of these executions. Next, the verifier uses the obtained values to finish the simulation of $V_{\mathsf{pair}}(1^\ell,\alpha_i)$ for every $i\in[\tau]$, and rejects immediately if any of the $V_{\mathsf{pair}}(1^\ell,\alpha_i)=0$. Finally, the verifier finishes the simulation of the local decoder for Enc to obtain an output $\omega\in\{0,1\}$, and accepts iff $\omega=1$.

[38] More formally, querying the i-th bit of the proof oracle corresponding to $\mathsf{Enc}(f_\ell)$ translates to querying $(\Lambda_\ell)_i$, and querying the i-th bit of the proof oracle corresponding to π translates to querying $(\Lambda_\ell)_{i+|\mathtt{Enc}(f_\ell)|}$.

[39] This holds since for every $j\in[\bar{t}]$ $(\rho_i)_j=1$ implies $(d_i)_j=1$ by the definition of ρ_i and D_x.

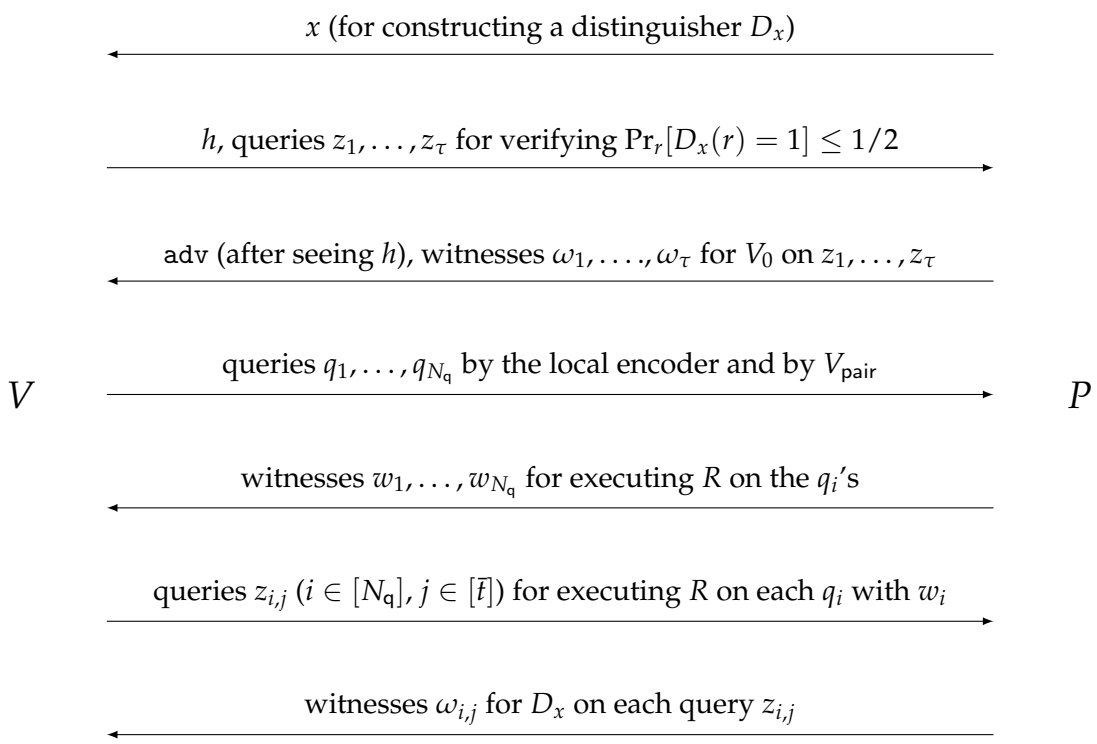


Figure 13-1: An illustration of the $\mathsf{MATIME}^{[\rightleftharpoons 7]}$ protocol for f_ℓ in the reconstruction argument An illustration of the $\mathsf{MATIME}^{[\rightleftharpoons 7]}$ protocol for f_ℓ in the reconstruction argument. The verifier V is on the left-hand side and the prover P is on the right-hand side, and observe that the prover speaks first.

Completeness. For every $\mu \in [|f_\ell|]$ such that $(f_\ell)_\mu = 1$, we will show that there exists a prover strategy in the protocol Π_1 such that the verifier accepts, with high probability. In Step (1) the prover sends the ("bad") input $x \in \{0,1\}^n$ such that (13.5.2) holds. Since $x \notin L$, we have that $\Pr[D_x(\mathbf{u}_N) = 1] \leq 1/3$ and that $\Pr_{z \sim \mathbf{u}_N}[\exists\, \omega\ V_0(x, z, \omega) = 1] = 1$. Therefore, in Step (3), the prover can always send $\omega_1, \ldots, \omega_\tau$ so that the verifier does not reject in Step (4a) in Π_1.

By the completeness case of the "Honest oracle" part of Proposition 13.5.2, and since Eq. (13.5.2) holds, with probability $1 - 1/N$ over $h \sim \mathbf{h}$, there exists $\mathtt{adv} \in \{0,1\}^{|f_\ell|^{1-2\delta_0}}$ such that by sending adv to the verifier in Step (3), the following holds: Given correct witnesses in Step (5), with probability at least $1 - O(N_q)/N$, for all of the verifier's $O(N_q)$ simulations of R, given correct witnesses in Step (7), the verifier obtains the correct values in Λ_ℓ. In particular, it means that $V_{\mathsf{pair}}(1^\ell, \alpha_i) = 1$ for all $i \in [\tau]$, and the local decoder returns the correct value $(f_\ell)_\mu = 1$. Putting the above together, the verifier accepts with probability at least $2/3$.

Soundness. Let $\mu \in [|f_\ell|]$ such that $(f_\ell)_\mu = 0$. We first note that if thr prover sends $x \in L$ in Step (1), then we have $\Pr_{z\sim \mathbf{u}_N}[\exists\ \omega\ V_0(x,z,\omega) = 1] \leq 1/3$, meaning that with probability at least $1 - 1/N$, the verifier rejects in Step (4a), no matter what witnesses $\omega_1, \ldots, \omega_\tau$ it receives in Step (3). Therefore, we can assume that $x \notin L$ in Step (1) and that the verifier does not reject immediately in Step (4a). In particular, since $x \notin L$, we have that $\Pr_{z\sim \mathbf{u}_N}[D_x(z)] \leq 1/3$.

Now, by the "Dishonest oracles" case of Proposition 13.5.2, with probability $1 - 1/N$ over $h \sim \mathbf{h}$, for every possible $\mathtt{adv}$ sent by the prover in Step (3), there exists $g \in \{0,1\}^{N^{1+\varepsilon/3}}$ such that the following holds for every $i \in [N_\mathsf{q}]$: for every possible w_i sent by the prover in Step (5), with probability at least $1 - 1/N$ over the verifier's randomness γ_i drawn in Step (6), either the verifier rejects in Step (8a), or the simulated $R(q_i, w_i)$ in Step (8b) given advice $(h, \mathtt{adv})$ outputs either $g(q_i)$ or $\perp$ (this holds since ρ_i in in Step (8) is either (s,α)-deficient, in which case the verifier rejects, or (s,α)-indicative of d_i, in which case $R(q_i, w_i)$ outputs either $g(q_i)$ or $\perp$).

By the above discussions and a union bound, with probability at least $1 - O(N_\mathsf{q})/N$, at Step (8b), either the verifier rejects (meaning that some of $R(q_i, w_i)$ outputs $\perp$), or all the simulated $R(q_i, w_i)$ outputs $g(x_i)$. We will denote the above as event $\mathcal{E}$.

From now on we condition on the event $\mathcal{E}$ and we assume that the verifier does not reject in Step (8a). Let y be the first $|\mathtt{Enc}(f_\ell)|$ bits of g, and π be the next $|\pi_\ell|$ bits of g. Note that g is already determined by the end of Step (3) and hence the verifier's randomness $\alpha_1, \ldots, \alpha_\tau$ in Step (4) is independent of g. Hence, if y has Hamming distance at least $|\mathtt{Enc}(f_\ell)|/20$ from $\mathtt{Enc}(f_\ell)$, then with probability at least $1 - 1/N$, $V^{y,\pi}_{\mathsf{pair}}(1^\ell, \alpha_i) = 0$ for at least one $i \in [\tau]$, and the verifier rejects in Step (8b).[40] Hence, we can assume that y has hamming distance at most $|\mathtt{Enc}(f_\ell)|/20$ from $\mathtt{Enc}(f_\ell)$. By Theorem 13.3.11, the local decoder returns the correct value $(f_\ell)_\mu = 0$ with probability at least $1 - 1/N$, and the verifier rejects at the end. Putting everything together, the verifier rejects with probability at least $2/3$.

Protocol Π_0 for the complement of f_ℓ. Finally, we modify Π_1 to obtain another $\mathsf{MATIME}^{[\rightleftharpoons 7]}$ protocol Π_0. The only difference between Π_0 and Π_1 is that at Step (8b) of Π_0, the verifier in Π_0 accepts if and only if $\omega = 0$ instead of $\omega = 1$. The completeness and soundness of Π_0 for computing the complement of f_ℓ follow from the same proof as that for Π_1.

[40] Here we crucially used the fact that g (and thus y and π) is fixed before the verifier draws the α_i's.

Running time of the protocols Π_1 and Π_0. Finally, we bound the running time of the verifier in Π_1 (and hence also in Π_0), as follows:

$$\begin{aligned}
&\underbrace{\tilde{O}\left(|\Lambda_\ell|^{1-2\delta_0}\right)}_{\text{Step (2)}} + \underbrace{\tilde{O}(N) + \mathrm{polylog}(N) + |f_\ell|^\eta}_{\text{Steps (4a)+ (4b)+ (4c)}} + \underbrace{O(N_{\mathsf{q}} \cdot |\Lambda_\ell|^{1-\delta_0})}_{\text{Step (6)}} \\
&\qquad + \underbrace{O(N_{\mathsf{q}} \cdot N \cdot \bar{t})}_{\text{Step (8a)}} + \underbrace{O(N_{\mathsf{q}} \cdot |\Lambda_\ell|^{1-\delta_0} + \mathrm{polylog}(N) + |f_\ell|^\eta)}_{\text{Step (8b)}} \\
&\leq N_q \cdot O(N \cdot \bar{t} + |\Lambda_\ell|^{1-\delta_0}) \\
&= |f_\ell|^\eta \cdot O(N^{1+\varepsilon_0/10} + |\Lambda_\ell|^{1-\delta_0}) \,,
\end{aligned}$$

and this can be made smaller than $|\Lambda_\ell|^{1-\delta}$ by choosing η and δ to be sufficiently small. This contradicts the hardness of f_ℓ. ■

13.6 Optimality under #NSETH

In this section we prove that the derandomization conclusions in Theorem 13.1.1 and Theorem 13.1.2 are essentially optimal, under the assumption #NSETH. First we lower bound the derandomization overhead of protocols in which the prover speaks first (*i.e.*, of MA and $\mathsf{MATIME}^{[\rightleftharpoons c]}$), as follows:

Theorem 13.6.1 (a lower bound on derandomization of $\mathsf{MATIME}^{[\rightleftharpoons c]}$, under #NSETH)**.** *Suppose that* #NSETH *holds. Then, for every integer $c \geq 2$ and real number $d \geq 1$ and $\varepsilon \in (0,1)$, letting $T(n) = n^d$, it holds that*

$$\mathsf{MATIME}^{[\rightleftharpoons c]}[T] \not\subseteq \mathsf{NTIME}[T^{\lfloor c/2 \rfloor + 1 - \varepsilon}] \,.$$

Proof. Without loss of generality we can assume $\varepsilon \in (0, 0.01)$. For the sake of contradiction, we assume that

$$\mathsf{MATIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{NTIME}[T^{\lfloor c/2 \rfloor + 1 - \varepsilon}] \,.$$

We instantiate Theorem 13.3.17 with $k = \lfloor c/2 \rfloor$ and $\delta = \frac{1}{k+1}$, in which case $\gamma = \frac{1}{k+1}$. It follows that there is an $\mathsf{MATIME}^{[\rightleftharpoons 2k]}[2^{n/(k+1)+o(n)}]$ protocol Π that computes the number of satisfying assignment to a formula C with $2^{o(n)}$ size and n bits input.

We first define a decisional $\mathsf{MATIME}^{[\rightleftharpoons 2k]}$ protocol Π^{D}, such that $\Pi^{\mathsf{D}}(C, z) = 1$ for an n-input formula C and an integer $z \in \{0, 1, \ldots, 2^n\}$, if the number of satisfying assignments to C is z.

(Indeed, Π^{D} can be constructed by simply simulating Π on the input C and only accepting if Π accepts and the accepted output is z.)

Now we pad the input of the protocol Π^{D} to be of length $N = 2^{\frac{n(1+\tau)}{(k+1)d}}$ for a sufficiently small constant $\tau \in (0,1)$ to be specified later. Then, the running time of the protocol Π^{D} is bounded by $2^{n/(k+1)+o(n)} \leq 2^{(1+\tau)n/(k+1)} = T(N)$. Hence, by our assumption, Π^{D} has a $T(N)^{k+1-\varepsilon} = 2^{\frac{(1+\tau)n(k+1-\varepsilon)}{(k+1)}}$ time nondeterministic algorithm M_{D}.

Setting $\tau = \varepsilon/4(k+1)$, it holds that $\frac{(1+\tau)n(k+1-\varepsilon)}{(k+1)} < (1-\tau)\cdot n$. Now we can construct a nondeterministic algorithm refuting #NSETH as follows: given a formula $C\colon \{0,1\}^n \to \{0,1\}$ of size $2^{o(n)}$, guess $z \in \{0,1,\ldots,2^n\}$, simulate M_{D} on input (C,z), output z if M_{D} accepts and $\perp$ otherwise. By the above discussion, this is a nondeterministic algorithm that counts the number of solutions to n-bit formulas of size $2^{o(n)}$ in time $2^{(1-\tau)\cdot n}$, a contradiction to #NSETH. ■

By a more careful argument, we now lower bound the derandomization overhead of protocols in which the verifier speaks first (*i.e.*, of $\mathsf{AMTIME}^{[\rightleftharpoons c]}$), as follows:

Theorem 13.6.2 (a lower bound on derandomization of $\mathsf{AMTIME}^{[\rightleftharpoons c]}$, under #NSETH)**.** *Suppose that* #NSETH *holds. Then, for every integer $c \geq 2$ and real number $d \geq 1$ and $\varepsilon \in (0,1)$, letting $T(n) = n^d$, it holds that*

$$\mathsf{AMTIME}^{[\rightleftharpoons c]}[T] \not\subseteq \mathsf{NTIME}[n \cdot T^{\lceil c/2 \rceil - \varepsilon}] .$$

Proof. Without loss of generality we can assume $\varepsilon \in (0, 0.01)$. For the sake of contradiction, we assume that

$$\mathsf{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{NTIME}[n \cdot T^{\lceil c/2 \rceil - \varepsilon}] .$$

We instantiate Theorem 13.3.17 with $k = \lceil c/2 \rceil$ and $\delta = \frac{1}{dk+1}$ and $\gamma = (1-\delta)/k$. Note that since $d \geq 1$, we have $\delta \leq \gamma$. It follows that there is an $\mathsf{MATIME}^{[\rightleftharpoons 2k]}[2^{\gamma \cdot n + o(n)}]$ protocol Π that computes the number of satisfying assignment to a formula C with $2^{o(n)}$ size and n bits input, and the first message of Π has length $2^{\delta \cdot n + o(n)}$.

As in the proof of Theorem 13.6.1, we first define a decisional $\mathsf{MATIME}^{[\rightleftharpoons 2k]}$ protocol Π^{D}, such that $\Pi^{\mathsf{D}}(C,z) = 1$ for an n-input formula C and an integer $z \in \{0,1,\ldots,2^n\}$, if the number of satisfying assignments to C is z. We note that the prover messages of the honest prover in Π^{D} are identical to those in Π. Furthermore, by the moreover part of Theorem 13.3.17, the first message of Π^{D} is such that the acceptance probability of the subsequent protocol is either 1 or at most $1/3$. (Indeed, Π^{D} inherits this property from Π.)

Let $\Pi^{\mathsf{D}}_{>1}$ be the sub-protocol of Π^{D} after the first message and let $\ell(n) = 2^{(1+\tau)\cdot\delta n}$, where $\tau \in (0,1)$ is a small enough constant to be specified later. We define a new language L' such that $L'(x,\pi,1^{\ell(n)-|x|-|\pi|}) = 1$ if $\Pi^{\mathsf{D}}_{>1}$ accepts the input/first-message pair (x,π) with probability 1, and $L'(x,\pi,1^{\ell(n)-|x|-|\pi|}) = 0$ if $\Pi^{\mathsf{D}}_{>1}$ accepts (x,π) with probability at most $1/3$. Note that (by the discussion above) the problem L' is indeed a language (*i.e.*, a total function rather than a promise problem), and L' can be decided by $\Pi^{\mathsf{D}}_{>1}$, which is an $\mathsf{AMTIME}^{[\rightleftharpoons 2k-1]}$ protocol with running time $2^{\gamma\cdot n+o(n)}$.

Note that $\Pi^{\mathsf{D}}_{>1}$ takes $\ell(n)$ bits as input, and that $\ell(n)^d = 2^{(1+\tau)\delta\cdot d\cdot n} > 2^{\gamma\cdot n+o(n)}$ by the definition of δ and γ. Hence, the language $L' \in \mathsf{AMTIME}^{[\rightleftharpoons c]}[T]$. By our assumption, $L' \in \mathsf{NTIME}[n\cdot T^{k-\varepsilon}]$.

Now we construct an algorithm that refutes #NSETH: Given an n-bit formula C of size $2^{o(n)}$, guess $z \in \{0,1,\ldots,2^n\}$, guess a proof π of length $2^{\delta\cdot n+o(n)}$, outputs z if $L'((C,z),\pi,1^{\ell(n)-|(C,z)|-|\pi|}) = 1$, and outputs $\perp$ otherwise. This nondeterministic algorithm indeed counts the number of satisfying assignments, and its running time is at most

$$\ell(n)\cdot\ell(n)^{d(k-\varepsilon)} = \ell(n)^{d(k-\varepsilon)+1} = 2^{\frac{(1+\tau)\cdot(d(k-\varepsilon)+1)}{dk+1}\cdot n} < 2^{(1-\Omega(1))\cdot n},$$

where the last inequality follows by setting τ to be small enough. This is a contradiction to #NSETH. ■

13.7 Deterministic Doubly Efficient Argument Systems

In this section we prove the results from Section 13.1.3 concerning derandomization of doubly efficient proof systems; that is, we prove Theorem 13.1.7, Theorem 13.1.4, and Theorem 13.1.8.

Let us first set up some preliminaries. The derandomization algorithms in this section will be *non-black-box*, and in particular will use the following construction of a *reconstructive targeted HSG* from our very recent work [CT21a].

Theorem 13.7.1 (a reconstructive targeted HSG, see [CT21a, Proposition 6.2])**.** *For every $\alpha',\beta' > 0$ and sufficiently small $\eta = \eta_{\alpha',\beta'} > 0$ the following holds. Let $\bar{T},k\colon \mathbb{N}\to\mathbb{N}$ be time-computable functions such that $\bar{T}(N) \geq N$, and let $g\colon \{0,1\}^N \to \{0,1\}^k$ (where $k = k(N)$) such that the mapping of $(x,i) \in \{0,1\}^N \times [k]$ to $g(x)_i$ is computable in time $\bar{T}(N)$. Then, there exists a deterministic algorithm G_g and a probabilistic algorithm* Rec *that for every $z \in \{0,1\}^N$ satisfy the following:*

1. **Generator.** *When G_g gets input z and $\eta > 0$, it runs in time $k\cdot\bar{T}(N) + \mathrm{poly}(k)$ and outputs a list*

of $\mathrm{poly}(k)$ *strings in* $\{0,1\}^{k^\eta}$. [41]

2. **Reconstruction.** *When* Rec *gets as input* z *and* $\eta > 0$*, and gets oracle access to a function* $D_z \colon \{0,1\}^{k^\eta} \to \{0,1\}$ *that* $(1/k^\eta)$*-distinguishes the uniform distribution over the output-list of* $G_g(z,\eta)$ *from a uniform* k^η*-bit string, it runs in time* $\tilde{O}(k^{1+\beta'}) + k^{\beta'} \cdot \bar{T}(N)$*, makes* $\tilde{O}(k^{1+\beta'})$ *queries to* D_z*, and with probability at least* $1 - 2^{-k^\eta}$ *outputs a string that agrees with* $g(z)$ *on at least* $1 - \alpha'$ *of the bits.*

The reconstruction algorithm above can be thought of as *approximately printing* the string $g(z)$ (*i.e.*, printing a string that agrees with $g(z)$ on at least $1 - \alpha'$ of the bits). For convenience, we define the following corresponding notion of hardness, which is *failing* to approximately print.

Definition 13.7.2 (failing to approximately print)**.** *We say that a probabilistic algorithm* M fails to approximately print a function $f \colon \{0,1\}^n \to \{0,1\}^*$ with error α on a given string $x \in \{0,1\}^n$ *if* $\Pr[M(x)_i = f(x)_i] \leq 1 - \alpha$*, where the probability is over* $i \in [|f(x)|]$ *and over the random coins of* M*. We will use shorthand notation and say that* M fails to approximately print $f(x)$ with error α*.*

13.7.1 Warm-up: The Case of an MA-style System

Towards presenting our result, we first present an appealing special case whose proof is far less involved. Denote by $\mathsf{delP}^{[\rightleftharpoons 2]}_{\mathsf{MA}}[T]$ a doubly efficient proof system in which the prover speaks first, sending a proof π, and then the verifier tosses random coins and decides whether to accept or reject the input x with the proof π. Under suitable hardness assumptions, we simulate $\mathsf{delP}^{[\rightleftharpoons 2]}_{\mathsf{MA}}$ by deterministic doubly efficient argument systems, with essentially no time overhead.

Theorem 13.7.3 (derandomizing $\mathsf{delP}^{[\rightleftharpoons 2]}_{\mathsf{MA}}$ into deterministic doubly efficient argument systems, with almost no overhead)**.** *Suppose that non-uniformly secure one-way functions exist. Let* $T(n)$ *be any polynomial, and assume that for every* $\varepsilon' > 0$ *there exist* $\alpha, \beta \in (0,1)$ *and a function* $f = f^{(\varepsilon')}$ *mapping* $n + T(n)$ *bits to* $k(n) = n^{\varepsilon'}$ *bits such that:*

1. *There exists a nondeterministic unambiguous machine that gets input* $((x,\pi),i) \in \{0,1\}^{n+T} \times [n^{1+\varepsilon'}]$ *and outputs the* i^{th} *bit of* $f(x,\pi)$ *in time* $\bar{T} = T(n) \cdot k$*.*
2. *For every probabilistic algorithm* M *running in time* $\bar{T} \cdot n^\beta$ *and every distribution* $\mathbf{P}$ *over* $\{0,1\}^{n+T}$ *that is samplable in polynomial time, with probability at least* $1 - n^{-\omega(1)}$ *over* $(x,\pi) \sim \mathbf{P}$ *it holds that* M *fails to approximately print* $f(x,\pi)$ *with error* α*.*

[41] The fact that the number of strings is $\mathrm{poly}(k)$ is not mentioned in the original statement in [CT21a, Proposition 6.2], but this is just an omission. This fact is established in the proof of the proposition and the applications of the proposition rely on it.

Then, for every $\varepsilon > 0$ it holds that $\mathsf{delP}^{[\rightleftharpoons 2]}_{\mathsf{MA}}[T] \subseteq \mathsf{NARG}[n^{\varepsilon} \cdot T]$.

Proof. Let $L \in \mathsf{delP}^{[\rightleftharpoons 2]}_{\mathsf{MA}}[T]$, let V be a corresponding verifier for L, and let

$$\mathsf{Y}_V = \left\{(x,\pi) : \Pr_r[V(x,\pi,r)=1] \geq 2/3\right\}$$
$$\mathsf{N}_V = \left\{(x,\pi) : \Pr_r[V(x,\pi,r)=1] \leq 1/3\right\} ,$$

where in the expressions $V(x,\pi,r)$ above x is an input and π is a proof and r is a random string. Note that the promise-problem $(\mathsf{Y}_V, \mathsf{N}_V)$ can be decided in probabilistic linear time.

Let $\varepsilon' = \varepsilon/c$ for a sufficiently large constant $c > 1$, and let $k(n) = n^{\varepsilon'}$. Let $f = f^{(\varepsilon')}$ be the corresponding function from our hypothesis, and note that the upper bound in the first item is $\bar{T} = T \cdot k$, whereas the lower bound in the second item is $\bar{T} \cdot k^{1+\beta}$.

First step: Reduce the number of random coins to n^{η}. For a sufficiently small constant $\mu = \mu(\varepsilon') > 0$ that will be determined later, let $G^{\mathtt{cry}}$ be the PRG from Theorem 13.3.15, instantiated with stretch $n^{\mu} \mapsto T(n)$. Consider the verifier V' that uses only n^{μ} coins and is defined by $V'(x,\pi,s) = V(x,\pi,G^{\mathtt{cry}}(s))$. Note that for every fixed x,π we have that $\Pr_s[V'(x,\pi,s)=1] \in \Pr_r[V(x,\pi,r)=1] \pm n^{-\omega(1)}$. Hence, $\mathsf{Y}_{V'} = \mathsf{Y}_V$, where $\mathsf{Y}_{V'} \stackrel{\text{def}}{=\!=} \{(x,\pi) : \Pr_r[V'(x,\pi,r)=1] \geq .66\}$, and similarly $\mathsf{N}_{V'} = \mathsf{N}_V$ where $\mathsf{N}_{V'} \stackrel{\text{def}}{=\!=} \{(x,\pi) : \Pr_r[V'(x,\pi,r)=1] \leq .33\}$. The running time of V' is $O(T^{1+\mu})$ and its number of random coins is n^{μ}.

Main step: Targeted PRG using the transcript as a source of hardness. Now, let D be the following deterministic verifier. On input $x \in \{0,1\}^n$ and proof $\pi \in \{0,1\}^T$, consider the generator from Theorem 13.7.1, instantiated with parameters

$$N = n + T \,, \qquad \bar{T}(N) = T(n) \cdot k \,,$$
$$g = f^{\varepsilon} \colon \{0,1\}^N \to \{0,1\}^{n^{1+\varepsilon'}} \,, \qquad \text{sufficiently small } \alpha', \beta', \eta \quad ;$$

we now also fix the parameter μ of $G^{\mathtt{cry}}$ above to be such that $k^{\eta} = n^{\mu}$. The verifier D runs G_g to obtain a set of $k \cdot \bar{T} + \mathrm{poly}(k) \leq T \cdot \mathrm{poly}(k)$ strings of length n^{μ} denoted $s_1, \ldots, s_{T \cdot \mathrm{poly}(k)}$ and outputs $\mathtt{MAJ}_{i \in [T \cdot \mathrm{poly}(k)]} \{V'(x,\pi,s_i)\}$. Assuming that the constant $c > 1$ is sufficiently large, the running time of this algorithm is at most $T \cdot n^{\varepsilon}$.

Analysis. The honest prover for D is identical to that of V. We now show an algorithm F that runs in time $\bar{T} \cdot n^{\beta}$, and for every fixed $(x,\pi) \in \mathsf{Y}_V$ such that $D(x,\pi) = 0$ it holds that $F(x,\pi)$

α-approximates $f(x,\pi)$. By a symmetric argument (which is identical and omitted), there exists another algorithm F' with precisely the same guarantee for any $(x,\pi) \in \mathsf{N}_V$ such that $D(x,\pi) = 1$. By our hypothesis, for every polynomial-time samplable distribution $\mathbf{P}$ over $\{0,1\}^{n+T}$, with probability at least $1 - n^{-\omega(1)}$ over choice of $(x,\pi) \sim \mathbf{P}$ both algorithms F and F' fail to α-approximate $f(x,\pi)$. Hence, the probability that a probabilistic polynomial-time algorithm can find (x,π) such that $D(x,\pi)$ errs is at most $n^{-\omega(1)}$.

Thus, it is left to construct the algorithm F. Fix $(x,\pi) \in \mathsf{Y}_V$ such that $D(x,\pi) = 0$, and denote by $D_{x,\pi} \colon \{0,1\}^{k^\eta} \to \{0,1\}$ the function $D_{x,\pi}(r) = V(x,\pi,r)$. By our assumption $D_{x,\pi}$ is a $(1/10)$-distinguisher for the uniform distribution over the output-set of G_g. We invoke the reconstruction Rec from Theorem 13.7.1; the running time of algorithm, accounting for answering its $\tilde{O}(k^{1+\beta'})$ queries to $D_{x,\pi}$, is

$$\tilde{O}(k^{1+\beta'}) + k^{\beta'} \cdot \bar{T}(n) + \tilde{O}(k^{1+\beta'}) \cdot T < \tilde{O}(T \cdot k \cdot k^{\beta'}) < \bar{T} \cdot n^{\beta} \,,$$

for a sufficiently small choice of β'; and with probability $1 - o(1) > 1 - \alpha/2$ the reconstruction outputs a string string that agrees with $f(x,\pi)$ on at least $1 - \alpha' > 1 - \alpha/2$ of the bits, for a sufficiently small $\alpha' > 0$. ■

Note that the proof above works as-is even if the initial $\mathsf{delP}^{[\rightleftharpoons 2]}_{\mathsf{MA}}$ system has imperfect completeness.

13.7.2 Basic Case: Doubly Efficient Proof Systems with Few Random Coins

The main goal in this section is to state and prove Theorem 13.1.7, which asserts that under strong hardness assumptions, we can simulate every $\mathsf{delP}^{[\rightleftharpoons c]}$ protocol by a deterministic doubly efficient argument system, with essentially no time overhead.

In Section 13.7.2 we state Theorem 13.1.7 and discuss its hypothesis, and then in Section 13.7.2 we prove the result. In Section 13.7.2 we state and prove a strong version of the result that holds for doubly efficient proof systems that have an efficient univesal prover, and in Section 13.7.2 we deduce Theorem 13.1.4 as a corollary of the latter.

The result statement and a discussion of the hypothesis

The following is a generic form of the hardness hypothesis that we will use in Theorem 13.1.7. It is inspired by a hardness assumption from Chapter 12, but the current assumption is stronger because it refers to probabilistic *oracle machines* (rather than just to probabilistic algorithms). In the assumption we fix a "complexity parameter" $n \in \mathbb{N}$, and think of all other parameters as functions of n.

Assumption 13.7.4 (non-batch-computability assumption; Assumption 13.1.6, restated)**.** *For $N, K, K', \bar{T} \colon \mathbb{N} \to \mathbb{N}$ and $\eta \colon \mathbb{N} \to (0,1)$, the $(N \mapsto K, K', \eta)$-non-batch-computability assumption for time $\bar{T}$ with oracle access to $\mathcal{O}$ is the following. There exists $f \colon \{0,1\}^* \to \{0,1\}^*$ that for every $n \in \mathbb{N}$ maps $N(n)$ bits to $K(n)$ bits and satisfies:*

1. *There exists a deterministic algorithm that gets input $(z, i) \in \{0,1\}^{N(n)} \times [K(n)]$ and outputs the i^{th} bit of $f(z)$ in time $\bar{T}(n)$.*
2. *For every oracle machine M running in time $\bar{T} \cdot K'$ and having oracle access to $\mathcal{O}$, and every collection $\mathbf{z} = \left\{\mathbf{z}_{N(n)}\right\}_{n \in \mathbb{N}}$ of distributions such that $\mathbf{z}_{N(n)}$ is over $\{0,1\}^{N(n)}$ and can be sampled in time polynomial in $\bar{T}(n)$, and every sufficiently large $n \in \mathbb{N}$, with probability at least $1 - \bar{T}(n)^{-\omega(1)}$ over choice of $z \sim \mathbf{z}_{N(n)}$ it holds that $M^{\mathcal{O}}$ fails to approximately print $f(z)$ with error $\eta(n)$.*

We will use Assumption 13.7.4 with $\bar{T}$ that is a polynomial, in which case the error bound $\bar{T}(n)^{-\omega(1)}$ is just $n^{-\omega(1)}$; that is, the error bound is any negligible function in the input length. We are now ready to state Theorem 13.1.7.

Theorem 13.7.5 (derandomizing constant-round doubly efficient proof systems into deterministic doubly efficient argument systems, with almost no overhead; Theorem 13.1.7, restated)**.** *For every $\alpha, \beta \in (0,1)$ there exists $\eta > 0$ such that the following holds. Let $c \in \mathbb{N}$ be a constant, let $T(n)$ be a polynomial, let $R(n) < T(n)$ be time-computable, and let $N(n) = n + c \cdot T(n)$ and $K(n) = R(n)^{1/\eta}$. Assume that the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T} = T \cdot K$ with oracle access to $\mathsf{prAMTIME}_2^{[\rightleftharpoons c]}[n]$ on inputs of length $O(T)$. Then,*

$$\mathsf{delP}^{[\rightleftharpoons c]}[T, R] \subseteq \mathsf{NARG}[T \cdot R^{O(c/\eta)}] \,,$$

where the O-notation hides a universal constant.

To discuss the hardness hypothesis, let us fix the parameter value $R = T^{o(1)}$, which will be the value we use in the proofs of Theorem 13.1.4 and Theorem 13.1.8. Then, loosely speaking, the

assumption in Theorem 13.7.5 can be described as follows: There exists a function f from T bits to $K = T^{o(1)}$ bits such that –

1. Each output bit of f can be computed in time $\bar{T} = T \cdot K = T^{1+o(1)}$.
2. The entire string $f(x)$ cannot be (approximately) printed in probabilistic time $\bar{T} \cdot K^{\beta}$, while making oracle queries of length T to a $\mathsf{prAMTIME}^{[\rightleftharpoons c]}$ protocol that runs in time linear in T. (And this hardness holds whp over any polynomial-time samplable distribution of T-bit strings x.)

Our proof allows relaxing the hypothesis further: For example, the oracle machine against which we assume hardness only makes *non-adaptive* queries to its oracle, and the number of queries is only $\mathrm{poly}(k)$. (We avoid including these relaxations in the assumption statement for simplicity of presentation.)

The difference from known results about batch-computability. The complexity of the oracle machine for which we assume hardness is reminiscent of the complexity of *unconditionally known* interactive protocols for batch-computing functions. However, there is a crucial difference between the two settings, which we now explain.

For any $f\colon \{0,1\}^T \to \{0,1\}^K$ whose individual output bits are computable in time $\bar{T}$, Reingold, Rothblum, and Rothblum [RRR18, Theorem 13] (following their previous result in [RRR21]) constructed a constant-round protocol for batch verification of the K output bits of f such that the verifier runs in time $\tilde{O}(K \cdot T) + K^{\beta} \cdot \bar{T}^{1+\beta} = O(K^{\beta} \cdot \bar{T}^{1+\beta})$, where $\beta > 0$ is an arbitrarily small constant.[42]

The running time of their protocol almost matches (from above) the complexity of our oracle machine, which might seem alarming. However, we stress that our oracle machine is not an interactive protocol by itself, but rather only *makes queries* to a protocol; these queries are of length $T = \bar{T}/K$, and *the protocol is only allowed linear running time* $O(T)$. Therefore, the main point of difference is that the interactive proof oracle in our hypothesized lower bound *is allowed less running time than the upper bound* $\bar{T}$ *on computing even a single output bit of* f. Needless to say, the techniques for batch-verification from [RRR21, RRR18] do not extend to such a setting.

In fact, a hardness assumption that is even stronger than the one we make still makes sense: The hypothesis would still seem reasonable if the probabilistic machine would make queries of

[42] The result of [RRR18] applies in a more general setting than the one we compare to, since they consider applying f to K different inputs $x_1, \ldots, x_k$, since their prover is efficient (*i.e.* runs in time $\mathrm{poly}(\bar{T})$), and since their result holds even when the upper bound on the complexity of f is UP rather than P.

length T to a linear-space oracle (*i.e.*, to a machine running in space $O(T)$ and time $2^{O(T)}$), rather than to a linear-time interactive proof.

Proof of Theorem 13.7.5

Let $L \in \mathsf{delP}^{[\rightleftharpoons c]}[T, R]$, and let V be a T-time verifier in a protocol with c rounds for L such that V sends R random coins in each turn. Denote by $c' = \lceil c/2 \rceil$ be the number of turns of the verifier in the interaction. Given any prover P and input x, we think of the corresponding interaction as a function of c' strings of R random coins that are chosen (uniformly) in advance, but are revealed to P during the interaction. We denote by

$$\langle V, P, x \rangle (r_1, \ldots, r_{c'})$$

the result of the interaction between V and P on input x with random coins $r_1, \ldots, r_{c'}$.

A verifier for L with $O(\log(R))$ random coins. Our goal now is to construct an alternative verifier V' for L such that in each round of interaction, V' uses $O(\log(R))$ random coins. Let $G = G_f$ be the generator from Theorem 13.7.1, instantiated with the function $f\colon \{0,1\}^N \to \{0,1\}^{R^{1/\eta}}$, with sufficiently small $\beta' < \beta$ and $\alpha' < \alpha$, and with output length R; the number of output strings of G_f is $\bar{R} \stackrel{\text{def}}{=\!=} \mathrm{poly}(K) = R^{O(1/\eta)}$, where $\eta = \eta_{\alpha',\beta'}$ is the parameter from Theorem 13.7.1, and its running time is $\bar{T} \cdot \mathrm{poly}(K) = T \cdot R^{O(1/\eta)}$. For any given $z \in \{0,1\}^N$ and $s \in [\bar{R}]$, let $G^z(s) = G(z)_s \in \{0,1\}^R$ be the s^{th} output string of G when G is given input z.

In turn $i \in [c']$, the verifier V' chooses a random seed $s \in [\bar{R}]$ and sends $G^{x,\pi_i}(s)$ to the prover, where x is the input and π_i is the interaction communicated between the parties up to turn i of the interaction, padded with zeroes to be of length precisely $N - n$ (for convience we denote $\pi_1 = 0^{N-n}$). Then it applies the same final predicate to the interaction as V. In other words, continuing our notation for interactions as functions of random coins, we have that

$$\langle V', P, x \rangle (s_1, \ldots, s_{c'}) = \langle V, P, x \rangle (G^{x,\pi_1}(s_1), \ldots, G^{x,\pi_{c'}}(s_{c'})) \,.$$

The running time of V' is larger than that of V, but we will carefully account for it later on. For now it suffices to observe that the only runtime overhead of V' on top of V comes from computing the generator G in each of the c' turns, rather than just sending random coins.

Analysis: Soundness of V'. For convenience, we will think of any probabilistic polynomial-time al-

gorithm $\mathbf{P}$ as an efficiently samplable distribution over *deterministic* polynomial-time algorithms, obtained in the natural way (*i.e.*, by randomly choosing coins in advance and running the algorithm with the fixed chosen coins).

Consider an arbitrary probabilistic polynomial-time algorithm $\mathbf{P}$. Our analysis uses the following hybrid argument. Let V_0 be the original verifier, and for $i \in [c']$ let V_i be the verifier in which in the first i turns, the verifier uses $\log(\bar{R})$ coins as above, instead of $R(n)$ random coins; that is, for any fixed $P \sim \mathbf{P}$,

$$\langle V_i, P, x \rangle (s_1, \ldots, s_i, r_{i+1}, \ldots, r_{c'}) = \langle V, P, x \rangle (G^{x,\pi_1}(s_1), \ldots, G^{x,\pi_i}(s_i), r_{i+1}, \ldots, r_{c'}) .$$

Observe that $V_0 = V$ and that $V_{c'} = V'$. For any fixed prover P and $1 \leq i < j \leq c'$, denote by $P^{i \ldots j}$ the partial prover strategy of P that refers only to rounds $i, i+1, \ldots, j$. [43] That is, we think of the prover as a sequence of c' functions, where the i^{th} function maps a transcript in the i^{th} round (which is of length $i \cdot R + (i-1) \cdot T$) to the prover's response (which is of length T); then, $P^{i \ldots j}$ is simply a subsequence of the c' functions that define P. We use the notation $P^{i \ldots j} \sim \mathbf{P}$ to denote the random variable that is obtained by sampling $P \sim \mathbf{P}$ and outputting the partial prover strategy $P^{i \ldots j}$. And for two partial prover strategies $P^{1 \ldots i-1}$ and $P^{i \ldots c'}$, we denote by $P^{1 \ldots i-1} \circ P^{i \ldots c}$ the (complete) prover strategy that is obtained by combining both partial strategies in the obvious way (*i.e.*, by simply concatenating the two sequences of functions).

Our main definition for the hybrid argument is a sequence of hybrids that involves both a partial replacement of the random coins, and a partial replacement of the "existential" prover strategy in the soundness condition (*i.e.*, that there does not exist an all-powerful prover that convinces the verifier) with a partial strategy chosen according to $\mathbf{P}$. In more detail, for any fixed $i \in [c']$ we denote

$$p_{x,i} = \max_{P^{i+1 \ldots c'}} \left\{ \Pr_{s_1, \ldots, s_i, r_{i+1}, \ldots, r_{c'}, P^{1 \ldots i} \sim \mathbf{P}} \left[\left\langle V_i, P^{1 \ldots i} \circ P^{i+1, \ldots, c'}, x \right\rangle (s_1, \ldots, s_i, r_{i+1}, \ldots, r_{c'}) = 1 \right] \right\} .$$

The perfect completeness of V' follows immediately from the perfect completeness of V, since V' simply simulates V with pseudorandom choices of coins. Thus, we focus on establishing the soundness of V'. To do so, we first argue that:

[43] Note that we are now referring to *rounds* rather than to *turns*. That is, the interaction consists of c turns (where a turn is when one of the players speaks) and of $c' = \lceil c/2 \rceil$ rounds (where a round consists of two turns, except possibly the last round).

Fact 13.7.6. *For any fixed $x \notin L$ such that*

$$\Pr_{P\sim\mathbf{P},s_1,\ldots,s_{c'}}[\langle V', P, x\rangle(s_1,\ldots,s_{c'}) = 1] > .4\,, \tag{13.7.1}$$

there exists $i \in [c']$ such that $p_{x,i} - p_{x,i-1} > 1/20c'$.

Proof. Fix $x \notin L$ such that Eq. (13.7.1) holds. Observe that $p_{x,0} \leq 1/3$, by the soundness of the original protocol V (which holds for every fixed x and an all-powerful P). On the other hand, we have that $p_{x,c'} = \Pr_{P\sim\mathbf{P},s_1,\ldots,s_{c'}}[\langle V', P, x\rangle(s_1,\ldots,s_{c'}) = 1]$, and thus $p_{x,c'} > .4$ by Eq. (13.7.1). Since $p_{x,c'} - p_{x,0} = \sum_{i\in[c']} p_{x,i} - p_{x,i-1}$, for some $i \in [c']$ it holds that $p_{x,i} - p_{x,i-1} > (p_{x,c'} - p_{x,0})/c' > 1/20c'$. □

The main claim in the analysis is the following:

Claim 13.7.7. *For any $i \in [c']$, the probability over $x \sim \mathbf{x}$ that $x \notin L$ and*

$$p_{x,i} > p_{x,i-1} + 1/20c'$$

is negligible.

Since the proof of Claim 13.7.7 is quite involved, we defer it for a moment, and first complete the argument assuming that Claim 13.7.7 is true.

By combining Fact 13.7.6 and Claim 13.7.7, we deduce that for every probabilistic polynomial-time algorithm $\mathbf{P}$, with probability $1 - \mathsf{neg}(|x|)$ over $x \sim \mathbf{x}$, if $x \notin L$ then $\Pr_{P\sim\mathbf{P},s_1,\ldots,s_{c'}}[\langle V', P, x\rangle(s_1,\ldots,s_{c'}) = 1] \leq .4$. (Intuitively, this establishes that V' is an argument system that uses few random coins.)

From V' to a deterministic doubly efficient argument system. Given any input x, the total number of possible messages from V' to a prover (across all turns) is $\bar{R}^{c'}$, corresponding to a choice of seeds $\bar{s} \in [\bar{R}]^{c'}$. Our deterministic verifier V'' expects the prover to send a corresponding transcript for each choice of $\bar{s}$. It then:

1. Verifies that the sent transcripts are consistent with a prover strategy (*i.e.*, that the prover did not respond to two identical prefixes in different ways).
2. For each transcript corresponding to a choice of $\bar{s}$, it verifies that the pseudorandom coins in each message (which are part of the transcript) are what one obtains by applying G to the transcript at that point, using the corresponding seed (which is part of $\bar{s}$).
3. Computes the average acceptance probability of V' over all choices of $\bar{s}$.

Recall that there is an efficient prover P such that for any $x \in L$ it holds that $\Pr_{r_1,\ldots,r_{c'}}[\langle V, P, x\rangle\,(r_1,\ldots,r_{c'}) = 1] = 1$. When interacting with P, the verifier V' simply uses a pseudorandom coins instead of random ones, and thus $\Pr_{s_1,\ldots,s_{c'}}[\langle V', P, x\rangle\,(s_1,\ldots,s_{c'}) = 1] = 1$. The honest prover for the verifier V'' enumerates over $\bar{s} \in [\bar{R}]^{c'}$, and for every choice of $\bar{s}$ it simulates the corresponding interaction with P, while computing the generator G at each round. Since P runs in time $\mathrm{poly}(T)$ and G can be computed in time polynomial in $\bar{T} \leq \mathrm{poly}(T)$, the honest prover for V'' runs in time $\mathrm{poly}(T)$.

Now, the verifier V'' inherits its soundness immediately from that of V'.[44] Relying on the fact that the soundness holds against all polynomial-time algorithms $\mathbf{P}$, we further argue that the soundness error of V'' is not only .4 but actually $\mathsf{neg}(n)$:

Claim 13.7.8. *For every probabilistic polynomial-time algorithm* $\mathbf{P}$ *with probability* $1 - \mathsf{neg}(n)$ *over an* n*-bit* $x \sim \mathbf{x}$*, if* $x \notin L$ *then* $\Pr_{P\sim\mathbf{P}}[V(x, P(x)) = 1] < \mathsf{neg}(n)$.

Proof. Assume towards a contradiction that for some $\mathbf{P}$ and polynomial $p(n)$ there are infinitely many $n \in \mathbb{N}$ such that, with non-negligible probability over an n-bit $x \sim \mathbf{x}$ it holds that $x \notin L$ and $\Pr_{P\sim\mathbf{P}}[V''(x, P(x)) = 1] \geq 1/p(n)$.

Consider the prover $\mathbf{P}'$ that on input x runs $\mathbf{P}$ for $t = p(n)^2$ times to obtain candidate proofs $\pi_1, \ldots, \pi_t$, for each $i \in [t]$ checkes whether $V''(x, \pi_i) = 1$, and if it finds π_i for which the latter holds then it prints this π_i (otherwise it prints some fixed default proof). Note that $\mathbf{P}'$ runs in polynomial time, and for every x such that $\Pr_{P\sim\mathbf{P}}[V''(x, P(x)) = 1] \geq 1/p(n)$ we have that $\Pr_{P'\sim\mathbf{P}'}[V''(x, P'(x)) = 1] \geq .99$. Thus, there is a polynomial-time prover $\mathbf{P}'$ and infinitely many $n \in \mathbb{N}$ such that with non with non-negligible probability over an n-bit $x \sim \mathbf{x}$ it holds that $x \notin L$ and $\Pr_{P\sim\mathbf{P}}[V''(x, P(x)) = 1] > .4$, contradicting the soundness of V''. ■

We now bound the running time of V''. Recall that the prover sends $[\bar{R}]^{c'}$ transcripts to V'', corresponding to all possible choices of seeds $\bar{s} \in [\bar{R}]^{c'}$ by V'. We assume that the prover sends the transcripts in prefix-order of $\bar{s}$. For each prefix $s_1, \ldots, s_i$ of $\bar{s}$, the verifier checks that all transcripts corresponding to that prefix are identical (*i.e.*, verifies that the prover strategy is consistent); and then computes the set of pseudorandom strings obtained by applying G to the foregoing transcript with all possible choices of s_{i+1}, "in a batch"; by Theorem 13.7.1, for each prefix this can be done in time $\bar{T} \cdot K + \mathrm{poly}(K) \leq T \cdot R^{O(1/\eta)}$. In the end, it computes the original V on each of the $|\bar{R}|^{c'} = R^{O(c/\eta)}$ choices of seeds. The final running time of V'' is thus $T \cdot R^{O(c/\eta)}$.

[44] To see this, note that if the prover sends consistent answers to all message sequences then those answers yield a prover strategy that could have been used in the interaction with V'.

Finally, to complete the proof the only missing piece is to prove Claim 13.7.7.

Proof of Claim 13.7.7. Fix $i \in [c']$ and assume that with non-negligible probability over $x \sim \mathbf{x}$ we have that $p_{x,i} > p_{x,i-1} + 1/20c'$. For any fixed $\bar{p} = P^{1\ldots i-1}$ and $\bar{s} = s_1, \ldots, s_{i-1}$, the interaction in the first $i-1$ rounds between any prover whose partial strategy in these rounds is $\bar{p}$ and any verifier that behaves like V_{i-1} in these rounds is also fixed (*i.e.*, it is a deterministic function of $\bar{p}$ and $\bar{s}$); we denote the transcript of this interaction by $\pi_{\bar{p},\bar{s}}$. We denote by $(\mathbf{x}, \boldsymbol{\pi})$ the distribution that is obtained by sampling $x \sim \mathbf{x}$ and $\bar{p} = P^{1\ldots i-1} \sim \mathbf{P}$ and $\bar{s} \in ([\bar{R}])^{i-1}$ and outputting $(x, \pi_{\bar{p},\bar{s}})$. By our assumptions about $\mathbf{P}$ and $\mathbf{x}$ and the fact that G runs in polynomial time, the distribution $(\mathbf{x}, \boldsymbol{\pi})$ is samplable in polynomial time.

Fixing the first $i-1$ rounds of interaction. For every fixed $(x, \pi_{\bar{s},\bar{p}})$, denote

$$(p_{x,i-1}|\pi_{\bar{s},\bar{p}}) = \max_{P^{i\ldots c'}} \left\{ \Pr_{r_i, r_{i+1}, \ldots, r_{c'}} \left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, r_i, r_{i+1}, \ldots, r_{c'}) \right] \right\} ,$$

$$(p_{x,i}|\pi_{\bar{s},\bar{p}}) = \max_{P^{i+1\ldots c'}} \left\{ \Pr_{s_i, r_{i+1}, \ldots, r_{c'}, P^i \sim \mathbf{P}} \left[\left\langle V_i, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, s_i, r_{i+1}, \ldots, r_{c'}) \right] \right\} ,$$

and observe that:

Claim 13.7.9. *We have that* $p_{x,i} = \mathbb{E}_{\bar{s},\bar{p}} \left[p_{x,i} | \pi_{\bar{s},\bar{p}} \right]$ *and* $p_{x,i-1} = \mathbb{E}_{\bar{s},\bar{p}} \left[p_{x,i-1} | \pi_{\bar{s},\bar{p}} \right]$.

Proof. Note that in $p_{x,i}$ and in $(p_{x,i}|\pi_{\bar{s},\bar{p}})$ (resp., in $p_{x,i-1}$ and $(p_{x,i-1}|\pi_{\bar{s},\bar{p}})$) the maximum is over functions $P^{i+1\ldots c'}$ (resp., $P^{i\ldots c'}$) that take $\pi_{\bar{s},\bar{p}}$ as part of their input. Thus, first choosing $\pi_{\bar{s},\bar{p}}$ from some distribution and then taking the maximum-achieving function is identical to first taking the maximum-achieving function and then choosing $\pi_{\bar{s},\bar{p}}$ from the same distribution.[45] □

We call a pair $(x, \pi_{\bar{s},\bar{p}})$ good if $(p_{x,i}|\pi_{\bar{s},\bar{p}}) > (p_{x,i-1}|\pi_{\bar{s},\bar{p}}) + 1/40c'$, and argue that:

Claim 13.7.10. *With non-negligible probability over* $(x, \pi_{\bar{s},\bar{p}}) \sim (\mathbf{x}, \boldsymbol{\pi})$ *we have that* $(x, \pi_{\bar{s},\bar{p}})$ *is good.*

Proof. By our assumption, with non-negligible probability over $x \sim \mathbf{x}$ we have that $p_{x,i} > p_{x,i-1} +$

[45] In other words, we use the fact that for every $\mathcal{D}$ and $\mathcal{R}$ and $\nu \colon \mathcal{R} \to \mathbb{R}$ it holds that

$$\max_{f \colon \mathcal{D} \to \mathcal{R}} \left\{ \mathop{\mathbb{E}}_{r \in \mathcal{D}} [\nu(f(r))] \right\} = \mathop{\mathbb{E}}_{r \in \mathcal{D}} \left[\max_{f \colon \mathcal{D} \to \mathcal{R}} \{\nu(f(r))\} \right] = \mathop{\mathbb{E}}_{r \in \mathcal{D}} [\nu(f^*(r))] ,$$

where f^* is the function that maps any $r \in \mathcal{D}$ to $t = f(r)$ such that $\nu(t)$ is maximal.

$1/20c'$. Now, let $\Delta_{\bar{s},\bar{p}} = (p_{x,i}|\pi_{\bar{s},\bar{p}}) - (p_{x,i-1}|\pi_{\bar{s},\bar{p}})$, and note that

$$\mathop{\mathbb{E}}_{\bar{s},\bar{p}}[\Delta_{\bar{s},\bar{p}}] = \mathop{\mathbb{E}}_{\bar{s},\bar{p}}\left[p_{x,i}|\pi_{\bar{s},\bar{p}}\right] - \mathop{\mathbb{E}}_{\bar{s},\bar{p}}\left[p_{x,i-1}|\pi_{\bar{s},\bar{p}}\right] = p_{x,i} - p_{x,i-1} > 1/20c' \ ,$$

where the second equality above relies on Claim 13.7.9. Thus, if we'd have that $\Pr_{\bar{s},\bar{p}}[\Delta_{\bar{s},\bar{p}} > 1/40c'] < n^{-\omega(1)}$, then we'd also have $\mathbb{E}_{\bar{s},\bar{p}}[\Delta_{\bar{s},\bar{p}}] \leq n^{-\omega(1)} \cdot (1/40c') + (1/40c') < 1/20c'$, a contradiction. □

Obtaining an $\mathsf{AMTIME}_2^{[\rightleftharpoons c]}$ distinguisher. Fixing a good $(x, \pi_{\bar{s},\bar{p}})$, we denote

$$p(r_i) = \max_{P^{i\ldots c'}}\left\{\Pr_{r_{i+1},\ldots,r_{c'}}\left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x\right\rangle(\bar{s}, r_i, \ldots, r_{c'}) = 1\right]\right\} \ ,$$

and argue that:

Claim 13.7.11. *The following two statements hold:*

1. $\mathop{\mathbb{E}}_{r_i\in\{0,1\}^R}[p(r_i)] = (p_{x,i-1}|\pi_{\bar{s},\bar{p}}) \ .$
2. $\mathop{\mathbb{E}}_{s_i\in[\bar{R}]}[p(G^{x,\pi_{\bar{s},\bar{p}}}(s_i))] \geq (p_{x,i}|\pi_{\bar{s},\bar{p}}) \ .$

Proof. For the first item, note that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{r_i\in\{0,1\}^R}[p(r_i)] \\
&= \mathop{\mathbb{E}}_{r_i\in\{0,1\}^R}\left[\max_{P^{i\ldots c'}}\left\{\Pr_{r_{i+1},\ldots,r_{c'}}\left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x\right\rangle(\bar{s}, r_i, \ldots, r_{c'}) = 1\right]\right\}\right] \\
&= \max_{P^{i\ldots c'}}\left\{\Pr_{r_i,r_{i+1},\ldots,r_{c'}}\left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x\right\rangle(\bar{s}, r_i, \ldots, r_{c'}) = 1\right]\right\} \\
&= (p_{x,i-1}|\pi_{\bar{s},\bar{p}}) \ ,
\end{aligned}$$

where the second inequality is because (as in the proof of Claim 13.7.10) the maximum is over functions $P^{1\ldots c'}$ that take r_i as part of their input (and thus first drawing a random r_i and then choosing a maximum-achieving function is identical to first choosing a maximum-achieving function and then drawing a random r_i).

For the second item, the argument is a bit more subtle, as follows:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{s_i\in[\bar{R}]}\left[p(G^{x,\pi_{\bar{s},\bar{p}}}(s_i))\right] \\
&= \mathop{\mathbb{E}}_{s_i\in[\bar{R}]}\left[\max_{P^{i\ldots c'}}\left\{\Pr_{r_{i+1},\ldots,r_{c'}}\left[\left\langle V_{i-1},\bar{p}\circ P^{i\ldots c'},x\right\rangle(\bar{s},G^{x,\pi_{\bar{s},\bar{p}}}(s_i),r_{i+1},\ldots,r_{c'})=1\right]\right\}\right] \\
&= \max_{P^{i\ldots c'}}\left\{\Pr_{s_i,r_{i+1},\ldots,r_{c'}}\left[\left\langle V_{i-1},\bar{p}\circ P^{i\ldots c'},x\right\rangle(\bar{s},G^{x,\pi_{\bar{s},\bar{p}}}(s_i),r_{i+1},\ldots,r_{c'})=1\right]\right\} \\
&= \max_{P^{i\ldots c'}}\left\{\Pr_{s_i,r_{i+1},\ldots,r_{c'}}\left[\left\langle V_{i},\bar{p}\circ P^{i\ldots c'},x\right\rangle(\bar{s},s_i,r_{i+1},\ldots,r_{c'})=1\right]\right\} \\
&\geq \max_{P^{i+1\ldots c'}}\left\{\Pr_{s_i,r_{i+1},\ldots,r_{c'},P^i\sim\mathbf{P}}\left[\left\langle V_{i},\bar{p}\circ P^{i\ldots c'},x\right\rangle(\bar{s},s_i,r_{i+1},\ldots,r_{c'})=1\right]\right\} \\
&= (p_{x,i}|\pi_{\bar{s},\bar{p}})\ ,
\end{aligned}
$$

where the main difference from the proof of the first item is the inequality, which asserts that taking the maximum-achieving prover strategy in round i can only increase the acceptance probability compared to choosing $P_i\sim\mathbf{P}$. [46] □

Relying on Claim 13.7.11, for the fixed good $(x,\pi_{\bar{s},\bar{p}})$ we have that

$$
\mathop{\mathbb{E}}_{s_i\in[\bar{R}]}[p(G^{x,\pi_{\bar{s},\bar{p}}}(s_i))]-\mathop{\mathbb{E}}_{r_i\in\{0,1\}^R}[p(r_i)]\geq(p_{x,i}|\pi_{\bar{s},\bar{p}})-(p_{x,i-1}|\pi_{\bar{s},\bar{p}})>1/40c'\ .
$$

The foregoing assertion implies that the real-valued function $p(\cdot)$ behaves differently on the pseudorandom distribution $G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]})$ and on the uniform distribution $\mathbf{u}_T$. To obtain a Boolean-valued distinguisher (rather than a real-valued one such as $p(\cdot)$), and furthermore a Boolean-valued distinguisher that is computable by an $\mathsf{AMTIME}^{[\rightleftharpoons c]}$ protocol, we rely on the following claim: It asserts that if two real-valued RVs $\mathbf{x}$ and $\mathbf{y}$ in $[0,1]$ have expectations that differ by $\varepsilon>0$, then there are two thresholds ℓ' and $\ell'+\Theta(\varepsilon)$ such that the probability that $\mathbf{x}$ exceeds $\ell'+\Theta(\varepsilon)$ is noticeably higher than the probability that $\mathbf{y}$ exceeds ℓ'.

Lemma 13.7.12. *Let $\mathbf{x},\mathbf{y}$ be two random variables taking values from $[0,1]$ and $\varepsilon\in(0,1)$ such that*

[46] To see that the third equality above holds, note the following. In the upper row we are referring to the verifier V_{i-1}, which uses the i^{th} random string given to it as-is, and are giving V_i the string $G^{x,\pi_{\bar{s},\bar{p}}}(s_i)$ for a random s_i. In the bottom row we're referring to the verifier V_{i-1}, which applies $G^{x,\pi_{\bar{s},\bar{p}}}$ to the i^{th} random string given to it, and are giving V_{i-1} a random s_i. Thus, in both cases the random string used in the i^{th} round is $G^{x,\pi_{\bar{s},\bar{p}}}(s_i)$ for a random s_i.

$\varepsilon^{-1} \in \mathbb{N}$. *If* $\mathsf{E}[\mathbf{x}] - \mathsf{E}[\mathbf{y}] > \varepsilon$, *then there exists* $j \in [4/\varepsilon]$ *such that*

$$\Pr[\mathbf{x} \geq (\varepsilon/4) \cdot (j+1)] - \Pr[\mathbf{y} \geq (\varepsilon/4) \cdot j] > \varepsilon/2.$$

The proof of Lemma 13.7.12 is elementary but tedious, so for convenience we defer it to Section 13.9. Now, denote $\varepsilon = 1/40c'$ and consider the following promise problem:

$$\begin{aligned} \mathsf{Y} &= \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \geq \tau + \varepsilon/4\right\} \\ \mathsf{N} &= \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \leq \tau\right\} \ . \end{aligned}$$

Observe that the problem (Y, N) can be decided in $\mathsf{prAMTIME}_2^{[\rightleftharpoons c]}[O(n)]$, since the verifier can run the original protocol V_{i-1} from the definition of $p(\cdot)$ for $O(1)$ times in parallel, estimate the acceptance probability of V_{i-1} with the given prover up to accuracy $\varepsilon/8$ and with high probability, and accept if and only if this probability is more than $\tau + \varepsilon/8$. Note that the runtime of this protocol is linear, because $\varepsilon = \Omega(1)$ and the input size to this problem is $O(T)$.

Now, by instantiating Lemma 13.7.12 with the RVs $\mathbf{x} = p(G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]}))$ and $\mathbf{y} = p(\mathbf{u}_R)$, for some $\tau \in \{0, \varepsilon/4, 2\varepsilon/4, \ldots, 1\}$ it holds that

$$\Pr[(x, \pi_{\bar{s},\bar{p}}, G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]}, \tau)) \in \mathsf{Y}] > 1 - \Pr[(x, \pi_{\bar{s},\bar{p}}, \mathbf{u}_R, \tau)) \in \mathsf{N}] + \varepsilon/2 \ .$$

For any fixed (x, π) and $\tau \in \{0, \varepsilon/4, 2\varepsilon/4, \ldots, 1\}$ we define the following procedure $D = D_{x,\pi,\tau}$: Given $r \in \{0,1\}^R$ (which we think of as coming either from the uniform distribution or from the pseudorandom distribution) the procedure creates the string $z = (x, \pi, r, \tau)$ and solves the promise problem (Y, N) on input z. Note that indeed $D \in \mathsf{prAMTIME}_2^{[\rightleftharpoons c]}[O(n)]$, and that

$$\Pr[D(G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]})) = 1] > \Pr[D(\mathbf{u}_R) = 1] + \varepsilon/2 \ ,$$

where the inequality relies on the fact that $\Pr[D(\mathbf{u}_R) = 1] = 1 - \Pr[D(\mathbf{u}_R) = 0 \geq 1 - \Pr[\mathbf{u}_T \in \mathsf{N}]$.

The reconstruction argument. To obtain a contradiction, we show an algorithm that runs in time $\bar{T} \cdot K^{\beta}$, where $\bar{T} = T \cdot K$, and with non-negligible probability over the polynomial-time samplable distribution $(x, \pi) \sim (\mathbf{x}, \boldsymbol{\pi})$ manages to approximately print $f(x, \pi)$ (with high probability over its random coins).

Consider an algorithm F_0 gets input (x, π) and randomly chooses $\tau \in \{0, \varepsilon/4, 2\varepsilon/4, \ldots, 1\}$.

Then F_0 runs the reconstruction Rec with input (x, π), giving it oracle access to $D^{\tau,x,\pi}$. Specifically, whenever Rec queries $D^{\tau,x,\pi}$ on input r, the algorithm F_0 queries D^{τ} on input (x, π, r, τ) and returns the corresponding answer. The algorithm F_0 runs in time

$$\underbrace{\tilde{O}(K^{1+\beta'})}_{\text{\#queries}} \cdot \underbrace{T}_{\text{length of a query to } D^{\tau}} + \underbrace{\bar{T} \cdot K^{\beta'}}_{\text{add'l runtime of Rec}} = \tilde{O}(\bar{T} \cdot K^{\beta'})$$

and assuming that the guess of τ' was correct, with high probability F_0 prints a string that agrees with $f(x, \pi)$ on $1 - \alpha'$ of the bits.

To construct an algorithm F that succeeds with high probability, we run F_0 for $O(c')$ times, such that with high probability at least one iteration successfully printed a string that agrees with $f(x, \pi)$ on $1 - \alpha'$ of the bits. Then, for each of the $O(c')$ candidate strings, the algorithm F estimates the agreement of this string with $f(x, \pi)$ up to a sufficiently small constant error, by randomly sampling output bits and computing the corresponding bits of $f(x, \pi)$ (recall that each bit can be computed in time $\bar{T}$). The running time of F is at most $\tilde{O}(\bar{T} \cdot K^{\beta'}) < \bar{T} \cdot K^{\beta}$, and with high probability it succeeds in outputting a string that agrees with f on at least $1 - \alpha$ of the bits. □

Having proved Claim 13.7.7, this concludes the proof of Theorem 13.7.5.

Remark 13.7.13 (handling imperfect completeness)**.** The proof of Theorem 13.7.5 implies similar derandomization for $\mathsf{delP}^{[\rightleftharpoons c]}$ protocols with *imperfect completeness* that use a small number of random coins. To see this, observe that the well-known transformation of AM protocols into protocols with perfect completeness [FGM+89] yields the following: Any $\mathsf{delP}^{[\rightleftharpoons c]}$ protocol in which the verifier uses only R coins can be simulated by a $\mathsf{delP}^{[\rightleftharpoons c]}$ protocol with perfect completeness such that the new protocol has the same number of rounds, the verifier uses $\bar{R} = \tilde{O}(R)$ random coins, its communication complexity and running time increase by a multiplicative factor of $O(\bar{R})$, and the prover is still efficient but it is now *probabilistic*.[47]

Since we think of R as small in the result above (indeed, we will use this result with $R = O(\log(T))$ in Theorem 13.7.16 and Corollary 13.7.20), we can start from a protocol with imperfect completeness, simulate it by a protocol with similar running time, and appeal to Theorem 13.7.5 as a black-box. Indeed, the only gap is that the honest prover in the resulting $\mathsf{delP}^{[\rightleftharpoons c]}$ protocol is

[47] These properties are not explicitly stated in the original work but they readily follow from the proof. Specifically, in the original proof the message lengths by the verifier and the prover are coupled, but the proof only relies on the error being smaller than $1/R$; and the original proof shows that for every $x \in L$, with high probability over choice of initial strings by the prover (as a basis for simulating copies of the protocol in parallel) the verifier to accept with probability 1.

now probabilistic rather than deterministic.

Remark 13.7.14 (argument systems for NP-relations)**.** The honest prover in the argument system that is constructed in the proof of Theorem 13.7.5 is very similar to the original honest prover, and in particular has almost the same time complexity. (This is because the new prover enumerates over all possible seeds for the targeted PRG, across all rounds, and for each choice it simulates the verifier's interaction with the original prover using the chosen seeds to generate pseudorandom coins.) One implication of this fact is that our results extend naturally to constant-round probabilistic proof systems in which the honest prover gets a witness as auxiliary input.

In more detail, let R be a relation, let L_R be the decision problem defined by R, and let Π be a probabilistic proof system for L_R with c turns of interaction such that the verifier in Π runs in time T, and the honest prover in Π runs in time $\text{poly}(T)$ when given a witness for the input (in the relation R).[48] Then, under the same assumption as in Theorem 13.7.5, our proof gives a deterministic argument system for L_R in which the verifier runs in time $T^{1+\varepsilon}$, soundness is precisely as in Definition 13.3.8, and the honest prover runs in time $\text{poly}(T)$ when given a witness (in the relation R) for the input.

A stronger result for doubly efficient proof systems with a universal prover

We now argue that in the special case of doubly efficient proof systems that have an efficient universal prover, we can derandomize such systems under a hypothesis that is weaker than the one in Theorem 13.7.5. Specifically, we require hardness only against probabilistic algorithms that have oracle access to $\mathsf{delP}^{[\rightleftharpoons c]}$, rather than to $\mathsf{AMTIME}^{[\rightleftharpoons c]}$. (We will use this generic claim in the proof of Theorem 13.1.4, since the proof system for #SAT indeed has an efficient universal prover.)

Theorem 13.7.15 (derandomizing constant-round doubly efficient proof systems with an efficient universal prover into deterministic doubly efficient argument systems)**.** *For every $\alpha, \beta \in (0,1)$ there exists $\eta > 0$ such that the following holds. Let $c \in \mathbb{N}$ be a constant, let $T(n)$ be a polynomial, let $R(n) < T(n)$ be time-computable, and let $N(n) = n + c \cdot T(n)$ and $K(n) = R(n)^{1/\eta}$. Assume that the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T} = T \cdot K$ with oracle access to*

[48] That is, we consider the interaction between the verifier and the honest prover when the former is given input x and the latter is given input $(x, w) \in R$. Note that any such relation is in $\mathsf{MATIME}[\text{poly}(T)]$, since a verifier can guess w and simulate the interaction with the honest prover.

$\mathsf{pr\text{-}delP}^{[\rightleftharpoons c]}[n]$ *on inputs of length* $O(T)$*. Then,*

$$\mathsf{delP}^{[\rightleftharpoons c]}_{\mathsf{uni}}[T, R] \subseteq \mathsf{NARG}[T \cdot R^{O(c/\eta)}] \ ,$$

where the O-notation hides a universal constant, and $\mathsf{delP}^{[\rightleftharpoons c]}_{\mathsf{uni}}[T, R]$ *denotes all problems solvable by* $\mathsf{delP}^{[\rightleftharpoons c]}[T, R]$ *systems that, for every constant* $\mu > 0$*, have a* μ*-approximate universal prover running in time* $\mathrm{poly}(T)$*.*

Proof. The proof is very similar to that of Theorem 13.7.5, the only difference being that the reconstruction algorithm in Claim 13.7.7 can now only access a $\mathsf{pr\text{-}delP}^{[\rightleftharpoons c]}[n]$ oracle rather than a $\mathsf{AMTIME}_2^{[\rightleftharpoons c]}[n]$ oracle.

Recall that in the proof of Claim 13.7.7, the algorithm uses an oracle that solves the following promise problem, which is defined with respect to a parameter τ:

$$\mathsf{Y} = \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \geq \tau + \varepsilon/4\right\}$$
$$\mathsf{N} = \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \leq \tau\right\} \ .$$

Our goal is to argue that $(\mathsf{Y}, \mathsf{N}) \in \mathsf{pr\text{-}delP}^{[\rightleftharpoons c]}[n]$, by arguing that there is an efficient honest prover. (The rest of the proof then continues without change.)

The key observation is that our current assumption asserts the existence of an *efficient* prover that, on any x and $(\pi_{\bar{s},\bar{p}}, r_i)$, almost maximizes the residual acceptance probability of the protocol when the prefix of the transcript is fixed to $(\pi_{\bar{s},\bar{p}}, r_i)$. For any constant μ, we denote the μ-approximate universal prover by P_μ, and on a fixed $\bar{x} = (x, \pi_{\bar{s},\bar{p}}, r_i)$, we denote by $\nu = \nu_{\bar{x}}$ the maximal acceptance probability of the residual protocol, across all provers.

The verifier for (Y, N) is identical to that in the original proof of Claim 13.7.7; that is, the verifier simulates the residual protocol for $\mathrm{poly}(1/\varepsilon)$ times in parallel, and accepts if and only if the average probability across simulations, denoted $\tilde{\nu}$, satisfies $\tilde{\nu} \geq \tau + \varepsilon/8$. When $(\bar{x}, \tau) \in \mathsf{N}$, for any prover, with high probability we have that $\tilde{\nu} < \tau + \varepsilon/8$. On the other hand, when $(\bar{x}, \tau) \in \mathsf{Y}$, a prover that simulates $P_{\varepsilon/16}$ on the $\mathrm{poly}(1/\varepsilon)$ parallel instantiations of the protocol yields $\tilde{\nu} \geq \tau + \varepsilon/8$, with high probability. Thus, $(\mathsf{Y}, \mathsf{N}) \in \mathsf{delP}^{[\rightleftharpoons c]}[n]$ as we wanted. ■

A deterministic argument system for #SAT with runtime $2^{\varepsilon \cdot n}$

We now state and prove Theorem 13.1.4, as a particularly appealing special case of Theorem 13.7.15. Specifically, we show that under strong hardness assumptions, we can solve #SAT by a deterministic doubly efficient argument system running in time $2^{\varepsilon \cdot n}$, for an arbitrarily small $\varepsilon > 0$.

Theorem 13.7.16 (a deterministic argument system for #SAT with runtime $2^{\varepsilon \cdot n}$)**.** *For every $\alpha, \beta \in (0,1)$ there exists $\eta > 0$ such that the following holds. Assume that for every constant $c \in \mathbb{N}$ and logarithmic function $\ell(n) = O(\log(n))$, the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T} = T \cdot K$ with oracle access to $\mathsf{pr\text{-}delP}^{[\rightleftharpoons c]}[n]$ on inputs of length $O(T)$, where*

$$\begin{aligned} T(n) &= O(n) \,, \\ N(n) &= n + c \cdot T(n) \,, \\ K(n) &= \ell(n)^{1/\eta} \,. \end{aligned}$$

Then, for every $\varepsilon > 0$ there exists a deterministic verifier V that gets as input an n-bit formula Φ of size at most $2^{o(n)}$, runs in time $2^{\varepsilon \cdot n}$, and satisfies the following:

1. *There exists an algorithm that gets input Φ, runs in time $2^{O(n)}$, and prints a proof π such that $V(\Phi, \pi) = \#\mathsf{SAT}(\Phi)$.*
2. *For every probabilistic algorithm P running in time $2^{O(n)}$ and every sufficiently large $n \in \mathbb{N}$, the probability that $P(1^n)$ outputs an n-bit formula Φ of size $2^{o(n)}$ and proof π such that $V(\Phi, \pi) \notin \{\bot, \#\mathsf{SAT}(\Phi)\}$ is $2^{-\omega(n)}$.*

Proof. Let $\varepsilon' > 0$ be a sufficiently small constant. Using Theorem 13.3.17 with a sufficiently large constant $k \in \mathbb{N}$ and with $\delta > 0$ such that $\delta/(1-\delta) = 1/k$, we have an $\mathsf{delP}^{[\rightleftharpoons 2k+1]}_{\mathsf{uni}}[2^{\varepsilon' \cdot n}]$ protocol for counting the number of satisfying assignments of an n-bit formula of size $2^{o(n)}$, which uses at most $O(n)$ random coins.[49]

By a padding argument, we think of the input as of size $\bar{n} = 2^{\varepsilon' \cdot n}$ and of the protocol as running in linear time with logarithmically many coins. Also, we consider the $2n = O(\log \bar{n})$ Boolean protocols $U_U, \ldots, P_n, \bar{U}_1, \ldots, \bar{U}_n$, where each U_i is a protocol for proving that the i^{th} bit of $\#\mathsf{SAT}(\Phi)$ is 1 and each $\bar{U}_i$ is a protocol for proving that the i^{th} bit of $\#\mathsf{SAT}(\Phi)$ is 0.

We apply Theorem 13.7.15 to each of $2n$ protocols, with parameters $T(\bar{n}) = O(\bar{n})$ and $R(\bar{n}) = O(\log \bar{n})$, to obtain a sequence of $2n$ deterministic verifiers $D_1, \ldots, D_n, \bar{D}_1, \ldots, \bar{D}_n$ each running

[49] Note that Theorem 13.3.17 yields an $\mathsf{MATIME}^{[\rightleftharpoons 2k]}$ protocol that has a universal prover running in time $2^{O(n)}$. In particular, such a protocol can be simulated by a $\mathsf{delP}^{[\rightleftharpoons 2k+1]}$ protocol with the same verifier running time.

in time $\tilde{O}(T)$. Given a formula Φ, our verifier V expects to receive from the prover a value $\rho \in \{0,1\}^n$ and n witnesses $w_1, \ldots, w_n \in \{0,1\}^{\tilde{O}(\bar{n})}$ such that for every $i \in [n]$ it holds that $\begin{cases} D_i(w_i) = 1 & \rho_i = 1 \\ \bar{D}_i(w_i) = 1 & \rho_i = 0 \end{cases}$. Whenever this happens V outputs ρ, otherwise it outputs $\perp$.

For every algorithm P running in time $2^{O(n)}$, by a union-bound, the probability over $\Phi \sim \boldsymbol{\Phi}$ that P outputs a proof that falsely convinces some D_i or $\bar{D}_i$ is negligible in N. Also, the running time of V is $\tilde{O}(n \cdot \bar{n}) < 2^{\varepsilon \cdot n}$. ■

13.7.3 The General Case: Constant-Round Doubly Efficient Proof Systems

In this section we prove Theorem 13.1.8. First, in Section 13.7.3, we show yet another refinement of the reconstructive PRG from Proposition 13.5.2, which will be used in the proof. Then in Section 13.7.3 we prove Theorem 13.1.8.

Yet another refinement of the reconstructive PRG

We now further refine the reconstructive PRG from Proposition 13.5.2. The goal now will be for the PRG to work not only with distinguishers, but also with distinguishers that solve a promise problem (*i.e.*, evaluate to Y on a pseudorandom input more than they evaluate to "not N" on a random input). The crux of the proof is to show that the advice depends only on the promise-problem, rather than on any particular oracle that solves the problem (and may behave arbitrarily on queries outside the promise).

Proposition 13.7.17 (an extension of the PRG from Proposition 13.5.2 to "promise problem" distinguishers)**.** *For any constant $\varepsilon' > 0$, we can replace the "furthermore" claim in Proposition 13.5.2, with the following claim. Fix a promise-problem (Y, N) such that*

$$\Pr[G^f(\mathbf{u}_{(1+\varepsilon_0)\cdot\log(N)}) \in \mathsf{Y}] > \Pr[\mathbf{u}_N \notin \mathsf{N}] + \varepsilon' .$$

Then, there exists $s \in \mathbb{N}$ satisfying $s|\bar{t}$, and $\alpha \in (0,1)$, and an advice string $\mathtt{adv}$ of length $|f|^{1-\delta_0}$ such that the following holds. Denoting by $a_{x,w,\gamma}$ the sequence of answers to R's queries on input $x \in [|f|]$ and witness w and random choices γ, we have that:

1. **(Completeness.)** *For any x there exists w such that with probability $1 - 1/N$ over γ, any function that agrees with (Y, N) yields a sequence of answers to the oracle queries that is (s, α)-valid, and if $a_{x,w,\gamma}$ is (s, α)-indicative of a sequence that agrees with (Y, N), then $R(x, w)$ outputs f_x.*

2. **(Soundness.)** *For every (x,w), with probability at least $1-1/N$ over γ, if $a_{x,w,\gamma}$ is (s,α)-indicative of a sequence that agrees with (Y,N) on the oracle queries, then $R(x,w)$ outputs either $\perp$ or f_x.*
3. **(Deficient oracles.)** *If $a_{x,w,\gamma}$ is (s,α)-deficient then R outputs $\perp$.*

Proof. We instantiate $\mathsf{Samp}\colon \{0,1\}^{\bar{N}} \times [\bar{L}] \to \{0,1\}^N$ just as in the proof of Proposition 13.4.1, and recall that its accuracy δ is sub-constant. We instantiate Enc with an agreement parameter $\rho = \rho(\varepsilon')$ that is now a sufficiently small constant that depends on $\varepsilon' > 0$. Other than that, we use the exact same generator G, and denote again the uniform distribution over the output-set of G^f by $\mathbf{G}$.

In the current proof, instead of assuming that we have oracle access to a function $D\colon \{0,1\}^N \to \{0,1\}$ that is a $(1/10)$-distinguisher for $\mathbf{G}$, we are first fixing a promise-problem (Y,N) such that

$$\Pr[\mathbf{G} \in \mathsf{Y}] > \Pr[\mathbf{u}_N \notin \mathsf{N}] + \varepsilon' \,, \tag{13.7.2}$$

and are only assuming that we have access to *some* (arbitrary) $D\colon \{0,1\}^N \to \{0,1\}$ that solves (Y,N). Our goal is for the advice to depend only on (Y,N), but for the reconstruction to work with *any* oracle D that agrees with (Y,N).

The error-reduced "promise distinguisher". We define the following two sets:

$$S = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j\in[\bar{L}]}[\mathsf{Samp}(z,j) \notin \mathsf{N}] \leq \Pr[\mathbf{u}_N \notin \mathsf{N}] + \varepsilon'/100 \right\} \,,$$

and

$$T = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j\in[\bar{L}]}[\mathsf{Samp}(z,j) \in \mathsf{Y}] > \Pr[\mathbf{u}_N \notin \mathsf{N}] + \varepsilon'/10 \right\} \,.$$

Observe that $T \subseteq \bar{S}$; this is the case because for any z we have that $\Pr_j[\mathsf{Samp}(z,j) \in \mathsf{Y}] \leq \Pr_j[\mathsf{Samp}(z,j) \notin \mathsf{N}]$, and thus if $z \in T$ then $z \notin S$. Also note that $|\bar{S}| \leq 2^{\bar{N}^{1-\gamma}}$, by the properties of Samp (we crucially use the fact that S is defined with respect to the specific event of being not in

N). Finally, similarly to Eq. (13.4.2), we have that

$$\begin{aligned}\Pr[\mathbf{G} \in \mathsf{Y}] &= \Pr_{i,j}\left[\mathsf{Samp}(\bar{f}_i, j) \in \mathsf{Y}\right] \\ &\leq \Pr_i[\bar{f}_i \in T] + \Pr_{i,j}\left[\mathsf{Samp}(\bar{f}_i, j) \in \mathsf{Y} | \bar{f}_i \notin T\right] \\ &\leq \Pr_i[\bar{f}_i \in T] + \left(\Pr[\mathbf{u}_N \notin \mathsf{N}] + \varepsilon'/10\right) \ ,\end{aligned}$$

and using Eq. (13.7.2) we deduce that $\Pr_i[\bar{f}_i \in T] \geq \rho$, relying on the assumption that $\rho = \rho(\varepsilon')$ is sufficiently small.

Computing a corrupted codeword. Analogously to the proofs of Proposition 13.4.1 and Proposition 13.5.2, we construct a machine M that, given any oracle D that agrees with (Y, N), computes a "corrupted" version of $\bar{f}$. We first fix a hash function $h \in \mathcal{H}$ such that there are no collision in $\bar{S}$, as in the previous proofs. The machine M gets as advice h, the value $\Pr[\mathbf{u}_N \notin \mathsf{N}]$, the set $I = \left\{(i, h(i)) :\in [L] \wedge \bar{f}_i \in T\right\}$, and the value $\alpha = \Pr[\mathbf{u}_N \notin \mathsf{N}] + .005$. We stress the following fact:

Observation 13.7.18. *The advice to M depends only on (Y, N), on Samp, and on h.*

Now, given $q \in [|\bar{f}|]$, the machine M first guesses a preimage $z \in \{0,1\}^{\bar{N}}$ for $\bar{f}_i$ and verifies that $h(z) = \bar{f}_i$ using the stored hash value. Then, M uniformly samples $s = O(\log(N))$ values $j_1, \ldots, j_s \in [\bar{L}]$, calls its oracle D on these values, and proceeds if and only if $\nu \stackrel{\text{def}}{=\!=} \Pr_{k \in [t]}[D(\mathsf{Samp}(z, j_k)) = 1] \geq \Pr[\mathbf{u}_N \notin \mathsf{N}] + \varepsilon'/50$. If both verifications succeeded, then M outputs the bit in z corresponding to index q, and otherwise M outputs $\perp$. The reconstruction R then uses the list-decoder (with fixed index and randomness) just as in the proof of Proposition 13.5.2.

The claim about deficient oracles follows immediately by the definition of M above (recall that R outputs $\perp$ whenever M returns $\perp$ in at least one execution). To demonstrate completeness, observe that for any x there exists w such that, with probability at least $1 - 1/N$, on every query q to M, any sequence of answers that is consistent with (Y, N) will have at least an α-fraction of answers that are not in N (since the corresponding z is in T); and any answers with an α-fraction of 1's cause M to correctly compute the corrupted codeword (since the corresponding z is the right preimage for the query under h). When this happens, Dec (and R) output f_x.

For the soundness case, fix (x, w) and again recall that the non-determinism w for R yields nondeterministic strings for each of the execution of M. For every execution of M on query q and

with non-determinism z, if $z \in S$ then with probability at least $1 - 1/N^2$ it holds that

$$\Pr_{k\in[t]}[\mathsf{Samp}(z, j_k) \notin \mathsf{N}] < \Pr[\mathbf{u}_N \notin \mathsf{N}] + \varepsilon'/50 \,,$$

in which case there does not exist a sequence of answers with at least an α-fraction of 1's that agrees with (Y, N). In this case the soundness claim holds vacuously (because there does not exist (s, α)-valid sequence of answers that agrees with (Y, N)).

Therefore, by a union-bound over the queries, we may assume that $z \in \bar{S}$ for all queries to M and that M outputs either $\perp$ (if $z \in \bar{S}$ is not the unique preimage under h for the corresponding query) or the corresponding bit in the corrupted codeword (if $z \in \bar{S}$ is indeed the unique preimage under h for the corresponding query). In case one of the queries of Dec is answered by $\perp$, then R outputs $\perp$ by definition; and otherwise, the execution of Dec is identical to an execution with access to the corrupted codeword, in with case $R(x, w) = f_x$. ■

The proof of Theorem 13.1.8

We now show that under strong hardness assumptions we can reduce the number of random coins in an arbitrary doubly efficient proof system to be logarithmic, without increasing the number of rounds. The hardness assumptions will refer to a function whose truth-tables can be recognized in near-linear time, but that is hard for multi-round AM protocols running in time $2^{(1-\delta)\cdot n}$ with $2^{(1-\delta)\cdot n}$ bits of non-uniform advice, for a small $\delta > 0$.

Theorem 13.7.19 (drastically reducing the number of random coins in a constant-round proof system)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. Let $c \in \mathbb{N}$ be a constant, and assume that there exists $L^{\mathsf{hard}} \notin$ i.o.-$\mathsf{MATIME}^{[\rightleftharpoons c+1]}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ such that given $n \in \mathbb{N}$, the truth-table of L^{hard} of n-bit inputs can be printed in time $2^{(1+\varepsilon/3)\cdot n}$. Then, for every polynomial $T(n)$ it holds that*

$$\mathsf{delP}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{delP}^{[\rightleftharpoons c]}[T^{1+\varepsilon}, (1+\varepsilon)\cdot\log(T)] \,.$$

Proof. Let $L \in \mathsf{delP}^{[\rightleftharpoons c]}[T]$, and let V be the corresponding T-time verifier for L. Denote by $c' = \lceil c/2 \rceil$ the number of turns of the verifier in the interaction. As in the proof of Theorem 13.7.5, we denote by $\langle V, P, x\rangle\,(r_1, \ldots, r_{c'})$ the result of an interaction between V and P on common input x where the coins $r_1, \ldots, r_{c'}$ are gradually revealed in the c' turns of V.

The new verifier. We define a verifier V' that gets input $x \in \{0,1\}^n$, and acts as follows. For $N = T(n)$, let f_n be the truth-table of L^{hard} on inputs of length $(1+\varepsilon/3)\cdot\log(N)$. Consider the generator G from Proposition 13.7.17 with $\varepsilon_0 = \varepsilon$ and input 1^N and oracle access to f_n, and denote the number of its outputs by $\bar{N} = N^{1+\varepsilon}$.

The verifier V' computes f_n, and in each turn i, it chooses a random $s_i \in [\bar{N}]$ and sends $G^{f_n}(s_i)$ it to the prover, instead of a random N-bit string. In the end V' applies the predicate V to the transcript. Note that V' uses $(1+\varepsilon)\cdot\log(N)$ random coins in each turn, and runs in time $O(N^{1+\varepsilon})$.

Completeness and soundness. The honest prover for V' behaves identically to the prover for V. Since the new protocol simulates the original protocol with a pseudorandom subset of the random strings, completeness follows immediately. We thus focus on proving the soundness of V'. Fix $x \notin L$.

Notation. Let us introduce some useful notation. For every $i \in [c']$ let V_i be the verifier that in the first i turns uses pseudorandom coins as above, and in the rest of the interaction uses random coins. By definition, we have that

$$\langle V_i, P, x\rangle\,(s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}) = \langle V, P, x\rangle\left(G^{f_n}(s_1),\ldots,G^{f_n}(s_i),r_{i+1},\ldots,r_{c'}\right)\ .$$

We define a sequence of hybrids, as follows:

$$p_{x,i} = \max_P\left\{\Pr_{s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}}\left[\langle V_i,P,x\rangle\,(s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'})=1\right]\right\}\ , \tag{13.7.3}$$

where $i = 0,\ldots,c'$. Indeed $p_{x,0}$ is just the maximal acceptance probability of V on input x whereas $p_{x,c'}$ is the maximal acceptance probability of V' on input x (where in both cases the maximum is taken over all provers).

Now, let $\bar{P}$ be the prover that maximizes $p_{x,c'}$. For any $i \in [c']$ and prover P, we think P as a concatenation $P^{1\ldots i}\circ P^{i+1\ldots c'}$,[50] and for any $i \in \{0,\ldots,c'\}$ we denote

$$(p_{x,i}|\bar{P}) = \max_{P^{i+1\ldots c'}}\left\{\Pr_{s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}}\left[\left\langle V_i,\bar{P}^{1\ldots i}\circ P^{i+1\ldots c'},x\right\rangle(s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'})=1\right]\right\}\ .$$

A hybrid argument. Assume that $p_{x,c'} = (p_{x,c'}|\bar{P}) \geq 1/2$. Since $x \notin L$, we have that $(p_{x,0}|\bar{P}) \leq 1/3$.

[50] Recall that, as in the proof of Theorem 13.7.5, we think of P as a sequence of c' functions, mapping transcripts to N-bit responses.

It follows that

$$1/6 < (p_{x,c'}|\bar{P}) - (p_{x,0}|\bar{P}) = \sum_{i\in[c']} (p_{x,i}|\bar{P}) - (p_{x,i-1}|\bar{P}) \,,$$

and hence for some $i \in [c']$ it holds that $(p_{x,i}|\bar{P}) - (p_{x,i-1}|\bar{P}) > 1/20c'$. Observe that

$$\begin{aligned}
&(p_{x,i}|\bar{P}) - (p_{x,i-1}|\bar{P}) \\
&= \max_{P^{i+1\ldots c'}} \left\{ \Pr_{s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}} \left[\left\langle V_i, \bar{P}^{1\ldots i} \circ P^{i+1\ldots c'}, x \right\rangle (s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}) = 1 \right] \right\} \\
&\quad - \max_{P^{i\ldots c'}} \left\{ \Pr_{s_1,\ldots,s_{i-1},r_i,\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\ldots i-1} \circ P^{i\ldots c'}, x \right\rangle (s_1,\ldots,s_{i-1},r_i,\ldots,r_{c'}) = 1 \right] \right\} \\
&\geq \max_{P^{i\ldots c'}} \left\{ \Pr_{s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}} \left[\left\langle V_i, \bar{P}^{1\ldots i-1} \circ P^{i\ldots c'}, x \right\rangle (s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}) = 1 \right] \right\} \\
&\quad - \max_{P^{i\ldots c'}} \left\{ \Pr_{s_1,\ldots,s_{i-1},r_i,\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\ldots i-1} \circ P^{i\ldots c'}, x \right\rangle (s_1,\ldots,s_{i-1},r_i,\ldots,r_{c'}) = 1 \right] \right\} \\
&= \mathop{\mathbb{E}}_{s_1,\ldots s_i} \left[\max_{P^{i\ldots c'}} \left\{ \Pr_{r_{i+1},\ldots,r_{c'}} \left[\left\langle V_i, \bar{P}^{1\ldots i-1} \circ P^{i\ldots c'}, x \right\rangle (s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}) = 1 \right] \right\} \right] \\
&\quad - \mathop{\mathbb{E}}_{s_1,\ldots,s_{i-1},r_i} \left[\max_{P^{i\ldots c'}} \left\{ \Pr_{r_{i+1},\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\ldots i-1} \circ P^{i\ldots c'}, x \right\rangle (s_1,\ldots,s_{i-1},r_i,\ldots,r_{c'}) = 1 \right] \right\} \right] \,, \qquad (13.7.4)
\end{aligned}$$

where the last equality is justified using the precise same argument as in Claim 13.7.9.

Defining a distinguisher. For any $\bar{s} = (s_1,\ldots,s_{i-1})$ and $r_i \in \{0,1\}^N$, denote

$$p_{\bar{s}}(r_i) = \max_{P^{i\ldots c'}} \left\{ \Pr_{r_{i+1},\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\ldots i-1} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, r_i,\ldots,r_{c'}) = 1 \right] \right\} \,,$$

and observe that Eq. (13.7.4) can thus be rewritten as

$$\mathop{\mathbb{E}}_{\bar{s}} \left[\mathop{\mathbb{E}}_{s_i\in[\bar{N}]} \left[p_{\bar{s}}(G^{f_n}(s_i)) \right] - \mathop{\mathbb{E}}_{r_i\in\{0,1\}^N} \left[p_{\bar{s}}(r_i) \right] \right] > 1/20c' \,.$$

We fix $\bar{s} = (s_1,\ldots,s_{i-1})$ such that the expected difference above is attained. Now, using Lemma 13.7.12 with $\mathbf{x} = p_{\bar{s}}(G^{f_n}(\mathbf{u}_{[\bar{N}]}))$ and $\mathbf{y} = p(\mathbf{u}_N)$ and $\varepsilon = 1/20c'$, there exists $\tau \in \{0, \varepsilon/4, 2\varepsilon/4, \ldots, 1\}$ such that

$$\Pr\left[p_{\bar{s}}(G^{f_n}(\mathbf{u}_{[\bar{N}]})) \geq \tau + \varepsilon/4 \right] - \Pr\left[p(\mathbf{u}_N) \geq \tau \right] > \varepsilon/2 \,. \qquad (13.7.5)$$

Now, consider the following promise problem:

$$\mathsf{Y} = \left\{(x, r_i, \pi_{\bar{s},\bar{p}}, \tau) : p_{\bar{s}}(r_i) \geq \tau + \varepsilon/4\right\}$$
$$\mathsf{N} = \left\{(x, r_i, \pi_{\bar{s},\bar{p}}, \tau) : p_{\bar{s}}(r_i) \leq \tau\right\} .$$

Note that by Eq. (13.7.5), we have that

$$\varepsilon/2 > \Pr[G^{f_n}(\mathbf{u}_{[\bar{N}]}) \in \mathsf{Y}] - \Pr[\mathbf{u}_N \notin \mathsf{N}] .$$

We further argue that (Y, N) is in $\mathsf{prAMTIME}^{[\rightleftharpoons c]}[O(n)]$, via the following protocol U. (Note that the input length to this problem is N, and thus the running time $O(N)$ that we will prove is linear in its input length.) The protocol U simulates the protocol underlying $p_{\bar{s}}$ (*i.e.*, interacts with its prover using random coins and applies the final predicate that V applies to the interaction, when the prefix is π) for constantly many times in parallel, to estimate its acceptance probability, with high probability, up to an error of $\varepsilon/8$. The procedure accepts if and only if its estimate is more than $\tau + \varepsilon/8$. Note that U is indeed an $\mathsf{AMTIME}^{[\rightleftharpoons c]}$ protocol that runs in linear time, uses a linear number of advice bits, and solves (Y, N).

A reconstruction argument. Let D be a protocol for the $\mathsf{prAMTIME}^{[\rightleftharpoons c]}[O(n)]$ above. Consider the reconstruction algorithm R from Proposition 13.7.17, with the advice string $\mathtt{adv}$ and $s \in \mathbb{N}$ and $\alpha > 0$ that correspond to (Y, N) above. Our protocol for the hard function f will simulate R, and resolve its queries by simulating D with our prover.

By the completeness of R, for every z there exists w for which with high probability over R's random coins, any sequence of answers to oracle queries that agrees with (Y, N) is (s, α)-valid, and any sequence of answers that is (s, α)-indicative of a sequence that agrees with (Y, N) causes R to output f_z. Thus, the prover can simply convince the verifier of the veracity of all queries that are Y instances: In this case, regardless of what the protocol answers for N instances, the sequence of answers agrees with (Y, N) and is thus (s, α)-indicative of itself, causing R to output f_z.

To establish the soundness of the protocol, denote the sequence of answers to the queries of R by $d_1, \ldots, d_{\bar{t}}$, and observe that with high probability, for every query q_i we have $d_i = 1 \Rightarrow q_i \notin \mathsf{N}$. Then, by Proposition 13.7.17, with high probability the following holds: If the sequence of answers to the verifier's queries is (α, s)-deficient, then R outputs $\perp$; and otherwise, it means that the answers are (s, α)-indicative of the sequence of answers that agree with (Y, N), in which case we

are in the soundness case and the output is either f_z or $\perp$.

The procedure above is an $\mathsf{MATIME}^{[\rightleftharpoons c+1]}$ protocol (since we are using the first round to receive w, then c rounds to simulate D) and it runs in time

$$\underbrace{|f_n|^{1-\delta_0}}_{\text{complexity of } R} + \underbrace{N^{\varepsilon/10}}_{\text{number of queries}} \cdot \underbrace{O(N)}_{\text{length of each query}} < |f_n|^{1-\delta},$$

relying on a sufficiently small choice of $\delta > 0$. Similarly, the total number of advice bits that it uses is $|f_n|^{1-\delta_0} + O(N) < |f_n|^{1-\delta}$. This contradicts the hardness of L^{hard}. ■

By combining Theorem 13.7.19 and Theorem 13.7.5, we now show that under strong hardness assumptions we can simulate any $\mathsf{AMTIME}^{[\rightleftharpoons c]}$ protocol by a deterministic doubly efficient argument system, with essentially no time overhead.

Corollary 13.7.20 (simulating proof systems by deterministic doubly efficient argument systems; Theorem 13.1.8, restated)**.** *For every $\alpha, \beta, \varepsilon \in (0,1)$ there exists $\eta, \delta > 0$ such that for every $c \in \mathbb{N}$ the following holds. Assume that:*

1. *There exists $L^{\mathsf{hard}} \notin$ i.o.-$\mathsf{MATIME}^{[\rightleftharpoons c+1]}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ such that given $n \in \mathbb{N}$, the truth-table of L^{hard} of n-bit inputs can be printed in time $2^{(1+\varepsilon/3)\cdot n}$.*
2. *The $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time T' with oracle access to $\mathsf{prAMTIME}_2^{[\rightleftharpoons c]}[n]$ on inputs of length $\Theta(T^{1+\varepsilon/2} \cdot R)$, where $R = (1+\varepsilon) \cdot \log(T)$ and $N = n + (c \cdot \lceil c/w \rceil) \cdot T^{1+\varepsilon/2} \cdot R$ and $K = R^{1/\eta}$ and $T' = T^{1+\varepsilon/2} \cdot K \cdot R$.*

Then, $\mathsf{delP}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{NARG}[T^{1+\varepsilon}]$.

Proof. Let $L \in \mathsf{delP}^{[\rightleftharpoons c]}[T]$. By the first hypothesis and Theorem 13.7.19 we have that $L \in \mathsf{delP}^{[\rightleftharpoons c]}[\bar{T}, R]$ for $\bar{T} = T^{1+\varepsilon/2}$ and $R = (1+\varepsilon/2) \cdot \log(T)$. And by the second hypothesis and Theorem 13.7.5 with time bound $\bar{T}$ and randomness R, we have that $L \in \mathsf{NARG}[\bar{T} \cdot \mathrm{poly}(R)] \subseteq \mathsf{NARG}[T^{1+\varepsilon}]$. ■

As in Remark 13.7.14, the derandomization in Corollary 13.7.20 extends to the setting in which the original honest prover (in the probabilistic proof system) was efficient only when given a witness in a relation, and in this case the honest prover in the argument system is also efficient only when given a witness in the same relation. (This is because, similarly to Remark 13.7.14, the honest prover in the argument system underlying Theorem 13.7.19 is identical to the original honest prover.)

13.8 Useful Properties Are Necessary for Derandomization of MA

In this appendix we prove that useful properties that are *constructive in near-linear time* are necessary for derandomization of MA. We first prove that *infinitely often* useful properties are necessary for *any* derandomization of MA, following the proof approach of Williams [Wil13a, Wil16b]; and then we prove that useful properties (in the standard sense, *i.e.* not only infinitely often) are necessary for black-box superfast derandomization of MA.

Definition 13.8.1 (io-useful property; compare with Definition 13.3.3)**.** *We say that a collection $\mathcal{L} \subseteq \{0,1\}^*$ of strings is a $\mathcal{C}'$-constructive property io-useful against $\mathcal{C}$ if for every $N = 2^n$ it holds that $\mathcal{L}_n = \mathcal{L} \cap \{0,1\}^N \neq \emptyset$, and $\mathcal{L} \in \mathcal{C}'$, and for every $L \in \mathcal{C}$ there are infinitely many $n \in \mathbb{N}$ such that $L_n \notin \mathcal{L}_n$, where $L_n \in \{0,1\}^{2^n}$ is the truth-table of L on n-bit inputs.*

Proposition 13.8.2 (io-useful properties are necessary for derandomization of MA)**.** *If every unary language in* MA *is also in* NP *then there is a quasilinear-time-constructive property io-useful against circuits of size $2^{\varepsilon \cdot n}$, for some constant $\varepsilon > 0$.*

Proof. We first use the well-known proof approach from [Wil13a, Wil16b] to argue that witnesses for a certain unary language in $\mathsf{NTIME}[2^n]$ have exponential circuit complexity, infinitely often; and then follow [Wil16b] in observing that the latter statement yields an io-useful property for exponential-sized circuits. Details follow.

Let L be a unary language in $\mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^{n/2}]$ (see [Žák83]), and let V be a PCP verifier for L with running time $\mathrm{poly}(n)$ and randomness $n + O(\log n)$ and perfect completeness and soundness error $1/3$ (see [BSGH+06]). Consider an MA verifier V^{PCP} that on n-bit inputs guesses a circuit C of size $2^{\varepsilon \cdot n}$, where $\varepsilon > 0$ is sufficiently small, and runs V while resolving its oracle queries using C.

Assume towards a contradiction that for every sufficiently large $n \in \mathbb{N}$, if $1^n \in L$ then there is $w \in \{0,1\}^{2^n}$ that is the truth-table of a function computable by circuits of size $2^{\varepsilon \cdot n}$ such that $\Pr[V(1^n, w) = 1] = 1$. In this case, V^{PCP} is an MA verifier that decides L in time $\tilde{O}(2^{\varepsilon \cdot n})$, and thus (by our hypothesis and relying on $\varepsilon > 0$ being small) it holds that $L \in \mathsf{NTIME}[2^{n/2}]$, a contradiction.

Thus, there is an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ there exists $w \in \{0,1\}^{2^n}$ such that $\Pr[V(1^n, w) = 1] = 1$, and every w satisfying this condition is the truth-table of a function whose circuit complexity is more than $2^{\varepsilon \cdot n}$. Our io-useful property consists of all such w's, for

every $n \in S$. The constructive algorithm for the property gets $f_n \in \{0,1\}^{2^n}$ and enumerates over the randomness of V to compute $\Pr[V(1^n, f_n) = 1]$; if the latter value is 1 then it accepts, otherwise it rejects. Indeed, the property is non-trivial infinitely often, its truth-tables are hard for circuits of size $2^{\varepsilon \cdot n}$, and it is constructive in time $\tilde{O}(2^n)$. ■

We now prove that useful properties (which are not only "infinitely often" useful) that are constructive in near-linear time are necessary for any superfast derandomization of MA that uses nondeterministic PRGs (NPRGs). The proof follows the standard transformation of PRGs into hard functions, adapting it to our setting: Replacing PRGs with NPRGs, and hard functions with useful properties.

Definition 13.8.3 (nondeterministic PRG)**.** *A nondeterministic machine M is a* nondeterministic ε-PRG *(ε-NPRG, in short) for a circuit class $\mathcal{C}$ if for every $n \in \mathbb{N}$, when M is given input 1^n it satisfies the following:*

1. *There exists a nondeterministic guess such that M prints $S \subseteq \{0,1\}^n$ for which there is no circuit on n inputs in $\mathcal{C}$ that is an ε-distinguisher.*[51]
2. *For every nondeterministic guess, either M prints $S \subseteq \{0,1\}^n$ for which there is no circuit on n inputs in $\mathcal{C}$ that is an ε-distinguisher, or M outputs $\perp$.*

Proposition 13.8.4 (useful properties are necessary for derandomization with NPRGs)**.** *For every $\varepsilon > 0$ there exists $\delta > 0$ such that the following holds. If there is a $(1/10)$-NPRG for linear-sized circuits that is computable in time $n^{1+\varepsilon}$, then there is an $\mathsf{NTIME}[N^{1+3\varepsilon}]$-computable property $\mathcal{L}$ useful against circuits of size $2^{(1-\delta)\cdot n}$.*

Proof. For every $n \in \mathbb{N}$, let $\mathcal{S}_n$ be the collection of all possible sets S that the NPRG can print on input length 1^n (*i.e.*, when considering all nondeterministic guesses that do not cause M to output $\perp$). Fix $S \in \mathcal{S}_n$, and note that $|S| \leq n^{1+\varepsilon}$ (due to the running time of M). For $\ell(n) = \lceil (1+2\varepsilon) \cdot \log(n) \rceil$, let $f_S \colon \{0,1\}^\ell \to \{0,1\}$ such that $f_S(x) = 1$ if and only if x is a prefix of some $z \in S$. Note that there is no circuit C of size $O(n) = 2^{\Omega(\ell/(1+2\varepsilon))} = 2^{(1-\delta)\cdot\ell}$ that computes f_S (otherwise we could use C to construct a circuit that avoids S but has high acceptance probability).

For every $\ell \in \mathbb{N}$, we include in $\mathcal{L}$ all the functions f_S obtained from $S \in \mathcal{S}_n$ such that $\ell = \ell(n)$. The argument above shows that $\mathcal{L}$ is useful against circuits of size $2^{(1-\delta)\cdot\ell}$, and its non-triviality follows since the NPRG works on all input lengths. Finally, the truth-tables of f_S's included in $\mathcal{L}$ can be recognized in nondeterministic time $\tilde{O}(N^{1+2\varepsilon}) < N^{1+3\varepsilon}$, by computing the NPRG. ■

[51] That is, for every $C \in \mathcal{C}$ on n input bits it holds that $\Pr_{s \in S}[C(s) = 1] \in \Pr_{x \in \{0,1\}^n}[C(x) = 1] \pm \varepsilon$.

Indeed, the proof above works also with the weaker notion of nondeterministic HSG (NHSG), and we formulated it using NPRGs merely since the latter notion is more well-known.

13.9 Proof of Lemma 13.7.12

We now restate Lemma 13.7.12 from the proof of Theorem 13.7.5 and prove it.

Lemma 13.9.1. *Let* $\mathbf{x}, \mathbf{y}$ *be two random variables taking values from* $[0,1]$ *and* $\varepsilon \in (0,1)$ *such that* $\varepsilon^{-1} \in \mathbb{N}$*, and let* $\tau = 4/\varepsilon$*. If* $\mathsf{E}[\mathbf{x}] - \mathsf{E}[\mathbf{y}] > \varepsilon$*, then there exists* $\ell \in [\tau]$ *such that*

$$\Pr[\mathbf{x} \geq (\ell+1)/\tau] - \Pr[\mathbf{y} \geq \ell/\tau] > \varepsilon/2 \,.$$

Proof. For every $i \in \{0, 1 \ldots, \tau+1\}$, let

$$p_i = \Pr\left[\mathbf{x} \in [i/\tau, (i+1)/\tau)\right] \quad \text{and} \quad q_i = \Pr\left[\mathbf{y} \in [i/\tau, (i+1)/\tau)\right] .$$

Note that $p_{\tau+1} = q_{\tau+1} = 0$ (we define them purely for notational convenience), and that $\sum_{i=0}^{\tau} p_i = \sum_{i=0}^{\tau} q_i = 1$. Also, we have that

$$\mathsf{E}[\mathbf{x}] \in \left[\sum_{i=0}^{\tau} p_i \cdot (i/\tau), \tau^{-1} + \sum_{i=0}^{\tau} p_i \cdot (i/\tau)\right) ,$$
$$\mathsf{E}[\mathbf{y}] \in \left[\sum_{i=0}^{\tau} q_i \cdot (i/\tau), \tau^{-1} + \sum_{i=0}^{\tau} q_i \cdot (i/\tau)\right) ,$$

which by our hypothesis $\mathsf{E}[\mathbf{x}] - \mathsf{E}[\mathbf{y}] > \varepsilon$ implies that

$$\left[\sum_{i=0}^{\tau} p_i \cdot (i/\tau)\right] - \left[\sum_{i=0}^{\tau} q_i \cdot (i/\tau)\right] > \varepsilon - \tau^{-1} .$$

Now, denote $p_{\geq \ell} = \sum_{j=\ell}^{\tau} p_j$ and $q_{\geq \ell} = \sum_{j=\ell}^{\tau} q_j$. Then, we have that

$$\sum_{i=0}^{\tau} p_i \cdot (i/\tau) = \sum_{\ell=1}^{\tau} p_{\geq \ell}/\tau = \mathop{\mathbb{E}}_{\ell \in [\tau]} [p_{\geq \ell}] ,$$
$$\sum_{i=0}^{\tau} q_i \cdot (i/\tau) = \mathop{\mathbb{E}}_{\ell \in [\tau]} [q_{\geq \ell}] ,$$

and therefore

$$\mathop{\mathbb{E}}_{\ell\in[\tau]}\left[p_{\geq\ell}-q_{\geq\ell}\right]>\varepsilon-\tau^{-1}\,.$$

Next, observe that

$$\mathop{\mathbb{E}}_{\ell\in[\tau]}\left[p_{\geq\ell+1}\right]=\mathop{\mathbb{E}}_{\ell\in[\tau]}\left[p_{\geq\ell}\right]-\mathop{\mathbb{E}}_{\ell\in[\tau]}\left[p_{\ell}\right]\geq\mathop{\mathbb{E}}_{\ell\in[\tau]}\left[p_{\geq\ell}\right]-1/\tau\,.$$

Putting them together, we have

$$\mathop{\mathbb{E}}_{\ell\in[\tau]}\left[p_{\geq\ell+1}-q_{\geq\ell}\right]>\varepsilon-2\tau^{-1}=\varepsilon/2\,,$$

and hence there exists $\ell\in[\tau]$ such that $p_{\geq\ell+1}-q_{\geq\ell}>\varepsilon/2$. ■

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[ABO84] Miklos Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 471–474, 1984.

[AC19] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2019.

[ACR98] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. A new general derandomization method. *Journal of the ACM*, 45(1):179–213, 1998.

[ACW16] Josh Alman, Timothy M. Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 467–476, 2016.

[ADH97] Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM J. Comput.*, 26(5):1524–1540, 1997.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[AIKS16] Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. Pseudorandomness when the odds are against you. In *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*, pages 9:1–9:35, 2016.

[Ajt83] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

[Ajt90] Miklós Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop, New Jersey, USA, December 3-7, 1990*, pages 1–20, 1990.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics. Second Series*, 160(2):781–793, 2004.

[AKS19] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Errata: PRIMES is in P [MR2123939]. *Annals of Mathematics. Second Series*, 189(1):317–318, 2019.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[Alm19] Josh Alman. An illuminating algorithm for the light bulb problem. In *Proc. 2nd Symposium on Simplicity in Algorithms (SOSA)*, pages 2:1–2:11, 2019.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[BCG20] Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM Journal of Computing*, 49(5), 2020.

[BF99] Harry Buhrman and Lance Fortnow. One-sided versus two-sided error in probabilistic computation. In *Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 100–109, 1999.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Comput. Complex.*, 1:3–40, 1991.

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

[BFS09] Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against advice. In *Proc. 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 195–209, 2009.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proc. 13th Annual IEEE Conference on Computational Complexity (CCC)*, pages 8–12, 1998.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proc. 20th Annual IEEE Conference on Computational Complexity (CCC)*, pages 120–134, 2005.

[BHK09] Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate bregman projections. In *Proc. 20th Annual IEEE Conference on Computational Complexity (CCC)*, pages 1193–1200, 2009.

[BHPT20] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular PCPs or: Hard claims have complex proofs. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 858–869, 2020.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012.

[Bjö14] Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014.

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

[BM88] László Babai and Shlomo Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[BSGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM Journal of Computing*, 36(4):889–974, 2006.

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *Proc. 7th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 200–215, 2003.

[BT94] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.

[BV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 163–173, 2014.

[BvzGH82] Allan Borodin, Joachim von zur Gathen, and John E. Hopcroft. Fast parallel matrix and GCD computations. *Inf. Control.*, 52(3):241–256, 1982.

[CGI+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proc. 7th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 261–270, 2016.

[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)*, pages 1281–1304, 2019.

[Che20] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory Comput.*, 16:1–50, 2020.

[CHLT19] Eshan Chattopadhyay, Pooya Hatami, Shachar Lovett, and Avishay Tal. Pseudorandom generators from the second Fourier level and applications to AC^0 with parity gates. In *Proc. 10th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 22:1–22:15, 2019.

[CHO+22] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. *Journal of the ACM*, 69(4):25:1–25:49, 2022.

[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)*, pages 27:1–27:16, 2018.

[CJSW21] Lijie Chen, Ce Jin, Rahul Santhanam, and Ryan Williams. Constructive separations and their consequences. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 646–657, 2021.

[CJW19] Lijie Chen, Ce Jin, and Richard Ryan Williams. Hardness magnification for all sparse NP languages. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1240–1255, 2019.

[CJW20] Lijie Chen, Ce Jin, and Richard Ryan Williams. Sharp threshold results for computational complexity. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1335–1348, 2020.

[CJWW22] Lijie Chen, Ce Jin, Ryan Williams, and Hongxun Wu. Truly low-space element distinctness and subset sum via pseudorandom hash functions. In *Proc. 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1661–1678, 2022.

[CKN18] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018.

[CKP+21] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 570–583, 2021.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. Optimal error pseudodistributions for read-once branching programs. In *Proc. 35th Annual IEEE Conference on Computational Complexity (CCC)*, pages 25:1–25:27, 2020.

[CL21] Lijie Chen and Xin Lyu. Inverse-exponential correlation bounds and extremely rigid matrices from a new derandomized XOR lemma. In *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 761–771, 2021.

[CLLO21] Lijie Chen, Zhenjian Lu, Xin Lyu, and Igor Carboni Oliveira. Majority vs. approximate linear sum and average-case complexity below NC^1. In *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 51:1–51:20, 2021.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[CLW20] Lijie Chen, Xin Lyu, and Richard Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–12, 2020.

[CLY22] Lijie Chen, Jiatu Li, and Tianqi Yang. Extremely efficient constructions of hash functions, with applications to hardness magnification and PRFs. In *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*, pages 23:1–23:37, 2022.

[CMMW19] 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)*, pages 30:1–30:21, 2019.

[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))*, pages 726–735, 1999.

[COS18] Ruiwen Chen, Igor Carboni Oliveira, and Rahul Santhanam. An average-case lower bound against ACC^0. In *LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Proceedings*, pages 317–330, 2018.

[CP16] Shiteng Chen and Periklis A. Papakonstantinou. Depth-reduction for composites. In *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 99–108, 2016.

[CR20] Lijie Chen and Hanlin Ren. Strong average-case lower bounds from non-trivial derandomization. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1327–1334, 2020.

[CR21] Lijie Chen and Hanlin Ren. Strong average-case circuit lower bounds from nontrivial derandomization. *SIAM Journal of Computing*, 51(3):STOC20–115, 2021.

[CRT22] Lijie Chen, Ron D. Rothblum, and Roei Tell. Unstructured hardness to average-case randomness. In *Proc. 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022.

[CRTY20] Lijie Chen, Ron D. Rothblum, Roei Tell, and Eylon Yogev. On exponential-time hypotheses, derandomization, and circuit lower bounds. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 13–23, 2020.

[Csa76] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5(4):618–623, 1976.

[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)*, pages 1:1–1:35, 2016.

[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)*, pages 34–41, 2019.

[CT21a] Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 125–136, 2021.

[CT21b] Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost. In *Proc. 53st Annual ACM Symposium on Theory of Computing (STOC)*, pages 283–291, 2021.

[CT22] Lijie Chen and Roei Tell. When Arthur has neither random coins nor time to spare: Superfast derandomization of proof systems. *Electronic Colloquium on Computational Complexity: ECCC*, 2022.

[CW16] Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1246–1255, 2016.

[CW19] Lijie Chen and R. Ryan Williams. Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity. In *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*, pages 19:1–19:43, 2019.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[DMOZ20] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1057–1068, 2020.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal of Computing*, 22(5):994–1005, 1993.

[FGHP99] Stephen A. Fenner, Frederic Green, Steven Homer, and Randall Pruim. Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy. *Electronic Colloquium on Computational Complexity (ECCC)*, 6(3), 1999.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 2–12, 1991.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5, 1989.

[FK18] Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 946–955, 2018.

[FKL+01] Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *Proc. 21st Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 171–182, 2001.

[FLvMV05] Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM*, 52(6):835–865, 2005.

[FS16] Lance Fortnow and Rahul Santhanam. New non-uniform lower bounds for uniform classes. In *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*, pages 19:1–19:14, 2016.

[FS17] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. *Inf. Comput.*, 2017.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[FSUV13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013.

[GGH+07] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 440–449, 2007.

[GHR92] Mikael Goldmann, Johan Håstad, and Alexander Razborov. Majority gates vs. general weighted threshold gates. In *Proc. 7th Annual Structure in Complexity Theory Conference*, pages 2–13, 1992.

[GJS76] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM*, 62(4):27:1–27:64, 2015.

[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)*, pages 25–32, 1989.

[GLR+91] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proc. 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 32–42, 1991.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 273–301. Springer, 2011.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, New York, NY, USA, 2008.

[Gol11a] Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*, pages 191–232, 2011.

[Gol11b] Oded Goldreich. Two comments on targeted canonical derandomizers. *Electronic Colloquium on Computational Complexity: ECCC*, 18:47, 2011.

[Gol18] Oded Goldreich. On doubly-efficient interactive proof systems. *Foundations and Trends® in Theoretical Computer Science*, 13(3):front matter, 1–89, 2018.

[GR17] Oded Goldreich and Guy N. Rothblum. Worst-case to average-case reductions for subclasses of P. *Electronic Colloquium on Computational Complexity: ECCC*, 26:130, 2017.

[GR18] Oded Goldreich and Guy N. Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. In *Proc. 9th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 94, pages 18:1–18:19, 2018.

[GSTS03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3-4):85–130, 2003.

[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)*, pages 469–482, 2008.

[GVW11] Oded Goldreich, Salil Vadhan, and Avi Wigderson. Simplified derandomization of BPP using a hitting set generator. In *Studies in complexity and cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 59–67. Springer, Heidelberg, 2011.

[GW02] Oded Goldreich and Avi Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *Proc. 6th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 209–223, 2002.

[GW14] Oded Goldreich and Avi Widgerson. On derandomizing algorithms that err extremely rarely. In *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 109–118, 2014.

[HAB02] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.

[Hås89] Johan Håstad. Almost optimal lower bounds for small depth circuits. *Advances in Computing Research*, 5:143–170, 1989.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999.

[HLV18] Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM Journal of Computing*, 47(2):493–523, 2018.

[HMP+93] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993.

[Hof96] Thomas Hofmeister. A note on the simulation of exponential threshold weights. In *Computing and Combinatorics, Second Annual International Conference, COCOON '96, Hong Kong, June 17-19, 1996, Proceedings*, pages 136–141, 1996.

[Hol05] Thomas Holenstein. Key agreement from weak bit agreement. In *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 664–673, 2005.

[Hoz19] William M. Hoza. Typically-correct derandomization for small time and space. In *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*, pages 9:1–9:39, 2019.

[HP10] Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Exact threshold circuits. In *Proc. 25th Annual IEEE Conference on Computational Complexity (CCC)*, pages 270–279, 2010.

[HR03] Tzvika Hartman and Ran Raz. On the distribution of the number of roots of polynomials and explicit weak designs. *Random Structures & Algorithms*, 23(3):235–263, 2003.

[HRST17] Johan Håstad, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *Journal of the ACM*, 64(5):35:1–35:27, 2017.

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Proc. 23rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 672–683, 2006.

[HV21] Xuangui Huang and Emanuele Viola. Average-case rigidity lower bounds. In *Proc. 16th International Computer Science Symposium in Russia (CSR)*, pages 186–205. Springer, 2021.

[HVV06] Alexander Healy, Salil P. Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal of Computing*, 35(4):903–931, 2006.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: simplified, optimized, and derandomized. *SIAM Journal of Computing*, 39(4):1637–1665, 2010.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 244–256, 2002.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

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

[IW97] 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)*, pages 220–229, 1997.

[IW98] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proc. 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 734–743, 1998.

[JMV15] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Proc. 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 749–760, 2015.

[Kab01] Valentine Kabanets. Easiness assumptions and hardness tests: trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KL18] Valentine Kabanets and Zhenjian Lu. Satisfiability and derandomization for small polynomial threshold circuits. In *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 46:1–46:19, 2018.

[Kli01] Adam R. Klivans. On the derandomization of constant depth circuits. In *Proc. 4th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 249–260, 2001.

[Koz78] Dexter Kozen. Indexing of subrecursive classes. In *Proc. 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 287–295, 1978.

[Koz92] Dexter Campbell Kozen. *Design and Analysis of Algorithms*. Texts and Monographs in Computer Science. Springer, 1992.

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[KvMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Computational Complexity*, 21(1):3–61, 2012.

[Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

[Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4):859–868, 1992.

[Lok09] Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009.

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 303–316, 2021.

[LP22] Yanyi Liu and Rafael Pass. On one-way functions from np-complete problems. In *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*, pages 36:1–36:24, 2022.

[LPT+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2190–2202, 2017.

[Lu01] Chi-Jen Lu. Derandomizing Arthur-Merlin games under uniform assumptions. *Computational Complexity*, 10(3):247–259, 2001.

[LV08] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Texts in Computer Science. Springer, New York, third edition, 2008.

[LV20] Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. *Theory of Computing*, 16:1–50, 2020.

[MNT93] Yishay Mansour, Noam Nisan, and Prasoon Tiwari. The computational complexity of universal hashing. *Theoretical Computer Science*, 107(1):121–133, 1993.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[MW18] Cody Murray and R. 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)*, pages 890–901, 2018.

[MW20] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM Journal of Computing*, 49(5), 2020.

[NIR03] Alan Nash, Russell Impagliazzo, and Jeff Remmel. Universal languages and the power of diagonalization. In *Proc. 18th Annual IEEE Conference on Computational Complexity (CCC)*, page 337, 2003.

[NIR06] Alan Nash, Russell Impagliazzo, and Jeff Remmel. Infinitely-often universal languages and diagonalization. *Electronic Colloquium on Computational Complexity: ECCC*, 13:51, 2006.

[Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

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

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[OL21] Igor Carboni Oliveira and Zhenjian Lu. An efficient coding theorem via probabilistic representations and its applications. In *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 94:1–94:20, 2021.

[Oli19] Igor Carboni Oliveira. Randomness and intractability in kolmogorov complexity. In *Proc. 46th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 32:1–32:14, 2019.

[PPSZ05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005.

[Raz87] Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematical Notes of the Academy of Science of the USSR*, 41(4):333–338, 1987.

[Raz89] Alexander A. Razborov. On rigid matrices (in Russian), 1989. Steklov Mathematical Institute. Paper at http://people.cs.uchicago.edu/~razborov/files/rigid.pdf.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1, part 1):24–35, 1997.

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Efficient Batch Verification for UP. In *Proc. 33rd Annual IEEE Conference on Computational Complexity (CCC)*, pages 22:1–22:23, 2018.

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM Journal of Computing*, 50(3):STOC16–255–STOC16–340, 2021.

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *Journal of Computer and System Sciences*, 65(1):97–128, 2002.

[RSS18] Ninad Rajgopal, Rahul Santhanam, and Srikanth Srinivasan. Deterministically counting satisfying assignments for constant-depth circuits with parity gates, with implications for lower bounds. In *Proc. 43rd International Symposium on Mathematical Foundations of Computer Science*, pages 78:1–78:15, 2018.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *Proc. 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022.

[RTTV08] Omer Reingold, Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Dense subsets of pseudorandom sets. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 76–85, 2008.

[San09] Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. *SIAM Journal of Computing*, 39(3):1038–1061, 2009.

[Sap17] Ramprasad Saptharishi. Algebra and computation, 2017.

[SFM78] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[Sha03] Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1-2):1–22, 2003.

[Sha10] Ronen Shaltiel. Typically-correct derandomization. *SIGACT News*, 41(2):5772, June 2010.

[Sha11] Ronen Shaltiel. Weak derandomization of weak algorithms: explicit versions of Yao's lemma. *Computational Complexity*, 20(1):87–143, 2011.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proc. 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 330–335, 1983.

[Sip88] Michael Sipser. Expanders, randomness, or time versus space. *Journal of Computer and System Sciences*, 36(3):379–383, 1988.

[SM73] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time. In *Proc. 5th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–9, 1973.

[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)*, pages 77–82, 1987.

[Smo93] Roman Smolensky. On representations by low-degree polynomials. In *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 130–138, 1993.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.

[SU06] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Comput. Complex.*, 15(4):298–341, 2006.

[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)*, pages 430–439, 2007.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complex.*, 13(1):180–193, 1997.

[Sud15] Madhu Sudan. Algebra and computation, lecture 9, 2015.

[Sud21] Madhu Sudan. Personal communication, 2021.

[SV10] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM Journal of Computing*, 39(7):3122–3154, 2010.

[SW13] Rahul Santhanam and Ryan Williams. On medium-uniformity and circuit lower bounds. In *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*, pages 15–23, 2013.

[Tam16] Suguru Tamaki. A satisfiability algorithm for depth two circuits with a sub-quadratic number of symmetric and threshold gates. *Electronic Colloquium on Computational Complexity: ECCC*, 23:100, 2016.

[Tel17] Roei Tell. Improved bounds for quantified derandomization of constant-depth circuits and polynomials. In *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*, pages 18:1 – 18:49, 2017.

[Tel18] Roei Tell. Quantified derandomization of linear threshold circuits. In *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 855–865, 2018.

[Tel19] Roei Tell. Proving that $prBPP = prP$ is as hard as proving that "almost NP" is not contained in P/poly. *Information Processing Letters*, 152:105841, 2019.

[Tel21] Roei Tell. How to find water in the ocean: A survey on quantified derandomization. *Electronic Colloquium on Computational Complexity: ECCC*, 28:120, 2021.

[Tha22] Justin Thaler. Proofs, arguments, and zero-knowledge, 2022. Accessed at https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html, March 28, 2022.

[Tou01] Iannis Tourlakis. Time-space tradeoffs for SAT on nonuniform machines. *Journal of Computer and System Sciences*, 63(2):268–287, 2001.

[TSZS06] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. Extractors from Reed-Muller codes. *Journal of Computer and System Sciences*, 72(5):786–812, 2006.

[TTV09] Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *Proc. 24th Annual IEEE Conference on Computational Complexity (CCC)*, pages 126–136, 2009.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[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 *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176, 1977.

[Vio09a] Emanuele Viola. Guest column: correlation bounds for polynomials over $\{0,1\}$. *SIGACT News*, 40(1):27–44, 2009.

[Vio09b] Emanuele Viola. On approximate majority and probabilistic time. *Computational Complexity*, 18(3):337–375, 2009.

[Vio09c] Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009.

[Vio20] Emanuele Viola. New lower bounds for probabilistic degree and AC0 with parity gates. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:15, 2020.

[VL99] Jacobus Hendricus Van Lint. *Introduction to coding theory*, volume 86. Springer-Verlag Berlin Heidelberg, 1999.

[vMS05] Dieter van Melkebeek and Rahul Santhanam. Holographic proofs and derandomization. *SIAM Journal of Computing*, 35(1):59–90, 2005.

[VW20] Nikhil Vyas and R. Ryan Williams. Lower bounds against sparse symmetric functions of ACC circuits: Expanding the reach of #SAT algorithms. In *Proc. 37th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 59:1–59:17, 2020.

[WB86] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 231–240, 2010.

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*, pages 115–125, 2011.

[Wil13a] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal of Computing*, 42(3):1218–1244, 2013.

[Wil13b] Ryan Williams. Natural proofs versus derandomization. In *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2013.

[Wil14a] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 194–202, 2014.

[Wil14b] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM*, 61(1):2:1–2:32, 2014.

[Wil16a] R. Ryan Williams. Natural proofs versus derandomization. *SIAM Journal of Computing*, 45(2):497–529, 2016.

[Wil16b] 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)*, volume 50, pages 2:1–2:17, 2016.

[Wil18] Richard Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In *Proc. 33rd Annual IEEE Conference on Computational Complexity (CCC)*, pages 6:1–6:24, 2018.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

[Yao85] Andrew C-C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1985.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *Proc. 27th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

[Yao90] Andrew Chi-Chih Yao. On ACC and threshold circuits. In *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 619–627, 1990.

[Žák83] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983.

[Zim08] Marius Zimand. Exposure-resilient extractors and the derandomization of probabilistic sublinear time. *Computational Complexity*, 17(2):220–253, 2008.