Algorithms and Lower Bounds for De Morgan Formulas of Low-Communication Leaf Gates

Valentine Kabanets
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
https://www.cs.sfu.ca/~kabanets/
kabanets@cs.sfu.ca

Sajin Koroth
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
http://www.sfu.ca/~skoroth/
sajin_koroth@sfu.ca

Zhenjian Lu
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
Department of Computer Science, University of Warwick, UK
zhenjian_lu@sfu.ca

Dimitrios Myrisiotis
Department of Computing, Imperial College London, UK
d.myrisiotis17@ic.ac.uk

Igor C. Oliveira
Department of Computer Science, University of Warwick, UK
https://www.dcs.warwick.ac.uk/~igorcarb/
igor.oliveira@warwick.ac.uk

Abstract

The class $\mathsf{FORMULA}[s] \circ \mathcal{G}$ consists of Boolean functions computable by size-s de Morgan formulas whose leaves are any Boolean functions from a class $\mathcal{G}$. We give *lower bounds* and (SAT, Learning, and PRG) *algorithms* for $\mathsf{FORMULA}[n^{1.99}] \circ \mathcal{G}$, for classes $\mathcal{G}$ of functions with *low communication complexity*. Let $R^{(k)}(\mathcal{G})$ be the maximum k-party number-on-forehead randomized communication complexity of a function in $\mathcal{G}$. Among other results, we show that:

- The Generalized Inner Product function GIP_n^k cannot be computed in $\mathsf{FORMULA}[s] \circ \mathcal{G}$ on more than $1/2 + \varepsilon$ fraction of inputs for

$$s = o\left(\frac{n^2}{\left(k \cdot 4^k \cdot R^{(k)}(\mathcal{G}) \cdot \log(n/\varepsilon) \cdot \log(1/\varepsilon)\right)^2}\right).$$

  This significantly extends the lower bounds against bipartite formulas obtained by [61]. As a corollary, we get an average-case lower bound for GIP_n^k against $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{PTF}^{k-1}$, i.e., sub-quadratic-size de Morgan formulas with degree-$(k-1)$ PTF (polynomial threshold function) gates at the bottom.
- There is a PRG of seed length $n/2 + O\left(\sqrt{s} \cdot R^{(2)}(\mathcal{G}) \cdot \log(s/\varepsilon) \cdot \log(1/\varepsilon)\right)$ that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$. For the special case of $\mathsf{FORMULA}[s] \circ \mathsf{LTF}$, i.e., size-$s$ formulas with LTF (linear threshold function) gates at the bottom, we get the better seed length $O\left(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)\right)$. In particular, this provides the first non-trivial PRG (with seed length $o(n)$) for intersections of n half-spaces in the regime where $\varepsilon \le 1/n$, complementing a recent result of [44].
- There exists a randomized 2^{n-t}-time #SAT algorithm for $\mathsf{FORMULA}[s] \circ \mathcal{G}$, where

$$t = \Omega\left(\frac{n}{\sqrt{s} \cdot \log^2(s) \cdot R^{(2)}(\mathcal{G})}\right)^{1/2}.$$

  In particular, this implies a nontrivial #SAT algorithm for $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{LTF}$.
- The Minimum Circuit Size Problem is not in $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$; thereby making progress on hardness magnification, in connection with results from [45, 12]. On the algorithmic side, we show that the concept class $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$ can be PAC-learned in time $2^{O(n/\log n)}$.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 15; pp. 15:1–15:41
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Circuit complexity; Theory of computation → Communication complexity; Theory of computation → Complexity classes

Keywords and phrases de Morgan formulas, circuit lower bounds, satisfiability (SAT), pseudorandom generators (PRGs), learning, communication complexity, polynomial threshold functions (PTFs), parities

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.15

Related Version A full version of the paper is available at https://eccc.weizmann.ac.il/report/2020/018/.

Funding *Igor C. Oliveira*: This work was funded in part by a Royal Society University Research Fellowship (URF\R1\191059).

Acknowledgements We would like to thank Rocco Servedio for bringing to our attention the work by Kalai, Mansour, and Verbin [36], which is a central ingredient in the proof of Theorem 7. We also thank Mahdi Cheraghchi for several discussions on the analysis of Boolean circuits with a bottom layer of parity gates.

1 Introduction

A (de Morgan) Boolean formula over $\{0,1\}$-valued input variables $x_1, \ldots, x_n$ is a binary tree whose internal nodes are labelled by AND or OR gates, and whose leaves are marked with a variable or its negation. The power of Boolean formulas has been intensively investigated since the early years of complexity theory (see, e.g., [57, 42, 38, 5, 47, 30, 27, 58, 19]). The techniques underlying these complexity-theoretic results have also enabled algorithmic developments. These include learning algorithms [52, 55], satisfiability algorithms (cf. [59]), compression algorithms [14], and the construction of pseudorandom generators [29] for Boolean formulas of different sizes. But despite many decades of research, the current non-trivial algorithms and lower bounds apply only to formulas of less than cubic size, and understanding larger formulas remains a major open problem in circuit complexity.

In many scenarios, however, understanding smaller formulas whose leaves are replaced by certain functions would also be very useful. Motivated by several recent works, we initiate a systematic study of the $\mathsf{FORMULA} \circ \mathcal{G}$ model, i.e., Boolean formulas whose leaves are labelled by an arbitrary function from a fixed class $\mathcal{G}$. This model unifies and generalizes a variety of models that have been previously studied in the literature:

- Oliveira, Pich, and Santhanam [45] show that proving certain lower bounds against formulas of size $n^{1+\varepsilon}$ over parity (XOR) gates would have significant consequences in complexity theory. Note that de Morgan formulas of size $n^{3+\varepsilon}$ can simulate such devices. Therefore, a better understanding of the $\mathsf{FORMULA} \circ \mathcal{G}$ model even when $\mathcal{G} = \mathsf{XOR}$ is *necessary* before we are able to analyze super-cubic size formulas.[1]
- Tal [61] obtains almost quadratic lower bounds for the model of bipartite formulas, where there is a fixed partition of the input variables into $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, and a formula leaf can compute an *arbitrary* function over either $\vec{x}$ or $\vec{y}$. This model was originally investigated by Pudlák, Rödl, and Savický [49], where it was referred to as graph complexity. The model is also equivalent to PSPACE-protocols in communication complexity (cf. [23]).

[1] We remark that even a single layer of XOR gates can compute powerful primitives, such as error-correcting codes and hash functions.

- Abboud and Bringmann [1] consider formulas where the leaves are threshold gates whose input wires can be arbitrary functions applied to either the first or the second half of the input. This extension of bipartite formulas is denoted by $\mathcal{F}_2$ in [1]. Their work establishes connections between faster $\mathcal{F}_2$-SAT algorithms, the complexity of problems in P such as Longest Common Subsequence and the Fréchet Distance Problem, and circuit lower bounds.
- Polytopes (i.e. intersection of half-spaces), which corresponds to $\mathcal{G}$ being the family of linear-threshold functions, and the formula contains only AND gates as internal gates. The constructing of PRGs for this model has received significant attention in the literature (see [44] and references therein).

We obtain in a unified way several new results for the $\mathsf{FORMULA} \circ \mathcal{G}$ model, for natural classes $\mathcal{G}$ of functions which include parities, linear (and polynomial) threshold functions, and indeed many other functions of interest. In particular, we show that this perspective leads to stronger lower bounds, general satisfiability algorithms, and better pseudorandom generators for a broad class of functions.

1.1 Results

We now describe in detail our main results and how they contrast to previous works. Our techniques will be discussed in Section 1.2, while a few open problems are mentioned in Section 1.3.

We let $\mathsf{FORMULA}[s] \circ \mathcal{G}$ denote the set of Boolean functions computed by formulas containing at most s leaves, where each leaf computes according to some function in $\mathcal{G}$. The set of parity functions and their negations will be denoted by XOR.

We use the following notation for communication complexity. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we let $D(f)$ be the two-party deterministic communication complexity of f, where each party is given an input of $n/2$ bits. Similarly, for a Boolean function $g\colon \{0,1\}^n \to \{0,1\}$, we denote by $R^{(k)}_\delta(g)$ the communication cost of the best k-party *number-on-forehead* (NOF) communication protocol that computes g with probability at least $1-\delta$ on every input, where the probability is taken over the random choices of the protocol. For simplicity, we might omit the superscript (k) from $R^{(k)}_\delta(g)$ when $k=2$. One of our results will also consider k-party *number-in-hand* (NIH) protocols, and this will be clearly indicated in order to avoid confusion. We always assume a canonical partition of the input coordinates in all statements involving k-party communication complexity, unless stated otherwise. We generalize these definitions for a class of functions $\mathcal{G}$ in the natural way. For instance, we let $R^{(k)}_\delta(\mathcal{G}) = \max_{g\in\mathcal{G}} R^{(k)}_\delta(g)$.

Our results refer to standard notions in the literature, but in order to fix notation, Section 2 formally defines communication protocols, Boolean formulas, and other notions relevant in this work. We refer to the textbooks [39] and [32] for more information about communication complexity and Boolean formulas, respectively. To put our results into context, here we only briefly review a few known upper bounds on the communication complexity of certain classes $\mathcal{G}$.

Parities (XOR) and Bipartite Formulas. Clearly, the deterministic two-party communication complexity of any parity function is at most 2, since to agree on the output it is enough for the players to exchange the parity of their relevant input bits. Moreover, note that the bipartite formula model discussed above precisely corresponds to formulas whose leaves are computed by a two-party protocol of communication cost at most 1.

Halfspaces and Polynomial Threshold Functions (PTFs). Recall that a halfspace, also known as a Linear Threshold Function (LTF), is a Boolean function of the form $\mathsf{sign}(\sum_i^n a_i \cdot x_i - b)$, where each $a_i, b \in \mathbb{R}$ and $x \in \{0,1\}^n$, and that a degree-d PTF is its natural generalization where degree-d monomials are allowed. It is known that if $g(x_1, \ldots, x_n)$ is a halfspace, then its randomized two-party communication complexity, namely $R^{(2)}_\delta(g)$, satisfies $R^{(2)}_\delta(g) = O(\log(n) + \log(1/\delta))$ [43]. On the other hand, if $g(x_1, \ldots, x_n)$ is a degree-d PTF, then $R^{(d+1)}_\delta(g) = O\big((d \log d)(d \log n + \log(1/\delta))\big)$ [43, 64].

Degree-d Polynomials over GF(2). It is well known that a degree-d $\mathsf{GF}(2)$-polynomial admits a $(d+1)$-party deterministic protocol of communication cost $d+1$ under *any* variable partition, since in the number-on-forehead model each monomial is entirely seen by some player. In particular, the Inner Product function $\mathsf{IP}_n(x,y) = \sum_i x_i \cdot y_i \pmod 2$ satisfies $R^{(3)}_{1/3}(\mathsf{IP}_n) = O(1)$.

1.1.1 Lower bounds

Prior to this work, the only known lower bound against $\mathsf{FORMULA} \circ \mathsf{XOR}$ or bipartite formulas was the recent result of [61] showing that IP_n is hard (even on average) against nearly sub-quadratic formulas. In contrast, we obtain a significantly stronger result and establish lower bounds for different Boolean functions. We define such functions next.

GIP^k_n. The Generalized Inner Product function $\mathsf{GIP}^k_n \colon \{0,1\}^n \to \{0,1\}$ is defined as

$$\mathsf{GIP}^k_n\left(x^{(1)}, x^{(2)}, \ldots, x^{(k)}\right) = \sum_{j=1}^{n/k} \bigwedge_{i=1}^{k} x^{(i)}_j \pmod 2,$$

where $x^{(i)} \in \{0,1\}^{n/k}$ for each $i \in [k]$.

MKtP. In the Minimum Kt Problem, where Kt refers to Levin's time-bounded Kolmogorov complexity[2], we are given a string $x \in \{0,1\}^n$ and a string 1^ℓ. We accept $(x, 1^\ell)$ if and only if $\mathsf{Kt}(x) \leq \ell$.

MCSP. In the Minimum Circuit Size Problem, we are given as input the description of a Boolean function $f \colon \{0,1\}^{\log n} \to \{0,1\}$ (represented as an n-bit string), and a string 1^ℓ. We accept $(f, 1^\ell)$ if and only the circuit complexity of f is at most ℓ.

▶ **Theorem 1** (Lower bounds). *The following unconditional lower bounds hold:*

1. *If GIP^k_n is $(1/2+\varepsilon)$-close under the uniform distribution to a function in $\mathsf{FORMULA}[s] \circ \mathcal{G}$, then*

$$s = \Omega\left(\frac{n^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right)^2 \cdot \log^2(1/\varepsilon)}\right).$$

[2] For a string $x \in \{0,1\}^*$, $\mathsf{Kt}(x)$ denotes the minimum value $|M| + \log t$ taken over M and t, where M is a machine that prints x when it computes for t steps, and $|M|$ is the description length of M according to a fixed universal machine U.

2. *If* $\mathsf{MKtP} \in \mathit{FORMULA}[s] \circ \mathcal{G}$*, then*

$$s = \widetilde{\Omega}\left(\frac{n^2}{k^2 \cdot 16^k \cdot R^{(k)}_{1/3}(\mathcal{G})}\right).$$

3. *If* $\mathsf{MCSP} \in \mathit{FORMULA}[s] \circ \mathsf{XOR}$*, then* $s = \widetilde{\Omega}(n^2)$*, where* $\widetilde{\Omega}$ *hides inverse* $\mathsf{polylog}(n)$ *factors.*

Observe that, while [61] showed that the Inner Product function IP_n is hard against sub-quadratic bipartite formulas, Theorem 1 Item 1 yields lower bounds against formulas whose leaves can compute bounded-degree PTFs and $\mathsf{GF}(2)$-polynomials, including IP_n. Previously, only sub-linear lower bounds were known [43, 64] for circuits with PTF gates of similar degree.

Let us now comment on the relevance of Items 2 and 3. Both MCSP and MKtP are believed to be computationally much harder than GIP^k_n. However, it is more difficult to analyze these problems compared to GIP^k_n because the latter is mathematically "structured," while the former problems do not seem to be susceptible to typical algebraic, combinatorial, and analytic techniques.

More interestingly, MCSP and MKtP play an important role in the theory of hardness magnification (see [45, 12]). In particular, if one could show that MCSP restricted to an input parameter $\ell \leq n^{o(1)}$ is not in $\mathsf{FORMULA}[n^{1+\varepsilon}] \circ \mathsf{XOR}$ for some $\varepsilon > 0$, then it would follow that NP cannot be computed by Boolean formulas of size n^c, where $c \in \mathbb{N}$ is arbitrary. Theorem 1 makes partial progress on this direction by establishing the first lower bounds for these problems in the $\mathsf{FORMULA} \circ \mathcal{G}$ model. (We note that the proof of Theorem 1 Item 3 requires instances where the parameter ℓ is $n^{\Omega(1)}$.)

1.1.2 Pseudorandom generators

We also get pseudorandom generators (PRGs) against $\mathsf{FORMULA} \circ \mathcal{G}$ for various classes of functions $\mathcal{G}$. Recall that a PRG against a class of functions $\mathfrak{C}$ is a function G mapping short Boolean strings (seeds) to longer Boolean strings, so that every function in $\mathfrak{C}$ accepts G's output on a uniformly random seed with about the same probability as that for an actual uniformly random string. More formally, $G\colon \{0,1\}^\ell \to \{0,1\}^n$ is a PRG that ε-fools $\mathfrak{C}$ if for every Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ in $\mathfrak{C}$, we have

$$\left| \mathop{\mathbf{Pr}}_{z\in\{0,1\}^\ell}[h(G(z)) = 1] - \mathop{\mathbf{Pr}}_{x\in\{0,1\}^n}[h(x) = 1] \right| \leq \varepsilon.$$

Furthermore, we require G to run in deterministic time $\mathsf{poly}(n)$ on an input string $z \in \{0,1\}^\ell$. The parameter $\ell = \ell(n)$ is called the seed length of the PRG and is the main quantity to be minimized when constructing PRGs.

There exists a PRG that fools formulas of size s and that has a seed of length $s^{1/3+o(1)}$ [29]. In particular, there are non-trivial PRGs for n-variate formulas of size nearly n^3. Unfortunately, such PRGs cannot be used to fool even linear size formulas over parity functions, since the naive simulation of these enhanced formulas by standard Boolean formulas requires size n^3. Moreover, it is not hard to see that this simulation is optimal: Andreev's function, which is hard against formulas of nearly cubic size (cf. [27]), can be easily computed in $\mathsf{FORMULA}[O(n)] \circ \mathsf{XOR}$. Given that a crucial idea in the construction of the PRG in [29] (shrinkage under restrictions) comes from this lower bound proof, new techniques are needed in order to approach the problem in the $\mathsf{FORMULA} \circ \mathsf{XOR}$ model.

More generally, extending a computational model for which strong PRGs are known to allow parities at the bottom layer can cause significant difficulties. A well-known example is AC^0 circuits and their extension to AC^0-XOR. While the former class admits PRGs of poly-logarithmic seed length (see e.g. [56]), the most efficient PRG construction for the latter has seed length $(1-o(1))\cdot n$ [21]. Consequently, designing PRGs of seed length $\leq (1-\Omega(1))\cdot n$ can already be a challenge. We are not aware of previous results on PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ for any non-trivial class $\mathcal{G}$.

By combining ideas from circuit complexity and communication complexity, we construct PRGs of various seed lengths for $\mathsf{FORMULA} \circ \mathcal{G}$, where $\mathcal{G}$ ranges from the class of parity functions to the much larger class of functions of bounded randomized k-party communication complexity.

▶ **Theorem 2** (Pseudorandom generators). *Let $\mathcal{G}$ be a class of n-bits functions. Then,*

1. *In the context of parity functions, there is a* PRG *that ε-fools $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ of seed length*

$$\ell \;=\; O\big(\sqrt{s}\cdot \log(s)\cdot \log(1/\varepsilon) + \log(n)\big)\,.$$

2. *In the context of two-party randomized communication complexity, there is a* PRG *that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$ of seed length*

$$\ell \;=\; n/2 + O\Big(\sqrt{s}\cdot \Big(R^{(2)}_{\varepsilon/(6s)}(\mathcal{G}) + \log(s)\Big)\cdot \log(1/\varepsilon)\Big)\,.$$

More generally, for every $k(n) \geq 2$, let $\mathcal{G}$ be the class of functions that have k-party number-in-hand (NIH) *$(\varepsilon/6s)$-error randomized communication protocols of cost at most $R^{(k\text{-NIH})}_{\varepsilon/(6s)}$. There exists a* PRG *that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$ with seed length*

$$\ell \;=\; n/k + O\Big(\sqrt{s}\cdot \Big(R^{(k\text{-NIH})}_{\varepsilon/(6s)} + \log(s)\Big)\cdot \log(1/\varepsilon) + \log(k)\Big)\cdot \log(k).$$

3. *In the setting of k-party* NOF *randomized communication complexity, there is a* PRG *that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$ of seed length*

$$\ell \;=\; n - \frac{n}{O\Big(\sqrt{s}\cdot k\cdot 4^k\cdot \Big(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G}) + \log(n)\Big)\cdot \log(n/\varepsilon)\Big)}.$$

A few comments are in order. Under a standard connection between PRGs and lower bounds (see e.g. [33]), improving the dependence on s in the seed length for $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ (Theorem 2 Item 1) would require the proof of super-quadratic lower bounds against $\mathsf{FORMULA} \circ \mathsf{XOR}$. We discuss this problem in more detail in Section 1.3. Note that the additive term $n/2$ is necessary in Theorem 2 Item 2, since the model computes in particular every Boolean function on the first $n/2$ input variables (i.e. a protocol of communication cost 1). Similarly, $\ell \geq (1-1/k)\cdot n$ in Theorem 2 Item 3. Removing the exponential dependence on k would also require advances in state-of-the-art lower bounds for multiparty communication complexity.

Theorem 2 Item 2 has an interesting implication for fooling a well-studied class of functions: *intersections of halfspaces.*[3] Note that an intersection of halfspaces is precisely a polytope, or equivalently, the set of solutions of a 0-1 integer linear program. Such objects have found applications in many fields, including optimization and high-dimensional geometry.

[3] Clearly, the intersection of s functions can be computed by an enhanced formula of size $s+1$.

After a long sequence of works on the construction of PRGs for bounded-weight halfspaces, (unrestricted) halfspaces, and generalizations of these classes,[4] the following results are known for the intersection of m halfspaces over n input variables. Gopalan, O'Donnell, Wu, and Zuckerman [24] gave a PRG for this class for error ε with seed length

$$O\big(m \cdot \log(m/\varepsilon) + \log n) \cdot \log(m/\varepsilon)\big).$$

Note that the seed length of their PRG becomes trivial if the number of halfspaces is linear in n. More recently, O'Donnell, Servedio and Tan [44] constructed a PRG with seed length

$$\mathsf{poly}(\log(m), 1/\varepsilon) \cdot \log(n).$$

Their PRG has a much better dependence on m, but it cannot be used in the small error regime. For example, the seed length becomes trivial if $\varepsilon = 1/n$. In particular, before this work it was open to construct a non-trivial PRG for the following natural setting of parameters (cf. [44, Section 1.2]): intersection of n halfspaces with error $\varepsilon = 1/n$.

We obtain the following consequence of Theorem 2 Item 2, which follows from a result of Viola [64] on the k-party *number-in-hand* randomized communication complexity of a halfspace.

▶ **Corollary 3** (Fooling intersections of halfspaces in the low-error regime). *For every $n, m \in \mathbb{N}$ and $\varepsilon > 0$, there is a pseudorandom generator with seed length*

$$O\Big(n^{1/2} \cdot m^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)\Big) .$$

that ε-fools the class of intersections of m halfspaces over $\{0,1\}^n$.

We note that the PRG from Theorem 2 Item 3 can fool, even in the exponentially small error regime, not only intersections of halfspaces, but also small formulas over bounded-degree PTFs.

Finally, Theorem 2 Item 2 yields the first non-trivial PRG for formulas over symmetric functions. Let SYM denote the class of symmetric Boolean functions on any number of input variables.

▶ **Corollary 4** (Fooling sub-quadratic formulas over symmetric gates). *For every $n, s \in \mathbb{N}$ and $\varepsilon > 0$, there is a pseudorandom generator with seed length*

$$O\Big(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(1/\varepsilon)\Big) .$$

that ε-fools n-variate Boolean functions in $\mathsf{FORMULA}[s] \circ \mathsf{SYM}$.

Prior to this work, Chen and Wang [13] proved that the number of satisfying assignments of an n-variate formula of size s over symmetric gates can be approximately counted to an additive error term $\leq \varepsilon \cdot 2^n$ in deterministic time $\exp(n^{1/2} \cdot s^{1/4+o(1)}\sqrt{(\log(n) + \log(s))})$, where $\varepsilon > 0$ is an arbitrary constant. While their upper bound is achieved by a white-box algorithm, Corollary 4 provides a (black-box) PRG for the same task.

[4] We refer to the recent reference [44] for an extensive review of the literature in this area.

1.1.3 Satisfiability algorithms

In the #SAT problem for a computational model $\mathcal{C}$, we are given as input the description of a computational device $D(x_1, \ldots, x_n)$ from $\mathcal{C}$, and the goal is to count the number of satisfying assignments for D. This generalizes the SAT problem for $\mathcal{C}$, where it is sufficient to decide whether D is satisfiable by some assignment.

In this section, we show that #SAT algorithms can be designed for a broad class of functions. We consider the $\mathsf{FORMULA} \circ \mathcal{G}$ model for classes $\mathcal{G}$ that admit two-party communication protocols of bounded cost. We establish a general result in this context which can be used to obtain algorithms for previously studied classes of Boolean circuits.

To put our #SAT algorithms for $\mathsf{FORMULA} \circ \mathcal{G}$ into context, we first mention relevant related work on the satisfiability of Boolean formulas. Recall that in the very restricted setting of CNF formulas, known algorithms run (in the worst-case) in time $2^{n-o(n)}$ when the input formulas can have a super-linear number of clauses (cf. [18]). On the other hand, for the class of general formulas, there is a better-than-brute-force algorithm for formulas of size almost n^3. In more detail, for any $\varepsilon > 0$, there is a deterministic #SAT algorithm for $\mathsf{FORMULA}[n^{3-\varepsilon}]$ that runs in time $2^{n-n^{\Omega(\varepsilon)}}$ [59]. No results are known for formulas of cubic size and beyond, and for the reasons explained in Section 1.1.2, the algorithm from [59] cannot even be applied to $\mathsf{FORMULA} \circ \mathsf{XOR}$.

Before stating our results, we discuss the input encoding in the #SAT problem for $\mathsf{FORMULA} \circ \mathcal{G}$. The top formula F is represented in some canonical way, while for each leaf ℓ of F, the input string contains the description of a protocol Π_ℓ computing a function in $\mathcal{G}$. Our results are robust to the encoding employed for Π_ℓ. Recall that a protocol for a two-party function is specified by a protocol tree and a sequence of functions, where each function is associated with some internal node of the tree and depends on $n/2$ input bits. Since a protocol of communication cost $o(n)$ has a protocol tree containing at most $2^{o(n)}$ nodes, it can be specified by a string of length $2^{n/2+o(n)}$. Our algorithms will run in time closer to 2^n, and using a fully explicit input representation for the protocols is not an issue. Another possibility for the input representation is to use "computational efficient" protocols. Informally, the next bit messages of such protocols can be computed in polynomial time from the current transcript of the protocol and a player input. An advantage of this representation is that an input to our #SAT problem can be succinctly represented. We observe that these input representations can be generalized to randomized two-party protocols in natural ways. We refer to Section 2 for a formal presentation.

We obtain non-trivial satisfiability algorithms assuming upper bounds on the two-party deterministic and randomized communication complexities of functions in $\mathcal{G}$.

▶ **Theorem 5** (Satisfiability algorithms). *The following results hold.*

1. *There is a deterministic* #SAT *algorithm for $\mathsf{FORMULA}[s] \circ \mathcal{G}$ that runs in time*

$$2^{n-t}, \text{ where } t = \Omega\left(\frac{n}{\sqrt{s} \cdot \log(s) \cdot (\log(s) + D(\mathcal{G}))}\right).$$

2. *There is a randomized* #SAT *algorithm for $\mathsf{FORMULA}[s] \circ \mathcal{G}$ that runs in time*

$$2^{n-t}, \text{ where } t = \Omega\left(\frac{n}{\sqrt{s} \cdot \log^2(s) \cdot R_{1/3}(\mathcal{G})}\right)^{1/2}.$$

Theorem 5 readily provides algorithms for many circuit classes. For instance, since one can effectively describe a randomized communication protocol for linear threshold functions [43, 64], the algorithm from Theorem 5 Item 2 can be used to count the number of satisfying assignments of Boolean devices from $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{LTF}$.

▶ **Corollary 6** (#SAT algorithm for formulas of linear threshold functions)**.** *There is a randomized* #SAT *algorithm for* $\mathsf{FORMULA}[s] \circ \mathsf{LTF}$ *that runs in time*

$$2^{n-t}, \text{ where } t = \Omega\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot\log(n)}\right)^{1/2}.$$

In connection with Corollary 6, prior to this work essentially two lines of research have been pursued. #SAT and/or SAT algorithms were known for bounded-depth circuits of almost-linear size whose gates can compute LTFs or (low-degree) PTFs (see [15, 34, 8]), and for sub-exponential size ACC^0 circuits with two layers of LTFs at the bottom, assuming a sub-quadratic number of them in the layer next to the input variables (see [2] for this result and further related work). Corollary 6 seems to provide the first non-trivial SAT algorithm that operates with unbounded-depth Boolean devices containing a layer with a sub-quadratic number of LTFs.

Theorem 5 can be seen as a generalization of several approaches to designing SAT algorithms appearing in the literature, which often employ ad-hoc constructions to convert bottlenecks in the computation of devices from a class $\mathcal{C}$ into non-trivial SAT algorithms for $\mathcal{C}$. We observe that, before this work, [48] had made a connection between faster SAT algorithms for CNFs and the 3-party communication complexity of a specific function. Their setting is different though: it seems to work only for CNFs, and they rely on conjectured upper bounds on the communication complexity of a particular problem. More recently, [13] employed quantum communication protocols to design *approximate counting* algorithms for several problems.[5] In comparison to previous works, to our knowledge Theorem 5 is the first unconditional result that yields faster #SAT algorithms via communication complexity in a generic way.[6]

1.1.4 Learning algorithms

We describe a learning algorithm for the $\mathsf{FORMULA} \circ \mathsf{XOR}$ class in Leslie Valiant's challenging PAC-learning model [63]. Recall that a (PAC) learning algorithm for a class of functions $\mathcal{C}$ has access to labelled examples $(x, f(x))$ from an unknown function $f \in \mathcal{C}$, where x is sampled according to some (also unknown) distribution $\mathcal{D}$. The goal of the learner is to output, with high probability over its internal randomness and over the choice of random examples (measured by a confidence parameter δ), a hypothesis h that is close to f under $\mathcal{D}$ (measured by an error parameter ε). We refer to [37] for more information about this learning model, and to Section 2 for its standard formalization.

It is known that formulas of size s can be PAC-learned in time $2^{\widetilde{O}(\sqrt{s})}$ [52]. Therefore, formulas of almost quadratic size can be non-trivially learned from random samples of an arbitrary distribution. A bit more formally, we say that a learning algorithm is *non-trivial* if it runs in time $2^n/n^{\omega(1)}$, i.e., noticeably faster than the trivial brute-force algorithm that takes time $2^n \cdot \mathsf{poly}(n)$. Obtaining non-trivial learning algorithms for various circuit classes is closely connected to the problem of proving explicit lower bounds against the class [46] (see also [55] for a systematic investigation of such algorithms). We are not aware of the existence of non-trivial learning algorithms for super-quadratic size formulas. However, it

[5] Recall that approximately counting satisfying assignments is substantially easier than solving #SAT, for which the fastest known algorithms run in time $2^{(1-o(1))n}$.

[6] It has been brought to our attention that Avishay Tal has independently discovered a SAT algorithm for bipartite formulas of sub-quadratic size (see the discussion in [1, Page 7]), which corresponds to a particular case of Theorem 5.

seems likely that such algorithms exist at least for formulas of near cubic size. As explained in Section 1.1.2, this would still be insufficient for the learnability of classes such as (linear size) $\mathsf{FORMULA} \circ \mathsf{XOR}$.

We explore structural properties of $\mathsf{FORMULA} \circ \mathsf{XOR}$ employed in previous results and boosting techniques from learning theory to show that sub-quadratic size devices from this class can be PAC-learned in time $2^{O(n/\log n)}$.

▶ **Theorem 7** (PAC-learning $\mathsf{FORMULA} \circ \mathsf{XOR}$ in sub-exponential time)**.** *For every constant $\gamma > 0$, there is an algorithm that* PAC *learns the class of n-variate Boolean functions $\mathsf{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ to accuracy ε and with confidence δ in time* $\mathsf{poly}\big(2^{n/\log n}, 1/\varepsilon, \log(1/\delta)\big)$.

Note that a sub-exponential running time cannot be achieved for $\mathsf{FORMULA} \circ \mathcal{G}$ when we consider the communication complexity of $\mathcal{G}$. Again, the class is too large, for the same reason discussed in Section 1.1.2. It might still be possible to design a non-trivial learning algorithm in this case, but this would possibly require the introduction of new lower bound techniques for $\mathsf{FORMULA} \circ \mathsf{XOR}$.

In contrast to the algorithm mentioned above that learns (standard) formulas of size $s \leq n^{2-o(1)}$ in time $2^{\widetilde{O}(\sqrt{s})}$, the algorithm from Theorem 7 does not learn smaller formulas over parities in time faster than $2^{O(n/\log n)}$. We discuss this in more detail in Sections 1.2 and 1.3.

Finally, we mention a connection to cryptography that provides a conditional upper bound on the size of $\mathsf{FORMULA} \circ \mathsf{XOR}$ circuits that can be learned in time $2^{o(n)}$. It is well known that if a circuit class $\mathcal{C}$ can compute pseudorandom functions (or some variants of this notion), then it cannot be learned in various learning models (see e.g. [37]). It has been recently conjectured that depth-two $\mathsf{MOD}_3 \circ \mathsf{XOR}$ circuits of linear size can compute weak pseudorandom functions of exponential security [10, Conjecture 3.7]. If this conjecture holds, then such circuits cannot be learned in time $2^{o(n)}$. Since MOD_3 gates over a linear number of input wires can be simulated by formulas of size at most $O(n^{2.8})$ [54], under this cryptographic assumption it is not possible to learn $\mathsf{FORMULA}[n^{2.8}] \circ \mathsf{XOR}$ in time $2^{o(n)}$, even if the learner only needs to succeed under the uniform distribution.

1.2 Techniques

In order to explain our techniques, we focus for the most part on the design of PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ when $\mathcal{G}$ is of bounded two-party randomized communication complexity (a particular case of Theorem 2 Item 2). This proof makes use of various ingredients employed in other results. After sketching this argument, we say a few words about our strongest lower bound (Theorem 1 Item 1) and the satisfiability and learning algorithms (Theorems 5 and 7, respectively).

We build on a powerful result showing that any small de Morgan formula can be approximated pointwise by a low-degree polynomial:

(A) For every formula $F(y_1, \dots, y_m)$ of size s, there is a polynomial $p(y_1, \dots, y_m) \in \mathbb{R}[y_1, \dots, y_m]$ of degree $O(\sqrt{s})$ such that $|F(a) - p(a)| \leq 1/10$ on every $a \in \{0,1\}^m$.

The only known proof of this result [52] relies on a sequence of works [9, 40, 28, 20, 50, 4, 53] on quantum query complexity, generalizing Grover's search algorithm for the OR predicate [25] to arbitrary formulas. The starting point of many of our results is a consequence of **(A)** which is implicit in the work of Tal [61].

(B) Let $\mathcal{D}$ be a distribution over $\{0,1\}^m$, and $F \in \mathsf{FORMULA}[s] \circ \mathcal{G}$. Then, for every function f,

$$\text{if} \quad \Pr_{x \sim \mathcal{D}}[F(x) = f(x)] \geq 1/2 + \varepsilon, \quad \text{then} \quad \Pr_{x \sim \mathcal{D}}[h(x) = f(x)] \geq 1/2 + \exp(-t)$$

for some function h which is the XOR of at most t functions in $\mathcal{G}$, where $t = \widetilde{\Theta}(\sqrt{s} \cdot \log(1/\varepsilon))$.

Intuitively, if we could understand well enough the XOR of any small collection of functions in $\mathcal{G}$, then we can translate this into results for $\mathsf{FORMULA}[s] \circ \mathcal{G}$, as long as $s \ll n^2$. We adapt the techniques behind **(B)** to provide a general approach to constructing PRGs against $\mathsf{FORMULA} \circ \mathcal{G}$:

Main PRG Lemma. In order for a distribution $\mathcal{D}$ to ε-fool the class $\mathsf{FORMULA}[s] \circ \mathcal{G}$, it is enough for it to $\exp(-t)$-fool the class $\mathsf{XOR}_t \circ \mathcal{G}$, where $t = \widetilde{\Theta}(\sqrt{s} \cdot \log(1/\varepsilon))$.

Recall that, in Theorem 2 Item 2, we consider a class $\mathcal{G}$ of functions that admit two-party randomized protocols of cost $R = R^{(2)}_{\varepsilon/6s}(\mathcal{G})$. It is easy to see that the XOR of any t functions from $\mathcal{G}$ is a function that can be computed by a protocol of cost at most $t \cdot R$. Thus the lemma above shows that it is sufficient to fool, to exponentially small error, a class of functions of bounded two-party randomized communication complexity. Moreover, since a randomized protocol can be written as a convex combination of deterministic protocols, it is possible to prove that fooling functions of bounded deterministic communication complexity is enough.

Pseudorandom generators in the two-party communication model have been known since [31]. Their construction exploits that the Boolean matrix associated with a function of small communication cost can be partitioned into a not too large number of monochromatic rectangles. We provide in Appendix A.2 a slightly modified and self-contained construction based on explicit extractors. It achieves the following parameters: There is an explicit PRG that δ-fools any n-bit function of two-party communication cost D and that has seed length $n/2 + O(D + \log(1/\delta))$. This PRG has non-trivial seed length even when the error is exponentially small, as required by our techniques. One issue here is that the INW PRG was only shown to fool functions with low *deterministic* communication complexity. To obtain our PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ when $\mathcal{G}$ admits low-cost *randomized* protocols, we first extend the analysis of the INW PRG to show that it also fools functions with low *randomized* communication complexity. Combining this construction with the aforementioned discussion completes the proof of Theorem 2 Item 2.

The argument just sketched reduces the construction of PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ when functions in $\mathcal{G}$ admit low-cost *randomized* protocols to the analysis of PRGs for functions that admit relatively low-cost *deterministic* protocols. Our lower bound proof for GIP^k_n in Theorem 1 Item 1 proceeds in a similar fashion. We combine statement **(B)** described above with other ideas to show:

Transfer Lemma (Informal). If a function correlates with some small formula whose leaf gates have low-cost *randomized* k-party protocols, then it also non-trivially correlates with some function that has relatively low-cost *deterministic* k-party protocols.

Given this result, we are able to rely on a strong average-case lower bound for GIP^k_n against k-party deterministic protocols from [7] to conclude that GIP^k_n is hard for $\mathsf{FORMULA} \circ \mathcal{G}$.

Our #SAT algorithms combine the polynomial representation of the top formula provided by **(A)**, for which we show that such a polynomial can be obtained *explicitly*, with a decomposition of the Boolean matrix at each leaf that is induced by a corresponding low-cost randomized or deterministic two-party protocol. A careful combination of these two representations allows us to adapt a standard technique employed in the design of non-trivial SAT algorithms (fast rectangular matrix multiplication) to obtain non-trivial savings in the running time.

Finally, our learning algorithm for $\mathsf{FORMULA} \circ \mathsf{XOR}$ is a consequence of statement **(B)** above coupled with standard tools from learning theory. In a bit more detail, since a parity of parities is just another parity function, **(B)** implies that, under any distribution, every function in $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$ is weakly correlated with some parity function. Using the agnostic learning algorithm for parity functions of [36], it is possible to weakly learn $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$ in time $2^{O(n/\log n)}$. This weak learner can then be transformed into a (strong) PAC learner using standard boosting techniques [22], with only a polynomial blow-up over its running time.

1.3 Concluding remarks

The main message of our results is that the *computational power* of a subquadratic-size top formula is *not* significantly enhanced by leaf gates of *low communication complexity.* We believe that the idea of decomposing a Boolean device into a computational part and a layer of communication protocols will find further applications in lower bound proofs and algorithm design.

One of our main open problems is to discover a method that can analyze $\mathsf{FORMULA}[s] \circ \mathcal{G}$ when $s \gg n^2$. For instance, is it possible to adapt existing techniques to show an explicit lower bound against $\mathsf{FORMULA}[n^{2.01}] \circ \mathcal{G}$, or achieving this is just as hard as breaking the cubic barrier for formula lower bounds? Results in this direction would be interesting even for $\mathcal{G} = \mathsf{XOR}$.

Finally, we would like to mention a few questions connected to our results and their applications. Is it possible to combine the techniques behind Corollary 3 and [44] to design a PRG of seed length $n^{o(1)}$ and error $\varepsilon = 1/n$ for the intersection of n halfspaces? Can we design a satisfiability algorithm for formulas over k-party number-on-forehead communication protocols? Is it possible to learn $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ in time $2^{\widetilde{O}(\sqrt{s})}$? (The learning algorithm for formulas from [52] relies on techniques from [35], and it is unclear how to extend them to the case of $\mathsf{FORMULA} \circ \mathsf{XOR}$.)

1.4 Organization

Theorem 1 Item 1 is proved in Section 3, while Items 2 and 3 rely on our PRG constructions and are deferred to Section 4. The latter describes a general approach to constructing PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$. It includes the proof of Theorem 2 and other applications. Our satisfiability algorithms (Theorem 5) appear in Section 5. Finally, Section 6 discusses learning results for $\mathsf{FORMULA} \circ \mathsf{XOR}$ and contains a proof of Theorem 7.

2 Preliminaries

2.1 Notation

Let $n \in \mathbb{N}$; we denote $\{1, \ldots, n\}$ by $[n]$, and denote by U_n the uniform distribution over $\{0,1\}^n$. We use $\widetilde{O}(\cdot)$ (and $\widetilde{\Omega}(\cdot)$) to hide polylogarithmic factors. That is, for any $f \colon \mathbb{N} \to \mathbb{N}$, we have that $\widetilde{O}(f(n)) = O(f(n) \cdot \mathsf{polylog}(f(n)))$.

In this paper, we will mainly use $\{-1, 1\}$ as the Boolean basis. In some parts of this paper, we will use the $\{0, 1\}$ basis for the simplicity of the presentation. This will be specified in corresponding sections.

2.2 De Morgan formulas and extensions

▶ **Definition 8.** *An n-variate* de Morgan formula *is a directed rooted tree; its non-leaf vertices (henceforth,* internal gates*) take labels from* $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\} = \{\wedge, \vee, \neg\}$ *and its leaves (henceforth,* variable gates*) take labels from the set of variables* $\{x_1, \ldots, x_n\}$*. Each internal gate has bounded in-degree (henceforth,* fan-in*); the* NOT *gate in particular has fan-in* 1 *and every variable gate has fan-in* 0*. The* size *of a de Morgan formula is the number of its leaf gates.*

In this work, we denote by FORMULA$[s]$ the class of Boolean functions computable by size-s de Morgan formulas. Let $\mathcal{G}$ denote some class of Boolean functions; then, we denote by FORMULA$[s] \circ \mathcal{G}$ the class of functions computable by some size-s de Morgan formula where its leaves are labelled by functions in $\mathcal{G}$.

2.3 Approximating polynomials

▶ **Definition 9** (Point-wise approximation)**.** *For a Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$, we say that the function $\tilde{f}\colon \{-1,1\}^n \to \mathbb{R}$ ε-approximates f if for every $z \in \{-1,1\}^n$,*

$$\left|f(z) - \tilde{f}(z)\right| \leq \varepsilon.$$

We will need the following powerful result for the approximating degree of de Morgan formulas.

▶ **Theorem 10** ([52], see also [11])**.** *Let $s > 0$ be an integer and $0 < \varepsilon < 1$. Any de Morgan formula $F\colon \{-1,1\}^n \to \{-1,1\}$ of size s has a ε-approximating polynomial of degree $d = O(\sqrt{s} \cdot \log(1/\varepsilon))$. That is, there exists a degree-d polynomial $p\colon \{-1,1\}^n \to \mathbb{R}$ over the reals such that for every $z \in \{-1,1\}^n$,*

$$|p(z) - F(z)| \leq \varepsilon.$$

Note that Theorem 10 still holds if we use $\{0,1\}$ as the Boolean basis.

2.4 Communication complexity

We use standard definitions from communication complexity. In this paper we consider the standard two party model of Yao and its generalizations to multiparty setting. We denote deterministic communication complexity of a Boolean function by $D(f)$ in the two party setting. We refer to [39] for standard definitions from communication complexity.

▶ **Definition 11.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. The* communication matrix of f*, namely M_f, is a $2^{n/2} \times 2^{n/2}$ matrix defined by $(M_f)_{x,y} := f(x,y)$.*

▶ **Definition 12.** *A* rectangle *is a set of the form $A \times B$, for $A, B \subseteq \{0,1\}^n$. A* monochromatic rectangle *is a rectangle S such that for all pairs $(x,y) \in S$ the value $f(x,y)$ is the same.*

▶ **Lemma 13.** *Let Π be a protocol that computes $f\colon \{0,1\}^n \to \{0,1\}$ with at most D bits of communication. Then, Π induces a partition of M_f into at most 2^D monochromatic rectangles.*

Given a protocol, its *transcript* is the sequence of bits communicated.

▶ **Lemma 14.** *For every transcript z of some communication protocol, the set of inputs (x,y) that generate z is a rectangle.*

Below, we recount the definitions of two multiparty communication models used in this work, namely the number-on-forehead and the number-in-hand models.

▶ **Definition 15** ("Number-on-forehead" communication model; informal)**.** *In the k-party "number-on-forehead" communication model, there are k players and k strings $x_1, \dots, x_k \in \{0,1\}^{n/k}$ and player i gets all the strings except for x_i. The players are interested in computing a value $f(x_1, \dots, x_k)$, where $f\colon \{0,1\}^n \to \{0,1\}$ is some fixed function. We denote by $D^{(k)}(f)$ the number of bits that must be exchanged by the best possible number on forehead protocol solving f.*

We also use the following weaker communication model.

▶ **Definition 16** ("Number-in-hand" communication model; informal)**.** *In the k-party "number-in-hand" communication model, there are k players and k strings $x_1, \dots, x_k \in \{0,1\}^{n/k}$ and player i gets only x_i. The players are interested in computing a value $f(x_1, \dots, x_k)$, where $f\colon \{0,1\}^n \to \{0,1\}$ is some fixed function. We denote by $D^{(k\text{-}\mathsf{NIH})}(f)$ the number of bits that must be exchanged by the best possible communication protocol.*

Note that $D^{(k\text{-}\mathsf{NIH})}(f) \le (1 - 1/k) \cdot n + 1$, for any n-variate Boolean function f, as if $k-1$ players write on the blackboard their string, then the player that did not reveal her input may compute $f(x_1, \dots, x_k)$ on her own and then publish it.

For the communication models mentioned above, there are also bounded-error randomized versions, denoted by R_δ, $R_\delta^{(k)}$, and $R_\delta^{(k\text{-}\mathsf{NIH})}$, respectively, where $0 < \delta < 1$ is an upper bound on the error probability of the protocol. In this setting, the players have access to some shared random string, say r, and the aforementioned error probability of the protocol is considered over the possible choices of r. Moreover, we require the error to be at most δ on each fixed choice of inputs.

We can extend the definitions of the communication complexity measures, defined above, to classes of Boolean functions, in a natural way. That is, for any communication complexity measure $M \in \left\{D, D^{(k)}, D^{(k\text{-}\mathsf{NIH})}, R_\delta, R_\delta^{(k)}, R_\delta^{(k\text{-}\mathsf{NIH})}\right\}$ and for any class of Boolean functions $\mathcal{G}$, we may define

$$M(\mathcal{G}) := \max_{g \in \mathcal{G}} M(g)\,.$$

We note that throughout this paper, we denote by n the number of input bits for the function regardless the communication models. In the k-party communication setting (either NOF or NIH), we assume without loss of generality that n is divisible by k.

2.5 Pseudorandomness

A PRG against a class of functions $\mathfrak{C}$ is a deterministic procedure G mapping short Boolean strings (seeds) to longer Boolean strings, so that G's output "looks random" to every function in $\mathfrak{C}$.

▶ **Definition 17** (Pseudorandom generators)**.** *Let $G\colon \{-1,1\}^\ell \to \{-1,1\}^n$ be a function, $\mathfrak{C}$ be a class of Boolean functions, and $0 < \varepsilon < 1$. We say that G is a* pseudorandom generator of seed length ℓ that ε-fools $\mathfrak{C}$ *if, for every function $f \in \mathfrak{C}$, it is the case that*

$$\left| \mathop{\boldsymbol{E}}_{z \sim \{-1,1\}^\ell}[f(G(z))] - \mathop{\boldsymbol{E}}_{x \sim \{-1,1\}^n}[f(x)] \right| \le \varepsilon.$$

A PRG G outputting n bits is called *explicit* if G can be computed in $\mathsf{poly}(n)$ time. All PRGs stated in this paper are explicit.

2.6 Learning

For a function $f : \{0,1\}^n \to \{0,1\}$ and a distribution $\mathcal{D}$ supported over $\{0,1\}^n$, we denote by $\mathrm{EX}(f, \mathcal{D})$ a randomized oracle that outputs independent identically distributed labelled examples of the form $(x, f(x))$, where $x \sim \mathcal{D}$.

▶ **Definition 18** (PAC learning model [63]). *Let $\mathcal{C}$ be a class of Boolean functions. We say that a randomized algorithm A* learns *$\mathcal{C}$ if, when A is given oracle access to $\mathrm{EX}(f, \mathcal{D})$ and inputs 1^n, ε, and δ, the following holds. For every n-variate function $f \in \mathcal{C}$, distribution $\mathcal{D}$ supported over $\{0,1\}^n$, and real-valued parameters $\varepsilon > 0$ and $\delta > 0$, $A^{\mathrm{EX}(f,\mathcal{D})}(1^n, \varepsilon, \delta)$ outputs with probability at least $1 - \delta$ over its internal randomness and the randomness of the example oracle $\mathrm{EX}(f, \mathcal{D})$ a description of a hypothesis $h : \{0,1\}^n \to \{0,1\}$ such that*

$$\Pr_{x \sim \mathcal{D}}[f(x) = h(x)] \;\geq\; 1 - \varepsilon.$$

The sample complexity *of a learning algorithm is the maximum number of random examples from $\mathrm{EX}(f, \mathcal{D})$ requested during its execution.*

3 Lower bounds

In this section, we prove an average-case lower bound for the generalized inner product function against $\mathsf{FORMULA} \circ \mathcal{G}$, where $\mathcal{G}$ is the set of functions that have low-cost randomized communication protocols in the number-on-forehead setting. This corresponds to Item 1 of Theorem 1. Items 2 and 3 rely on our PRG constructions, and the proofs are deferred to Section 4.

▶ **Theorem 19.** *For any integer $k \geq 2$, $s > 0$ and any class of functions $\mathcal{G}$, let $C \colon \{-1,1\}^n \to \{-1,1\}$ be a function in $\mathsf{FORMULA}[s] \circ \mathcal{G}$ such that*

$$\Pr_{x \sim \{-1,1\}^n}\left[C(x) = \mathsf{GIP}_n^k(x)\right] \geq 1/2 + \varepsilon.$$

Then

$$s = \Omega\left(\frac{n^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right)^2 \cdot \log^2(1/\varepsilon)}\right).$$

We need a couple useful lemmas from [60], whose proofs are presented in Appendix A.1 (Lemma 50 and Lemma 51) for completeness.

▶ **Lemma 20** ([60]). *Let $\mathcal{D}$ be a distribution over $\{-1,1\}^n$, and let $f, C \colon \{-1,1\}^n \to \{-1,1\}$ be such that*

$$\Pr_{x \sim \mathcal{D}}[C(x) = f(x)] \geq 1/2 + \varepsilon.$$

Let $\tilde{C} \colon \{-1,1\}^n \to \mathbb{R}$ be a ε-approximating function of C, i.e., for every $x \in \{-1,1\}^n$, $|C(x) - \tilde{C}(x)| \leq \varepsilon$. Then,

$$\mathop{\boldsymbol{E}}_{x \sim \mathcal{D}}[\tilde{C}(x) \cdot f(x)] \geq \varepsilon.$$

▶ **Lemma 21** ([60]). *Let $\mathcal{D}$ be a distribution over $\{-1,1\}^n$ and let $\mathcal{G}$ be a class of functions. For $f\colon \{-1,1\}^n \to \{-1,1\}$, suppose that $D\colon \{-1,1\}^n \to \{-1,1\} \in \mathsf{FORMULA}[s] \circ \mathcal{G}$ is such that*

$$\Pr_{x\sim\mathcal{D}}[D(x) = f(x)] \geq 1/2 + \varepsilon_0.$$

Then there exists some $h\colon \{-1,1\}^n \to \{-1,1\} \in \mathsf{XOR}_{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)} \circ \mathcal{G}$ such that

$$\mathop{\boldsymbol{E}}_{x\sim\mathcal{D}}[h(x)\cdot f(x)] \geq \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}}.$$

We also need the following communication-complexity lower bound for GIP.

▶ **Theorem 22** ([7, Theorem 2]). *For any $k \geq 2$, any function that computes GIP_n^k on more than $1/2+\delta$ fraction of the inputs (over uniformly random inputs) must have k-party deterministic communication complexity at least $\Omega\big(n/(k\cdot 4^k) - \log(1/\delta)\big)$.*

We first show that if a function correlates with some small formula, whose leaves are functions with low *randomized* communication complexity, then it also correlates non-trivially with some function of relatively low *deterministic* communication complexity.

▶ **Lemma 23.** *For any distribution $\mathcal{D}$ over $\{-1,1\}^n$, and any class of functions $\mathcal{G}$, let $f\colon \{-1,1\}^n \to \{-1,1\}$ and $C\colon \{-1,1\}^n \to \{-1,1\} \in \mathsf{FORMULA}[s] \circ \mathcal{G}$ be such that*

$$\Pr_{x\sim\mathcal{D}}[C(x) = f(x)] \geq 1/2 + \varepsilon.$$

Then there exists a function h, with k-party deterministic communication complexity at most

$$O\left(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G})\cdot\sqrt{s}\cdot\log(1/\varepsilon)\right),$$

such that

$$\Pr_{x\sim\mathcal{D}}[h(x) = f(x)] \geq 1/2 + 1/s^{O(\sqrt{s}\cdot\log(1/\varepsilon))}.$$

Proof. Let $C = F(g_1, g_2\ldots, g_s)$ be the function in $\mathsf{FORMULA}[s]\circ\mathcal{G}$, where F is a formula and $g_1, g_2, \ldots, g_s$ are leaf functions from the class $\mathcal{G}$. For each g_i, consider a k-party randomized protocol Π_i of cost at most $R = R^{(k)}_{\varepsilon/(2s)}(\mathcal{G})$ that has an error $\varepsilon/(2s)$. Now consider the following function

$$\tilde{C}(x) := \mathop{\mathbf{E}}_{\Pi_1,\Pi_2,\ldots,\Pi_s}[D(x)],$$

where

$$D(x) := F(\Pi_1(x), \Pi_2(x), \ldots, \Pi_s(x)).$$

Note that for any fixed choice of $(\Pi_1, \Pi_2, \ldots, \Pi_s)$, D is a formula whose leaves are functions with *deterministic* communication complexity at most R. Next, we show the following.

▷ Claim 24. The function $\tilde{C}$ ε-approximates C.

Proof. First note that since each Π_i is a $(\varepsilon/(2s))$-error randomized protocol, by taking the union bound over the s leaf functions, we have that for every input $x \in \{-1,1\}^n$,

$$\Pr_{\Pi_1,\Pi_2,\ldots,\Pi_s}[\Pi_1(x) = g_1(x) \wedge \Pi_2(x) = g_2(x) \wedge \cdots \wedge \Pi_s(x) = g_s(x)] \geq 1 - \varepsilon/2.$$

Denote by $\mathcal{E}$ the event $\Pi_1(x) = g_1(x) \wedge \Pi_2(x) = g_2(x) \wedge \cdots \wedge \Pi_s(x) = g_s(x)$. We have for every $x \in \{-1,1\}^n$,

$$\begin{aligned}\tilde{C}(x) &= \mathop{\mathbf{E}}_{\Pi_1,\Pi_2,\ldots,\Pi_s}[D(x)]\\ &= \mathbf{E}\left[D(x) \mid \mathcal{E}\right] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}\left[D(x) \mid \neg\mathcal{E}\right] \cdot \mathbf{Pr}[\neg\mathcal{E}]\\ &= C(x) \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}\left[D(x) \mid \neg\mathcal{E}\right] \cdot \mathbf{Pr}[\neg\mathcal{E}].\end{aligned}$$

On the one hand, we have

$$\tilde{C}(x) = C(x) \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}\left[D(x) \mid \neg\mathcal{E}\right] \cdot \mathbf{Pr}[\neg\mathcal{E}] \leq C(x) + \varepsilon/2.$$

On the other hand, we get

$$\tilde{C}(x) = C(x) \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}\left[D(x) \mid \neg\mathcal{E}\right] \cdot \mathbf{Pr}[\neg\mathcal{E}] \geq C(x) \cdot (1-\varepsilon/2) + (-1) \cdot (\varepsilon/2) \geq C(x) - \varepsilon.$$

This completes the proof of the claim. $\lhd$

Now by Claim 24 and Lemma 20, we have

$$\mathop{\mathbf{E}}_{x \sim \mathcal{D}}[\tilde{C}(x) \cdot f(x)] \geq \varepsilon. \tag{1}$$

By the definition of $\tilde{C}$, Equation (1) implies that there exists some D, which is a formula whose leaves are functions with *deterministic* communication complexity at most R, such that

$$\mathop{\mathbf{E}}_{x \sim \mathcal{D}}[D(x) \cdot f(x)] \geq \varepsilon,$$

which implies

$$\mathop{\mathbf{Pr}}_{x \sim \mathcal{D}}[D(x) = f(x)] \geq 1/2 + \varepsilon/2.$$

Then by Lemma 21, there exists a function h, which can be expressed as the XOR of at most $O(\sqrt{s} \cdot \log(1/\varepsilon))$ leaf functions in D, such that

$$\mathop{\mathbf{E}}_{x \sim \mathcal{D}}[h(x) \cdot f(x)] \geq \frac{1}{s^{O(\sqrt{s} \cdot \log(1/\varepsilon))}},$$

which again implies

$$\mathop{\mathbf{Pr}}_{x \sim \mathcal{D}}[h(x) = f(x)] \geq \frac{1}{2} + \frac{1}{s^{O\left(\sqrt{s} \cdot \log(1/\varepsilon)\right)}}.$$

Finally, note that the k-party deterministic communication complexity of h is at most

$$O(R \cdot \sqrt{s} \cdot \log(1/\varepsilon)),$$

where $R = R^{(k)}_{\varepsilon/(2s)}(\mathcal{G})$. ◀

We are now ready to show Theorem 19.

Proof of Theorem 19. Consider Lemma 23 with f being GIP^k_n and $\mathcal{D}$ being the uniform distribution. Consider Theorem 22 with $\delta = 1/s^{O(\sqrt{s} \cdot \log(1/\varepsilon))}$. We have

$$O\left(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G}) \cdot \sqrt{s} \cdot \log(1/\varepsilon)\right) \geq n/(k4^k) - O\left(\sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)\right),$$

which implies

$$s \geq \Omega\left(\frac{n^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right)^2 \cdot \log^2(1/\varepsilon)}\right). \quad ◀$$

4 Pseudorandom generators

Some of our PRGs are obtained from a general framework that allows us to reduce the task of fooling $\mathsf{FORMULA} \circ \mathcal{G}$ to the task of fooling the class of functions which are the parity or conjunction of few functions from $\mathcal{G}$.

4.1 The general framework

We show that in order to get a PRG for the class of subquadratic-size formulas with leaf gates in $\mathcal{G}$, it suffices to get a PRG for very simple sublinear-size formulas: either $\mathsf{XOR} \circ \mathcal{G}$ or $\mathsf{AND} \circ \mathcal{G}$.

▶ **Theorem 25** (PRG for $\mathsf{FORMULA} \circ \mathcal{G}$ from PRG for $\mathsf{XOR} \circ \mathcal{G}$ or $\mathsf{AND} \circ \mathcal{G}$)**.** *Let $\mathcal{G}$ be a class of gates on n bits. For any integer $s > 0$ and any $0 < \varepsilon < 1$, there exists a constant $c > 0$ such that the following holds. If a distribution $\mathcal{D}$ over $\{-1,1\}^n$ $\left(2^{-c \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)}\right)$-fools the XOR (parity) or the AND (conjunction) of $c \cdot \sqrt{s} \cdot \log(1/\varepsilon)$ arbitrary functions from $\mathcal{G}$, then $\mathcal{D}$ also ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$.*

Proof. We first show the case where $\mathcal{D}$ fools the parity of a few functions from $\mathcal{G}$. The proof can be easily adapted to the case of conjunction.

Let $C = F(g_1, g_2 \dots, g_s)$ be a function in $\mathsf{FORMULA}[s] \circ \mathcal{G}$, where F is a formula, and $g_1, g_2, \dots, g_s$ are functions from the class $\mathcal{G}$. Let U be the uniform distribution over $\{-1,1\}^n$. We need to show

$$\mathbf{E}[C(\mathcal{D})] \overset{\varepsilon}{\approx} \mathbf{E}[C(U)]. \tag{2}$$

Let p be a $(\varepsilon/3)$-approximating polynomial for F given by Theorem 10. Note that the degree of p is

$$d = O(\sqrt{s} \cdot \log(1/\varepsilon)).$$

Let us replace F, the formula part of C, with p and let

$$\tilde{C} := p(g_1, g_2 \dots, g_s).$$

Since $\tilde{C}$ point-wisely approximates C, we have

$$\mathbf{E}[\tilde{C}(U)] \overset{\varepsilon/3}{\approx} \mathbf{E}[C(U)],$$

and

$$\mathbf{E}[\tilde{C}(\mathcal{D})] \overset{\varepsilon/3}{\approx} \mathbf{E}[C(\mathcal{D})].$$

Then to show Equation (2), it suffices to show

$$\mathbf{E}[\tilde{C}(\mathcal{D})] \overset{\varepsilon/3}{\approx} \mathbf{E}[\tilde{C}(U)].$$

We have

$$\mathop{\mathbf{E}}_{x \sim \mathcal{D}}[\tilde{C}(x)] = \mathop{\mathbf{E}}_{x \sim D}\left[\sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \prod_{i \in S} g_i(x)\right]$$

$$= \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \mathop{\mathbf{E}}_{x \sim \mathcal{D}} \left[\prod_{i \in S} g_i(x) \right]. \tag{3}$$

Now note that for each $S \subseteq [s]$, $\prod_{i \in S} g_i(x)$ computes the XOR of at most d functions from $\mathcal{G}$. Using the fact the distribution $\mathcal{D}$ $\left(\delta = 1/2^{c \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)}\right)$-fools the XOR of any d functions from $\mathcal{G}$, we get

$$\begin{aligned}
\mathop{\mathbf{E}}_{x \sim \mathcal{D}}[\tilde{C}(x)] &= \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \mathop{\mathbf{E}}_{x \sim D} \left[\prod_{i \in S} g_i(x) \right] \\
&= \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \left(\mathop{\mathbf{E}}_{x \sim U} \left[\prod_{i \in S} g_i(x) \right] + \delta_S \right) && \text{(where } |\delta_S| \le \delta) \\
&= \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \left(\hat{p}(S) \cdot \mathop{\mathbf{E}}_{x \sim U} \left[\prod_{i \in S} g_i(x) \right] + \hat{p}(S) \cdot \delta_S \right) \\
&= \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \mathop{\mathbf{E}}_{x \sim U} \left[\prod_{i \in S} g_i(x) \right] + \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \delta_S \\
&= \mathop{\mathbf{E}}_{x \sim U}[\tilde{C}(x)] + \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \delta_S.
\end{aligned}$$

It remains to show

$$\left| \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \delta_S \right| \le \varepsilon/3.$$

Note that because $p(z) \in [1 - \varepsilon/3, 1 + \varepsilon/3]$ for every $z \in \{-1,1\}^s$, we have

$$|\hat{p}(S)| = \left| \mathop{\mathbf{E}}_{z \sim \{-1,1\}^s} \left[p(z) \cdot \prod_{i \in S} z_i \right] \right| \le 1 + \varepsilon/3 < 2.$$

Then,

$$\left| \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} \hat{p}(S) \cdot \delta_S \right| \le \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} |\hat{p}(S)| \cdot |\delta_S| \le \delta \cdot \sum_{\substack{S \subseteq [s]:\\ |S| \le d}} |\hat{p}(S)| \le \delta \cdot s^{O(\sqrt{s} \cdot \log(1/\varepsilon))} \le \varepsilon/3,$$

where the last inequality holds for some sufficiently large constant c.

To show the case of conjunction, we can write the approximating polynomial as the sum of all degree-d monomials, each of which is the AND of at most d variables. One way to do this is to use the domain $\{0,1\}$ instead of $\{-1,1\}$ in the above argument. We need to show that the coefficients in this case still have small magnitude.

▷ Claim 26. Let $p\colon \{-1,1\}^n \to \mathbb{R}$ be a degree-d polynomial of the form

$$p(x) = \sum_{\substack{S\subseteq[n]:\\|S|\le d}} \hat{p}(S)\cdot\prod_{i\in S} x_i,$$

and let $q\colon \{0,1\}^n \to \mathbb{R}$ be the corresponding polynomial of p over the domain $\{0,1\}^n$, of the form

$$q(y) = \sum_{\substack{T\subseteq[n]:\\|T|\le d}} \hat{q}(T)\cdot\prod_{i\in T} y_i.$$

Then,

$$|q|_1 = \sum_{\substack{T\subseteq[n]:\\|T|\le d}} |\hat{q}(T)| \le n^{O(d)}\cdot \max_{\substack{S\subseteq[n]:\\|S|\le d}} |\hat{p}(S)|.$$

Proof. We have

$$\begin{aligned} q(y_1,y_2,\ldots,y_n) &= p(1-2y_1,1-2y_2,\ldots,1-2y_n)\\ &= \sum_{\substack{S\subseteq[n]:\\|S|\le d}} \hat{p}(S)\cdot\prod_{i\in S}(1-2y_i)\\ &= \sum_{\substack{S\subseteq[n]:\\|S|\le d}} \hat{p}(S)\cdot\left(\sum_{\ell\in\{0,1\}^{|S|}}\prod_{\substack{j\in S:\\ \ell_j=1}} -2y_j\right)\\ &= \sum_{\substack{S\subseteq[n]:\\|S|\le d}}\sum_{\ell\in\{0,1\}^{|S|}} \hat{p}(S)\cdot(-2)^{|\ell|}\cdot\prod_{\substack{j\in S:\\ \ell_j=1}} y_j. \qquad \text{(where } |\ell| = \textstyle\sum_{i=1}^{|S|}\ell_i) \end{aligned}$$

For a pair (S,ℓ) where $S\subseteq[n]$, $|S|\le d$ and $\ell\in\{0,1\}^{|S|}$, let us define the polynomial $q_{(S,\ell)}$ as

$$q_{(S,\ell)}(y) = \hat{p}(S)\cdot(-2)^{|\ell|}\cdot\prod_{\substack{j\in S:\\ \ell_j=1}} y_j.$$

Note that there are at most $n^d\cdot 2^d$ many pairs of such (S,ℓ)'s and for each (S,ℓ), we have

$$|q_{(S,\ell)}|_1 = \left|\hat{p}(S)\cdot(-2)^{|\ell|}\right| \le 2^d\cdot|\hat{p}(S)|.$$

Finally we have

$$|q|_1 = \left|\sum_{(S,\ell)} q_{(S,\ell)}\right|_1 \le \sum_{(S,\ell)} |q_{(S,\ell)}|_1 \le n^d\cdot 2^d\cdot 2^d\cdot \max_{\substack{S\subseteq[n]:\\|S|\le d}}|\hat{p}(S)|,$$

as desired. ◁

This completes the proof of Theorem 25. ◀

4.2 Formulas of low-communication functions in the number-in-hand setting

In this subsection, we will use $\{0, 1\}$ as the Boolean basis.

▶ **Theorem 27.** *For any integers $k \geq 2$, $s > 0$ and any $0 < \varepsilon < 1$, let $\mathcal{G}$ be the class of functions that have k-party number-in-hand $(\varepsilon/6s)$-error randomized communication protocols of cost at most R. There exists a PRG that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$ with seed length*

$$n/k + O\left(\sqrt{s} \cdot (R + \log(s)) \cdot \log(1/\varepsilon) + \log(k)\right) \cdot \log(k).$$

We need the following PRG that fools single functions with low communication complexity in the number-in-hand model. The proof is presented in Appendix A.2 (Theorem 52) for completeness.

▶ **Theorem 28** ([6, 31]). *For any $k \geq 2$, there exists a PRG that δ-fools any n-bits functions with k-party number-in-hand deterministic communication complexity of at most D', with seed length*

$$n/k + O\left(D' + \log(1/\delta) + \log(k)\right) \cdot \log(k).$$

Next, we show a PRG for $\mathsf{FORMULA} \circ \mathcal{G}$, where $\mathcal{G}$ is the class of functions with low-cost communication protocols in the number-in-hand setting. We first show for the case of deterministic protocols.

▶ **Theorem 29.** *For any integers $k \geq 2$ and $s > 0$, let $\mathcal{G}$ be the class of functions whose k-party number-in-hand deterministic communication complexity are at most D. There is a PRG that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$ of size s with seed length*

$$n/k + O\left(\sqrt{s} \cdot \log(1/\varepsilon) \cdot (D + \log(s)) + \log(k)\right) \cdot \log(k).$$

Proof. By Theorem 25, it suffices to show a PRG that $\left(\delta = 1/2^{c \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)}\right)$-fools every function that is the XOR of $t = c \cdot \sqrt{s} \cdot \log(1/\varepsilon)$ arbitrary functions from $\mathcal{G}$. Note that such a function has deterministic communication complexity at most $D' = t \cdot D$. Then Theorem 29 follows from Theorem 28. ◀

We now establish the randomized case.

Proof of Theorem 27. Let C be a function in $\mathsf{FORMULA}[s] \circ \mathcal{G}$. For each of the leaf functions in C, consider a k-party number-in-hand randomized protocol of cost at most R that has an error at most $\varepsilon/(6s)$. By taking a union bound over the s leaf functions and by viewing a randomized protocol as a distribution of deterministic protocols (as shown in the proof of Claim 24), we get the following which is a (point-wisely) $(\varepsilon/3)$-approximating function for C:

$$\tilde{C}(x) := \sum_i p_i \cdot D_i(x),$$

where each $p_i \in [0, 1]$ is some probability density value (so $\sum_i p_i = 1$), and each D_i is a formula whose leaves are functions with *deterministic* communication complexity at most R. Then to ε-fool C, it suffices to $(\varepsilon/3)$-fool its $(\varepsilon/3)$-approximating function $\tilde{C}$. Also, since $\tilde{C}$ is a convex combination of the D_i's, it suffices to $(\varepsilon/3)$-fools all the D_i's. We will do this using the PRG form Theorem 29. We get that there exists a PRG that $(\varepsilon/3)$-fools each D_i with seed length

$$n/k + O\left(\sqrt{s} \cdot (R + \log(s)) \cdot \log(1/\varepsilon) + \log(k)\right) \cdot \log(k),$$

as desired. ◀

4.3 Applications: Fooling formulas of SYMs, LTFs, XORs, and AC^0 circuits

4.3.1 FORMULA ∘ SYM and FORMULA ∘ LTF

Here, we show how the PRG in Theorem 27 implies PRGs for FORMULA∘LTF and FORMULA∘SYM.

▶ **Theorem 30.** *For any size $s > 0$ and $0 < \varepsilon < 1$, there exists a PRG that ε-fools FORMULA$[s]$ ∘ LTF with seed length*

$$O\left(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)\right).$$

For FORMULA$[s]$ ∘ SYM, the seed length is

$$O\left(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(1/\varepsilon)\right).$$

We need the fact that the class of LTF has low communication complexity in the number-in-hand model. Consider the following k-party SUM-GREATER$_m$ problem where the i-th party holds a m-bit number z_i in hand and they want to determine whether $\sum_{i=1}^{k} z_i > \theta$, where θ is a fixed number known to all the parties. Nisan [43] gave an efficient randomized protocol (with public randomness) for this problem.[7]

▶ **Theorem 31** ([43]). *Let $m > 0$ be an integer. For any integer $2 \le k \le m^{O(1)}$, and any $0 < \delta < 1$, there exists a δ-error randomized protocol of cost $O(k \cdot \log(m) \cdot \log(m/\delta))$ for the k-party* SUM-GREATER$_m$ *problem.*

By Theorem 31 and the fact that every linear threshold function on n bits has a representation such that the weights are $O(n \log(n))$ integers [41], we get the following.

▶ **Corollary 32.** *For every $k \ge 2$ and $0 < \delta < 1$, the k-party number-in-hand δ-error randomized communication complexity of LTF is $O(k \cdot \log(n) \cdot \log(n/\delta))$.*

Proof of Theorem 30. By Corollary 32 and Theorem 27, for *every* $k \ge 2$ we get a PRG for FORMULA ∘ LTF of seed length

$$n/k + O\left(\sqrt{s} \cdot k \cdot \log(n) \cdot \log(ns/\varepsilon) \cdot \log(1/\varepsilon) + \log(k)\right) \cdot \log(k).$$

By choosing

$$k = \frac{n^{1/2}}{s^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)},$$

the claimed seed length follows from a simple calculation.

For FORMULA ∘ SYM, note that every n-bit symmetric function has a deterministic k-party number-in-hand communication protocol of cost at most $k \cdot \log(n)$. Then the rest can be shown using a similar argument as above (by choosing $k = n^{1/2} / \left(s^{1/4} \cdot \log(n)\right)$). ◀

[7] Viola [64] gave a δ-error randomized protocol for the k-party SUM-GREATER$_m$ problem of cost $O(k \cdot \log(k) \cdot \log(m/\delta))$, which is better than Nisan's protocol when $k = m^{o(1)}$.

4.3.2 FORMULA ∘ XOR

For the case of FORMULA ∘ XOR, we get a PRG with better seed length.

▶ **Theorem 33.** *For any size $s > 0$ and $0 < \varepsilon < 1$, there exists a PRG that ε-fools FORMULA$[s]$ ∘ XOR with seed length*

$$O\left(\sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon) + \log(n)\right).$$

Proof. By Theorem 25, to fool FORMULA$[s] \circ \mathcal{G}$, it suffices to $\left(\delta = 1/2^{O\left(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon)\right)}\right)$-fool the XOR of a few functions from $\mathcal{G}$, where $\mathcal{G}$ in this case is the set of all XOR functions. Note that the XOR of any set of XOR functions simply computes some XOR function. Therefore, we can use small-bias distribution, which fools every XOR function, to fool FORMULA$[s]$ ∘ XOR. Finally, note that there are known constructions for δ-bias distributions that use $O(\log(n/\delta))$ random bits (see e.g. [3]). ◀

Using the "locality" of this PRG for FORMULA ∘ XOR, we get a lower bound for MCSP against subquadratic-size formulas of XORs.

▶ **Theorem 34.** *For every integer $s > 0$, if* MCSP *on N-bit can be computed by some function in FORMULA$[s]$ ∘ XOR, then $s = \tilde{\Omega}(N^2)$.*

Proof sketch. There is a standard construction of δ-bias distributions that is local (see e.g. [3, Construction 3] and [16, Fact 7]) in the following sense: There exists a circuit of size at most $\tilde{O}(\log(n/\delta) \cdot \log(n))$ such that given a seed of length $O(\log(n/\delta))$ and a index $j \in [n]$, outputs the j-th bit of the distribution. Local PRGs imply MCSP lower bounds (see [16, Section 3]). ◀

4.3.3 FORMULA ∘ AC^0

Another application of Theorem 25 is to take $\mathcal{G}$ to be the set all functions that can be computed by small constant-depth circuits (AC^0). Note the state-of-the-art PRG against size-M depth-d AC^0 has a seed length of $\log^{d+O(1)}(Mn) \cdot \log(1/\varepsilon)$ [56]. Below, let $\mathsf{AC}^0_{d,M}$ denote the class of depth-d circuits of size at most M.

▶ **Theorem 35.** *For any size $s, m > 0$ and $0 < \varepsilon < 1$, there exists a PRG that ε-fools FORMULA ∘ $\mathsf{AC}^0_{d,M}$ of size s with seed length*

$$\log^{d+O(1)}(Mn) \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon).$$

Moreover, by inspecting the construction of PRG in [56], it is not difficult to see that the PRG is also local; there exists a circuit of size at most $\lambda = \log^{d+O(1)}(Mn) \cdot \log(1/\varepsilon)$ such that given a seed of length $O\left(\log^{d+O(1)}(Mn) \cdot \log(1/\varepsilon)\right)$ and a index $j \in [n]$, outputs the j-th bit of the PRG. As a result, we get MCSP lower bounds from the this PRG.

▶ **Theorem 36.** *For every $s, d, M \in \mathbb{N}$, if* MCSP *on N-bit can be computed by some function in FORMULA$[s]$ ∘ $\mathsf{AC}^0_{d,M}$, then*

$$s \geq N^2 / \log^{2d+O(1)}(Mn).$$

4.4 Formulas of low number-on-forehead communication leaf gates

In this section, we show a PRG with mild seed length for formulas of functions with low *multi-party number-on-forehead* communication complexity.

▶ **Theorem 37.** *Let $\mathcal{G}$ be a class of n-bits functions. For any size $s > 0$, there exists a PRG that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$, with seed length*

$$n - \frac{n}{O\left(\sqrt{s} \cdot k \cdot 4^k \cdot \left(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G}) + \log(n)\right) \cdot \log(n/\varepsilon)\right)}.$$

The PRG is constructed using the hardness vs. randomness paradigm.

4.4.1 Hardness based PRGs

We show how to construct the PRG using the average-case hardness result for formulas of functions with low multi-party communication complexity (Theorem 19). We start with some notations. For $x \in \{-1,1\}^m$ and an integer k such that k divides m, we consider a partition of x into k equal-sized consecutive blocks and write $x = x^{(1)}, x^{(2)}, \ldots, x^{(k)}$, where $x^{(i)} \in \{-1,1\}^{m/k}$ for each $i \in [k]$.

▶ **Lemma 38.** *For any integers $m, t, k > 0$ such that k divides m, t, let $\mathcal{G}$ be a class of functions on $mt + t$ bits, and let $G\colon \{-1,1\}^{m \times t} \to \{-1,1\}^{mt+t}$ be*

$$G(x_1, x_2, \ldots, x_t) = \Big(x_1^{(i)}, x_2^{(i)}, \ldots, x_t^{(i)}, \mathsf{GIP}^k_m\left(x_{(i-1)\cdot(t/k)+1}\right), \\ \mathsf{GIP}^k_m\left(x_{(i-1)\cdot(t/k)+2}\right), \ldots, \mathsf{GIP}^k_m\left(x_{i\cdot(t/k)+1}\right)\Big)_{i\in[k]},$$

where $x_1, x_2, \ldots, x_t \in \{-1,1\}^m$. Then G is a PRG that $(t \cdot \varepsilon)$-fools $\mathsf{FORMULA} \circ \mathcal{G}$ of size

$$s = \Omega\left(\frac{m^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2m^2)}(\mathcal{G}) + \log m\right)^2 \cdot \log^2(1/\varepsilon)}\right).$$

Proof. The high level idea is as follows. We argue that if there is a $\mathsf{FORMULA} \circ \mathcal{G}$ of the claimed size that breaks the PRG, then there is a $\mathsf{FORMULA} \circ \mathcal{G}'$ of the same size that computes GIP on m bits, where $\mathcal{G}'$ has a k-party communication complexity that is at most that of $\mathcal{G}$ *with respect to the m-bit input*, and hence contradicts the $\mathsf{FORMULA} \circ \mathcal{G}'$ complexity of the generalized inner product function. The resulting formula is obtained by fixing some input bits of the original $\mathsf{FORMULA} \circ \mathcal{G}$ which breaks the PRG.

We use a hybrid argument. First consider the distribution given by G, where we replace each $\mathsf{GIP}(x_j)$ ($j \in [t]$) with a uniformly random bit; let us denote those random bits as U_j for $j \in [t]$ (note that this is just the uniform distribution). Then for each $j \in [t]$, define H_j to be the distribution that we substitute back $\mathsf{GIP}(x_1), \mathsf{GIP}(x_2), \ldots, \mathsf{GIP}(x_j)$ for the corresponding uniform bits in the previous distribution.

For the sake of contradiction, suppose there exists a $\mathsf{FORMULA} \circ \mathcal{G}$ C of size s such that

$$|\mathbf{Pr}[C(H_t) = 1] - \mathbf{Pr}[F(H_0) = 1]| > t \cdot \varepsilon.$$

By the triangle inequality, there exists a $1 \le j \le k$ such that

$$|\mathbf{Pr}[C(H_j) = 1] - \mathbf{Pr}[C(H_{j-1}) = 1]| > \varepsilon.$$

Then by averaging, there exist some fixings of $x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_t$ and $U_{j+1}, \dots, U_t$ to C such that the above inequality still holds. Let us denote by C' the circuit obtained by C after such fixings and assume without loss of generality $(k-1)t/k \le j \le t$. Then we have

$$\left|\mathbf{Pr}\left[C'\left(x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(k)}, \mathsf{GIP}(x_j)\right) = 1\right] - \mathbf{Pr}\left[C'\left(x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(k)}, U_j\right) = 1\right]\right| > \varepsilon. \quad (4)$$

By a standard "unpredictability implies pseudorandomness" argument [66], we can show that there is some circuit C'', obtained from C' by fixing some value for the last bit, that computes the generalized inner product function on m bits with probability greater than $1/2 + \varepsilon$ over uniformly random inputs. Note that the size of C'' is the same as C' (hence also C) , and also C'' can be computed by some $\mathsf{FORMULA} \circ \mathcal{G}'$, where $R_\delta^{(k)}(\mathcal{G}') \le R_\delta^{(k)}(\mathcal{G})$ for every δ. This contradicts the hardness of GIP for such circuits (Theorem 19). ◀

We are now ready to prove Theorem 37.

Proof of Theorem 37. Consider Lemma 38. Let $n = mt + t$, and we have $m = \left(\frac{n}{t} - 1\right)$. Then Lemma 38 gives a PRG that ε-fools $\mathsf{FORMULA} \circ \mathcal{G}$ of size

$$\begin{aligned} s &= \Omega\left(\frac{m^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2m^2)}(\mathcal{G}) + \log m\right)^2 \cdot \log^2(t/\varepsilon)}\right) \\ &\ge \Omega\left(\left(\frac{n}{t}\right)^2 \Big/ \left(k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right)^2 \cdot \log^2(n/\varepsilon)\right)\right), \end{aligned}$$

which yields

$$t \ge \Omega\left(\frac{n}{\sqrt{s} \cdot k \cdot 4^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right) \cdot \log(n/\varepsilon)}\right).$$

Note that the seed length in this case is $n - t$. ◀

4.4.2 MKtP lower bounds

The PRG in Theorem 37 is sufficient to give an MKtP lower bound for formulas of functions with low multi-party communication complexity.

▶ **Theorem 39.** *For any integer $s > 0$ and any class of N-bit function $\mathcal{G}$, if* MKtP *on N-bit can be computed by some function $\mathsf{FORMULA}[s] \circ \mathcal{G}$, then*

$$s = \frac{N^2}{k^2 \cdot 16^k \cdot R^{(k)}_{1/3}(\mathcal{G}) \cdot \mathsf{polylog}(N)}.$$

Proof. Let C be a function in $\mathsf{FORMULA} \circ \mathcal{G}$ of size less than

$$\frac{N^2}{k^2 \cdot 16^k \cdot R^{(k)}_{1/3}(\mathcal{G}) \cdot \log^c(N)}$$

where $c > 0$ is some sufficiently large constant. By Theorem 37, we have that there is a PRG that $(1/3)$-fools C and its seed length is

$$N - \mathsf{polylog}(N).$$

Also, since the PRG is polynomial-time computable, we get that for every seed, the output of the PRG has Kt complexity at most $\theta = N - \mathsf{polylog}(N)$. However, consider the MKtP function with a threshold parameter θ; this function is not fooled by such a PRG, since it accepts every output of the PRG and rejects a uniformly random string with high probability. ◀

5 Satisfiability algorithms

In this section, we will use $\{0, 1\}$ as the Boolean basis.

5.1 Computational efficient communication protocols

▶ **Definition 40** (Computational efficient communication protocols). *Let $t \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We say that a two-party communication protocol is* t-efficient *if for each of the parties, given an input x and some previously sent messages $\pi \in \{0, 1\}^*$, the next message to send can be computed in time $t(|x|, |\pi|)$ ($\perp$ is being output if there is no next message). We say that such a protocol is* explicit *if $t(|x|, |\pi|) = 2^{o(|x|+|\pi|)}$.*

▶ **Lemma 41.** *Let $f \colon \{0, 1\}^n \to 1$ and let Π be a t-efficient communication protocol for f with communication cost at most D. Then the protocol tree of Π can be output in time $O\left(D \cdot t(n/2, D) \cdot 2^{n/2} \cdot 2^D\right)$. That is, there exists an algorithm that outputs a list of all (partial and full) transcripts of length at most D and the rectangles associated with each of the transcripts.*

Proof. It suffices to show that, given an input $x \in \{0, 1\}^{n/2}$ and a transcript $\ell \in \{0, 1\}^{\leq D}$, we can decide whether x belongs to the rectangle indexed by ℓ in time $D \cdot t(n/2, D)$. Suppose x is the input for Alice (resp. Bob), and we want to decide whether x belongs to the rectangle indexed by π. We can carry out the communication task by simulating the behavior of Alice (resp. Bob) using the protocol Π and simulating Bob's (resp. Alice's) behavior using the transcript π, and check whether the messages sent by Alice (resp. Bob) is consistent with the transcript π. This takes time at most $D \cdot t(n/2, D)$. To construct the tree, we do the above for every (partial and full) transcript $\pi \in \{0, 1\}^{\leq D}$ and every input $x \in \{0, 1\}^{n/2}$ for Alice (resp. Bob). The total running time is $O\left(D \cdot t(n/2, D) \cdot 2^{n/2} \cdot 2^D\right)$. ◀

For a protocol Π, we denote by Leaves(Π) the set of full transcripts of Π.

▶ Remark. We note that, in the *white-box* context of the satisfiability problem, there is no need to assume a canonical partition of the input variables among the players. For instance, a helpful partition can either be given as part of the input, or computed by the algorithm. As a consequence, in instantiations of Theorem 5 for a particular circuit class $\mathcal{C}$, it is sufficient to be able to convert the input circuit from $\mathcal{C}$ into some device from $\mathsf{FORMULA} \circ \mathcal{G}$ for which protocols of bounded communication cost can be described.

5.2 Explicit approximating polynomials for formulas

From Theorem 10, we know that every size-s formula has a degree-$O(\sqrt{s})$ polynomial that point-wisely approximates it. In our SAT algorithms, we will need to *explicitly construct* such an approximating polynomial given a formula. One way to do this is to use an *efficient* quantum query algorithm for formulas. It is known that a quantum query algorithm for a function f using at most T queries implies an approximating polynomial for f of degree at most $2T$ [9], and by classically simulating such an quantum algorithm, one can show that the approximating polynomial can be obtained in time that is polynomial in the number of its monomials, in addition to the time for the classical simulation. For our task, we can use the result of Reichardt [51] which showed an *efficient* quantum algorithm for evaluating size-s formulas with $O\left(\sqrt{s}\cdot\log s\right)$ queries.[8] Here, we present an alternate way to construct approximating polynomials for de Morgan formulas which rely only on the *existence* of such polynomials, without requiring an efficient quantum query algorithm. This "black-box" approach was suggested to us by an anonymous reviewer.

We first need the following structural lemma for formulas.

▶ **Lemma 42** ([29, 58]). *For every integer $s > 0$, there exists an algorithm such that given a size-s de Morgan formula F, runs in* $\mathsf{poly}(s)$ *time and outputs a top formula F' with $O(\sqrt{s})$ leaves and each leaf of F' is a sub-formula with $O(\sqrt{s})$ input leaves.*

▶ **Lemma 43.** *For any integer $s > 0$ and any $0 < \varepsilon < 1$, there exists an algorithm of running time $s^{O\left(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon)\right)}$ such that given a de Morgan formula F of size s, outputs an ε-approximating polynomial of degree $O(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon))$ for F. That is, the algorithm outputs a multi-linear polynomial (as sum of monomials) over the reals such that for every $x \in \{0,1\}^n$,*

$$|p(x) - F(x)| \leq \varepsilon.$$

Proof. We first note that it suffices to construct a (1/3)-approximating polynomial for F with degree $D = O(\sqrt{s}\cdot\log(s))$. This is because given a (1/3)-approximating polynomial one can obtain explicitly an ε-approximating polynomial of degree $D\cdot O(\log(1/\varepsilon))$, by feeding $O(1/\varepsilon)$ copies of the (1/3)-approximating polynomial to the polynomial computing MAJORITY on $O(1/\varepsilon)$ bits [11] (see also [58, Appendix B]).

We first invoke Lemma 42 on F to obtain a top formula F' with $t = O(\sqrt{s})$ leaves, each of which is a sub-formula of size $O(\sqrt{s})$. We construct a (1/20)-approximating (multi-linear) polynomial P for the top formula F', which has degree $d_1 = O(s^{1/4})$ by Theorem 10. Note that P can be constructed in time $2^{O(\sqrt{s})}$ because F' has at most $O(\sqrt{s})$ leaves. Next, for each of the t sub-formulas, denoted as $F_1, F_2, \ldots, F_t$, we construct a $(1/(20t))$-approximating polynomial. Note that these polynomials have degree $d_1 = O(s^{1/4}\cdot\log(s))$ and can be constructed in time $2^{O(\sqrt{s})}$. Let's denote these t polynomials as $Q_1, Q_2, \ldots, Q_t$. Now for each Q_i $(i \in [t])$, we define

$$q_i(x) = \frac{Q_i(x) + 1/(20t)}{1 + 1/(10t)}.$$

The final approximating polynomial for F is given as

$$p(x) = P\left(q_1(x), q_2(x), \ldots, q_t(x)\right).$$

Note that p has degree $d_1 \cdot d_2 = O(\sqrt{s}\cdot\log(s))$ and can be constructed (as sum of monomials) in time $s^{O(\sqrt{s}\cdot\log(s))}$. It remains to show that p (1/3)-approximates F.

[8] It is also known that there exists a quantum query algorithm for evaluating size-s formulas with $O\left(\sqrt{s}\right)$ queries [52], which implies the existence of an approximating polynomial for size-s formulas of degree $O\left(\sqrt{s}\right)$ (see Theorem 10). However, because this algorithm is not known to be efficient, it is unclear whether such an approximating polynomial can be constructed efficiently with respect to the number of monomials.

For $0 \le q \le 1$, let N_q be the distribution over $\{0,1\}$ such that $\mathbf{Pr}_{y\sim N_q}[y=1] = q$. Then for an fixed input $x \in \{0,1\}^s$, we have

$$p(x) = \mathop{\mathbf{E}}_{y_i \sim N_{q_i(x)}}[P(y_1, y_2, \ldots, y_t)]. \tag{5}$$

Let $\mathcal{E}$ be the event that $y_i = F_i(x)$ for all $i \in [t]$. Note that

$$\delta := \mathop{\mathbf{Pr}}_{y_i \sim N_{q_i(x)}}[\neg\mathcal{E}] \le 1/10. \tag{6}$$

To see Equation (6), note that for every $i \in [t]$, if $F_i(x) = 0$, then $0 \le q_i(x) \le 1/(10t)$, which implies

$$\mathop{\mathbf{Pr}}_{y_i \sim N_{q_i(x)}}[y_i \ne F_i(x)] \le 1/(10t).$$

Similar for the case when $F_i(x) = 1$ (which implies $1 - 1/(10t) < q_i(x) \le 1$). Then Equation (6) follows from a union bound. Now we can re-write Equation (5) as

$$\begin{aligned} p(x) &= \mathbf{E}[P(y_1, y_2, \ldots, y_t) \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[P(y_1, y_2, \ldots, y_t) \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}] \\ &= (F'(F_1(x), F_2(x), \ldots, F_t(x)) \pm 1/20) \cdot (1-\delta) + \mathbf{E}[P(y_1, y_2, \ldots, y_t) \mid \neg\mathcal{E}] \cdot \delta. \end{aligned}$$

Note that $P(y) \in [-1/(20t), 1 + 1/(20t)]$ for every $y \in \{0,1\}^t$, and that $\delta \le 1/10$. A simple calculation shows that

$$p(x) = F'(F_1(x), F_2(x), \ldots, F_t(x)) \pm \frac{1}{3},$$

as desired. ◀

5.3 The #SAT algorithm

In this subsection, we present our #SAT algorithm.

▶ **Theorem 44.** *For any integer $s > 0$, there exists a deterministic #SAT algorithm for FORMULA$[s] \circ \mathcal{G}$, where $\mathcal{G}$ is the class of functions with explicit two-party deterministic protocols of communication cost at most D, that runs in time*

$$2^{n - \frac{n}{\sqrt{s}\cdot\log(s)\cdot(\log(s)+D)}}.$$

In the case $\mathcal{G}$ is the class of functions with explicit randomized protocols of communication cost at most R, there exists an analogous randomized algorithm with a running time

$$2^{n - \left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot R}\right)^{1/2}}.$$

The algorithm is based on the framework for designing satisfiability algorithms developed by Williams [65]. The idea is to transform a given circuit into a "sparse polynomial" and solve satisfiability by evaluating the polynomial on all points in a faster-than-brute-force manner.

We first need the following fast matrix multiplication algorithm for "narrow" matrices.

▶ **Theorem 45** ([17]). *Multiplication of an $N \times N^{.172}$ matrix with an $N^{.172} \times N$ matrix can be done in $O(N^2 \log^2 N)$ arithmetic operations over any field.*

For an even number $n > 0$, and $x \in \{0,1\}^n$, we denote by x^{L} (resp. x^{R}) the first half of x and $x^{\mathrm{R}} \in \{0,1\}^{n/2}$ the second half. We now prove Theorem 44.

Proof of Theorem 44. We first prove the deterministic case.

Let $C = F(g^1, g^2 \ldots, g^s)$ be a device in $\mathsf{FORMULA} \circ \mathcal{G}$ where F is a formula and $g^1, g^2, \ldots, g^s$ are functions that have a explicit communication protocol of cost at most D. The first step is to output the protocol tree for each g^i ($i \in [s]$). Since each g^i has explicit protocol of cost at most D, by Lemma 41, these protocol trees can be output in time $s \cdot 2^{n/2+D+o(n)} \leq 2^{n/1.9}$ (here we assume $D = o(n)$ otherwise the theorem holds trivially).

Let n' be an integer whose value is determined later. Let T be a set of n' variables such that T contains $n'/2$ variables from the first half of the n variables and the rest are from the second half. For a partial assignment $z \in \{0,1\}^{n'}$ to T, denote by C_z the restricted function of C where the variables in T are fixed according to z. To count the number of satisfying assignments of C, we need to compute the following quantity:

$$\sum_{x \in \{0,1\}^{n-n'}} \sum_{z \in \{0,1\}^{n'}} C_z(x). \tag{7}$$

Now consider

$$Q(x) = \sum_{z \in \{0,1\}^{n'}} C_z(x).$$

We will try to obtain the value of $Q(x)$ for every $x \in \{0,1\}^{n-n'}$, in time about $2^{n-n'}$, which will allow us to compute the quantity in Equation (7) in time $O(2^{n-n'})$ by summing $Q(x)$ over all the x's. We do this by first transforming Q into an *approximating* polynomial with not-too-many monomials, and each monomial is a product of *functions that only rely on either the first or the second half of* x. With such a polynomial, we can perform fast multipoint evaluation using the fast matrix multiplication algorithm in Theorem 45.

For each $z \in \{0,1\}^{n'}$, we view the formula C_z as $F(g_z^1, g_z^2, \ldots, g_z^s)$, where F is the de Morgan formula part of C_z and $g_z^1, g_z^2, \ldots, g_z^s$ are the leaf gates. Let us now replace F by a ε-approximating polynomial p, where $\varepsilon = 1/\left(3 \cdot 2^{n'}\right)$, using Lemma 43. Note that the degree of p is at most

$$d \leq O(\sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)) \leq O(\sqrt{s} \cdot \log(s) \cdot n').$$

Now consider the following

$$Q'(x) = \sum_{z \in \{0,1\}^{n'}} p(g_z^1(x), g_z^2(x), \ldots, g_z^s(x)).$$

First, note that by the value that we've chosen for the approximating error ε, we have that, for every x,

$$|Q'(x) - Q(x)| \leq 2^{n'} \cdot \varepsilon = 1/3.$$

In other words, given $Q'(x)$, we can recover the value of $Q(x)$, which is supposed to be an integer.

Next, we perform fast multipoint evaluation on Q'. First of all, we re-write Q' as follows:

$$Q'(x) = \sum_{z \in \{0,1\}^{n'}} \sum_{\substack{S \subseteq [s]: \\ |S| \leq d}} \hat{p}(S) \cdot \prod_{i \in S} g_z^i(x). \tag{8}$$

Now let Π_i be the protocol of g^i, we can re-write g^i_z as follows:

$$g^i_z(x) = \sum_{\pi_i \in \mathrm{Leaves}(\Pi_i)} \alpha^i\left(z^{\mathrm{L}}x^{\mathrm{L}}, \pi_i\right) \cdot \beta^i\left(z^{\mathrm{R}}x^{\mathrm{R}}, \pi_i\right), \tag{9}$$

where $\alpha^i\left(z^{\mathrm{L}}x^{\mathrm{L}}, \pi_i\right)$ (resp. $\beta^i\left(z^{\mathrm{R}}x^{\mathrm{R}}, \pi_i\right)$) is 1 if and only if $\left(z^{\mathrm{L}}x^{\mathrm{L}}\right)$ (resp. $\left(z^{\mathrm{R}}x^{\mathrm{R}}\right)$) belongs to the rectangle indexed by π_i and the function value of that rectangle is 1. Note that for each $i \in [s]$, given the pre-computed protocol tree of the Π_i, α^i and β^i can be computed in polynomial time (for example, using binary search). After plugging Equation (9) into Equation (8) for every $i \in [s]$ and rearranging, we get

$$Q'(x) = \sum_{z \in \{0,1\}^{n'}} \sum_{\substack{S \subseteq [s]: \\ |S| \le d}} \sum_{\substack{\vec{\pi} = (\pi_i)_{i \in S}: \\ \pi_i \in \mathrm{Leaves}(\Pi_i)}} \hat{p}(S) \cdot \prod_{i \in S} \alpha^i\left(z^{\mathrm{L}}x^{\mathrm{L}}, \pi_i\right) \cdot \prod_{i \in S} \beta^i\left(z^{\mathrm{R}}x^{\mathrm{R}}, \pi_i\right). \tag{10}$$

Note that Q' can be expressed as the sum of at most m terms, where

$$m \le 2^{n'} \cdot s^{O(\sqrt{s} \cdot \log(s) \cdot n')} \cdot 2^{O(\sqrt{s} \cdot \log(s) \cdot n' \cdot D)} \le 2^{O(\sqrt{s} \cdot \log(s) \cdot (\log(s) + D) \cdot n')}.$$

Note that given Lemma 43, we can obtain Q' in time

$$2^{O(\sqrt{s} \cdot \log^2(s) \cdot D \cdot n')}. \tag{11}$$

Next, we construct a $2^{(n-n')/2} \times m$ matrix A and a $m \times 2^{(n-n')/2}$ matrix B as follows:

$$A_{x^{\mathrm{L}},(z,S,\vec{\pi})} = \hat{p}(S) \cdot \prod_{i \in S} \alpha^i\left(z^{\mathrm{L}}x^{\mathrm{L}}, \pi_i\right),$$

and

$$B_{(z,S,\vec{\pi}),x^{\mathrm{R}}} = \prod_{i \in S} \beta^i\left(z^{\mathrm{R}}x^{\mathrm{R}}, \pi_i\right).$$

It is easy to see that for each $x \in \{0,1\}^{n-n'}$,

$$Q'(x) = (A \cdot B)_{x^{\mathrm{L}},x^{\mathrm{R}}}.$$

We now want to compute $A \cdot B$. Therefore, we want $m \le 2^{.172(n-n')/2}$ so that computing $A \cdot B$ can be done in time $\tilde{O}(2^{n-n'})$ using Theorem 45. For this we can set n' to be

$$n' = \frac{n}{c \cdot \sqrt{s} \cdot \log(s) \cdot (\log(s) + D)},$$

where $c > 0$ is some sufficiently large constant. Together with the running time in Equation (11), The total running time of the algorithm is therefore

$$2^{n - \frac{n}{\sqrt{s} \cdot \log(s) \cdot (\log(s)+D)}}.$$

For the randomized case, for each g^i ($i \in [s]$), we consider a randomized protocol Π_i that has error $\varepsilon' \le 1/(3 \cdot s \cdot 2^{n'})$, and replace g^i with a randomly picked protocol from Π_i, so we can say that for every $x \in n - n'$, the algorithm computes $Q(x)$ (or $Q'(x)$) with probability at least 2/3 (via a union bound over all the g^i's and a union bound over all the z's in $\{0,1\}^{n'}$). Then we can repeat the above algorithm $\mathsf{poly}(n)$ times and obtain $Q(x)$ for all $x \in \{0,1\}^{n-n'}$ correctly with high probability. Note that the error of any randomized protocol with communication complexity R can be reduced to ε' by blowing up the communication complexity by a factor of $O(\log(1/\varepsilon'))$. In this case the, (as we are considering longer transcripts) the number of terms in Q' (as in Equation (10)) will be

$$2^{O(\sqrt{s}\cdot\log^2(s)\cdot R\cdot(n')^2)},$$

and we need to set accordingly

$$n' = \Omega\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot R}\right)^{1/2},$$

which gives the claimed running time for the randomized case. ◀

In fact, using the ideas above we can also get a randomized #SAT algorithm for the more expressive class $\mathsf{FORMULA}\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$, where $\mathsf{AC}^0_{d,M}$ is the class of depth-d size-M circuits and $\mathcal{G}$ is the class of functions that have low-communication complexity[9], by combining with the fact that AC^0 circuits have low-degree *probabilistic polynomials over the reals* (a probabilistic polynomial of a function f is a distribution on polynomials such that for every input x, a randomly picked polynomial from the distribution agrees with f on the input x). More specifically, we have the following.

▶ **Theorem 46.** *For any integers $s, d, M > 0$, there exists a randomized #SAT algorithm for $\mathsf{FORMULA}[s]\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$, where $\mathcal{G}$ is the class of functions with explicit two-party deterministic protocols of communication cost at most D, the algorithm outputs the number of satisfying assignments in time*

$$2^{n-\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot D}\right)^{1/2}}.$$

In the case $\mathcal{G}$ is the class of functions with explicit randomized protocols of communication cost at most R, there exists an analogous randomized algorithm with a running time

$$2^{n-\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot R}\right)^{1/3}}.$$

Proof sketch. We show the case where $\mathcal{G}$ has low randomized communication complexity. Let

- $\varepsilon_1 = 1/\left(3\cdot 2^{n'}\right)$,
- $\varepsilon_2 = 1/\left(6\cdot s\cdot 2^{n'}\right)$ and
- $\varepsilon_3 = 1/\left(6\cdot M\cdot 2^{n'}\right)$.

As in the proof of Theorem 44, we can replace the formula part of $\mathsf{FORMULA}[s]\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$ with a ε_1-approximating polynomial of degree

$$O(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon_1)) = O(\sqrt{s}\cdot\log(s)\cdot n').$$

Then we replace the $\mathsf{AC}^0_{d,M}$ circuit with a randomly picked polynomial from a ε_2-error probabilistic polynomial. By [26], such a probabilistic polynomial is constructive and has degree at most

$$(\log M)^{O(d)}\cdot\log(1/\varepsilon_2) = (\log M)^{O(d)}\cdot(n'+\log(s)).$$

[9] Here we define the size of a $\mathsf{AC}^0_{d,M}$ circuit to be the number of wires. Note that a circuit in $\mathsf{FORMULA}\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$ can have M functions from $\mathcal{G}$ at the bottom.

Finally, we replace each of the bottom functions, which is from $\mathcal{G}$, with a randomly picked protocol from a randomized protocol with error ε_3, and hence has cost at most

$$R \cdot O(\log(1/\varepsilon_3)) = O(R \cdot (n' + \log(M))).$$

As a result, we can express Q' as a polynomial with at most

$$2^{O\left(\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot R\cdot(n')^3\right)}$$

monomials, whose variables are functions that depend on either the first half or the second half of x. Note that with our choices of ε_2 and ε_3, for every $x \in \{0,1\}^{n-n'}$, the algorithm computes $Q(x)$ correctly that with probability at least $2/3$ (by union bounds). By the same reasoning as in the proof of Theorem 44, we get a randomized #SAT algorithm with running time

$$2^{n-\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot R}\right)^{1/3}},$$

as desired. ◀

It is worth noting that unlike Theorem 44, the algorithm in Theorem 46 is *randomized* even if $\mathcal{G}$ is the class of functions with low *deterministic* communication complexity, because of the use of probabilistic polynomials for the AC^0 circuits.

6 Learning algorithms

In this section, we prove the following learning result for the $\mathsf{FORMULA} \circ \mathsf{XOR}$ model.

▶ **Theorem 47.** *For every constant $\gamma > 0$, there is an algorithm that* PAC *learns the class of n-variate Boolean functions $\mathsf{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ to accuracy ε and with confidence δ in time $\mathsf{poly}\left(2^{n/\log n}, 1/\varepsilon, \log(1/\delta)\right)$.*

We first review some useful results that pertain to agnostically learning parities as well as boosting of learning algorithms.

6.1 Agnostically learning parities and boosting

For a parameter $n \geq 1$, let Δ be a distribution on labelled examples (x, y) supported over $\{0,1\}^n \times \{0,1\}$, and assume that for each x there is at most one y such that $(x,y) \in \mathsf{Support}(\Delta)$. For a function $h\colon \{0,1\}^n \to \{0,1\}$, we denote by $\mathrm{err}_\Delta(h)$ the error of h under this distribution:

$$\mathop{\mathrm{err}}_{\Delta}(h) \;=\; \mathop{\mathbf{Pr}}_{(x,y)\sim\Delta}[h(x) \neq y]\,.$$

Similarly, for a class of functions $\mathcal{C}$, we let $\mathrm{opt}_\Delta(\mathcal{C})$ be the error of the best function in the class:

$$\mathop{\mathrm{opt}}_{\Delta}(\mathcal{C}) \;=\; \min_{h\in\mathcal{C}} \mathop{\mathrm{err}}_{\Delta}(h)\,.$$

We will need a result established by Kalai, Mansour, and Verbin [36], which gives a non-trivial time agnostic learning algorithm for the class of parities.

▶ **Lemma 48** ([36]). *Let* XOR *be the class of parity functions on* n *variables. Then, for any constant* $\zeta > 0$*, there is a randomized learning algorithm* W *such that, for every parameter* $n \geq 1$ *and distribution* Δ *over labelled examples, when* W *is given access to independent samples from* Δ *it outputs with high probability a circuit computing a hypothesis* $h : \{0,1\}^n \to \{0,1\}$ *such that*

$$\underset{\Delta}{\mathrm{err}}(h) \;\leq\; \underset{\Delta}{\mathrm{opt}}(\mathsf{XOR}) + 2^{-n^{1-\zeta}}.$$

The sample complexity and running time of W *is* $2^{O(n/\log n)}$.

Recall that a boosting procedure for learning algorithms transforms a weak learner that outputs a hypothesis that is just weakly correlated with the unknown function into a (strong) PAC learning algorithm for the same class (i.e., a learner in the sense of Definition 18). We refer for instance to [37] for more information about boosting in learning theory. We shall make use of the following boosting result by Freund [22].

▶ **Lemma 49** ([22]). *Let* W *be a (weak) learner for a class* $\mathcal{C}$ *that runs in time* $t(n)$ *and outputs (under any distribution) a hypothesis of error up to* $1/2 - \beta$*, for some constructive function* $\beta(n) > 0$*. Then, there exists a* PAC *learning algorithm for* $\mathcal{C}$ *that runs in time* $\mathsf{poly}(n, t, 1/\varepsilon, 1/\beta, \log(1/\delta))$.

6.2 PAC-learning small formulas of parities

We are ready to show that sub-quadratic size formulas over parity functions can be learned in time $2^{O(n/\log n)}$. First, we argue that Lemma 48 provides a weak learner that works under any distribution $\mathcal{D}$ supported over $\{0,1\}^n$. This will follow from Lemma 21, which shows that any function in $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ is correlated with some parity function with respect to $\mathcal{D}$. We then obtain a standard PAC learner via the boosting procedure from Lemma 49.

Proof of Theorem 47. Let $\mathcal{C} = \mathsf{FORMULA} \circ \mathsf{XOR}$, where $s = n^{2-\gamma}$ for some constant $\gamma > 0$. For any function $f \in \mathsf{FORMULA}[s] \circ \mathsf{XOR}$ and distribution $\mathcal{D}$ supported over $\{0,1\}^n$, Lemma 21 shows that there exists a parity function $\chi = \chi(f, \mathcal{D})$ such that

$$\Pr_{x \sim \mathcal{D}}[f(x) = \chi(x)] \;\geq\; \frac{1}{2} + \frac{1}{2^{n^{1-\lambda}}},$$

for some $\lambda = \lambda(\gamma) > 0$ independent of n, under the assumption that n is sufficiently large. Let $\Delta = \Delta(\mathcal{D}, f)$ be the distribution over labelled examples induced by $\mathcal{D}$ and f. Note that $\mathrm{opt}_\Delta(\mathsf{XOR}) \leq 1/2 - \exp(n^{1-\lambda})$. Consequently, by invoking Lemma 48 with parameter $\zeta = \lambda$, it follows that $\mathsf{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ can be learned under an arbitrary distribution to error $\beta(n) \leq 1/2 - \exp(n^{1-\Omega(1)})$ in time $t(n) = 2^{O(n/\log n)}$. Consequently, we can obtain a PAC learner algorithm for $\mathsf{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ via Lemma 49 that runs in time $\mathsf{poly}(n, t(n), 1/\varepsilon, 1/\beta, \log(1/\delta)) = \mathsf{poly}(2^{n/\log n}, 1/\varepsilon, \log(1/\delta))$. ◀

References

1 Amir Abboud and Karl Bringmann. Tighter connections between formula-SAT and shaving logs. In *ICALP*, pages 8:1–8:18, 2018. `doi:10.4230/LIPIcs.ICALP.2018.8`.

2 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *FOCS*, pages 467–476, 2016. doi:10.1109/FOCS.2016.57.

3 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992. doi:10.1002/rsa.3240030308.

4 Andris Ambainis, Andrew M. Childs, Ben Reichardt, Robert Spalek, and Shengyu Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM J. Comput.*, 39(6):2513–2530, 2010. doi:10.1137/080712167.

5 Alexander E Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Moscow Univ. Math. Bull.*, 42(1):63–66, 1987.

6 Roy Armoni, Michael E. Saks, Avi Wigderson, and Shiyu Zhou. Discrepancy sets and pseudorandom generators for combinatorial rectangles. In *FOCS*, pages 412–421, 1996. doi:10.1109/SFCS.1996.548500.

7 László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992. doi:10.1016/0022-0000(92)90047-M.

8 Swapnam Bajpai, Vaibhav Krishan, Deepanshu Kush, Nutan Limaye, and Srikanth Srinivasan. A #sat algorithm for small constant-depth circuits with PTF gates. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.8.

9 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097.

10 Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In *TCC*, pages 699–729, 2018. doi:10.1007/978-3-030-03810-6_25.

11 Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory Comput. Syst.*, 40(4):379–395, 2007. doi:10.1007/s00224-006-1313-z.

12 Lijie Chen, Ce Jin, and Ryan Williams. Hardness magnification for all sparse NP languages. In *FOCS*, 2019. ECCC:TR19-118.

13 Lijie Chen and Ruosong Wang. Classical algorithms from quantum and Arthur-Merlin communication protocols. In *ITCS*, pages 23:1–23:20, 2019. doi:10.4230/LIPIcs.ITCS.2019.23.

14 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015. doi:10.1007/s00037-015-0100-0.

15 Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. *Theory of Computing*, 14(1):1–55, 2018. doi:10.4086/toc.2018.v014a009.

16 Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. In *ICALP*, pages 39:1–39:14, 2019. ECCC:TR19-022.

17 Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. doi:10.1137/0211037.

18 Evgeny Dantsin and Edward A Hirsch. Worst-case upper bounds. *Handbook of Satisfiability*, 185:403–424, 2009.

19 Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *Computational Complexity*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x.

20 Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum algorithm for the hamiltonian NAND tree. *Theory of Computing*, 4(1):169–190, 2008. doi:10.4086/toc.2008.v004a008.

21 Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013. doi:10.4086/toc.2013.v009a026.

22 Yoav Freund. Boosting a weak learning algorithm by majority. In *COLT*, pages 202–216, 1990. dl.acm.org/citation.cfm?id=92640.

23 Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Computational Complexity*, 27(2):245–304, 2018. `doi:10.1007/s00037-018-0166-6`.

24 Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. Fooling functions of halfspaces under product distributions. In *CCC*, pages 223–234, 2010. `arXiv:1001.1593`.

25 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219, 1996. doi:10.1145/237814.237866.

26 Prahladh Harsha and Srikanth Srinivasan. On polynomial approximations to AC. *Random Struct. Algorithms*, 54(2):289–303, 2019. doi:10.1002/rsa.20786.

27 Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.

28 Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *STOC*, pages 526–535, 2007. doi:10.1145/1250790.1250867.

29 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *FOCS*, pages 111–119, 2012. ECCC:TR12-057.

30 Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993. doi:10.1002/rsa.3240040202.

31 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *STOC*, pages 356–364, 1994. doi:10.1145/195058.195190.

32 Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

33 Valentine Kabanets. Derandomization: A brief overview. *Current Trends in Theoretical Computer Science*, 1:165–188, 2002.

34 Valentine Kabanets and Zhenjian Lu. Satisfiability and derandomization for small polynomial threshold circuits. In *APPROX/RANDOM*, pages 46:1–46:19, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.46.

35 Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008. doi:10.1137/060649057.

36 Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. On agnostic boosting and parity learning. In *STOC*, pages 629–638, 2008. doi:10.1145/1374376.1374466.

37 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. mitpress.mit.edu/books/introduction-computational-learning-theory.

38 Valeriy M Khrapchenko. Method of determining lower bounds for the complexity of π-schemes. *Mathematical Notes*, 10(1):474–479, 1971.

39 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

40 Sophie Laplante, Troy Lee, and Mario Szegedy. The quantum adversary method and classical formula size lower bounds. *Computational Complexity*, 15(2):163–196, 2006. doi:10.1007/s00037-006-0212-7.

41 Saburo Muroga, Iwao Toda, and Satoru Takasu. Theory of majority decision elements. *Journal of the Franklin Institute*, 271:376–418, 1961. doi:10.1016/0016-0032(61)90702-5.

42 E.I. Nechiporuk. On a Boolean function. *Doklady Akademii Nauk SSSR*, 169(4):765–766, 1966. English translation in Soviet Mathematics Doklady.

43 Noam Nisan. The communication complexity of threshold gates. In *Proceedings of "Combinatorics, Paul Erdos is Eighty"*, pages 301–315, 1994.

44 Ryan O'Donnell, Rocco A. Servedio, and Li-Yang Tan. Fooling polytopes. In *STOC*, pages 614–625, 2019. `arXiv:1808.04035`.

45 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *CCC*, pages 27:1–27:29, 2019. doi:10.4230/LIPIcs.CCC.2019.27.

46 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *CCC*, pages 18:1–18:49, 2017. ECCC:TR16-197.

47 Mike Paterson and Uri Zwick. Shrinkage of de Morgan formulae under restriction. *Random Struct. Algorithms*, 4(2):135–150, 1993. doi:10.1002/rsa.3240040203.

48 Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *SODA*, pages 1065–1075, 2010. doi:10.1137/1.9781611973075.86.

49 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Inf.*, 25(5):515–535, 1988. doi:10.1007/BF00279952.

50 Ben Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *FOCS*, pages 544–551, 2009. doi:10.1109/FOCS.2009.55.

51 Ben Reichardt. Faster quantum algorithm for evaluating game trees. In *SODA*, pages 546–559, 2011. arXiv:0907.1623.

52 Ben Reichardt. Reflections for quantum query algorithms. In *SODA*, pages 560–569, 2011. arXiv:1005.1601.

53 Ben Reichardt and Robert Spalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(1):291–319, 2012. doi:10.4086/toc.2012.v008a013.

54 Igor S Sergeev. Upper bounds for the size and the depth of formulae for MOD-functions. *Discrete Mathematics and Applications*, 27(1):15–22, 2017.

55 Rocco A. Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In *ITCS*, pages 30:1–30:21, 2017. doi:10.4230/LIPIcs.ITCS.2017.30.

56 Rocco A. Servedio and Li-Yang Tan. Improved pseudorandom generators from pseudorandom multi-switching lemmas. In *APPROX/RANDOM*, pages 45:1–45:23, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.45.

57 Bella Abramovna Subbotovskaya. Realization of linear functions by formulas using ∨, &, −. In *Doklady Akademii Nauk*, volume 136, pages 553–555. Russian Academy of Sciences, 1961.

58 Avishay Tal. Shrinkage of de Morgan formulae by spectral techniques. In *FOCS*, pages 551–560, 2014. ECCC: TR14-048.

59 Avishay Tal. #SAT algorithms from shrinkage. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:114, 2015. ECCC:TR15-114.

60 Avishay Tal. The bipartite formula complexity of inner-product is quadratic. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:181, 2016. ECCC:TR16-181.

61 Avishay Tal. Formula lower bounds via the quantum method. In *STOC*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472.

62 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

63 Leslie G. Valiant. A theory of the learnable. In *STOC*, pages 436–445, 1984. doi:10.1145/800057.808710.

64 Emanuele Viola. The communication complexity of addition. *Combinatorica*, 35(6):703–747, 2015. doi:10.1007/s00493-014-3078-3.

65 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1109/10.1145/2559903.

66 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions. In *FOCS*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

A Proofs of useful lemmas

A.1 Useful lemmas for formulas

The proofs in this section are essentially the same as that of [60].

▶ **Lemma 50** ([60], Lemma 20 restated). *Let $\mathcal{D}$ be a distribution over $\{-1,1\}^n$, and let $f, C\colon \{-1,1\}^n \to \{-1,1\}$ be such that*

$$\Pr_{x\sim\mathcal{D}}[C(x) = f(x)] \geq 1/2 + \varepsilon.$$

Let $\tilde{C}\colon \{-1,1\}^n \to \mathbb{R}$ be a ε-approximating function of C, i.e., for every $x \in \{-1,1\}^n$, $|C(x) - \tilde{C}(x)| \leq \varepsilon$. Then,

$$\mathop{\boldsymbol{E}}_{x\sim\mathcal{D}}[\tilde{C}(x)\cdot f(x)] \geq \varepsilon.$$

Proof. Note that since $\tilde{C}$ ε-approximate C, we have for every $x \in \{-1,1\}^n$

$$\tilde{C}\cdot C(x) \geq 1-\varepsilon,$$

and

$$\tilde{C}\cdot(1-C(x)) \geq -1-\varepsilon.$$

Then,

$$\begin{aligned}
\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[\tilde{C}(x)\cdot f(x)] = {} & \mathop{\mathbf{E}}_{x\sim\mathcal{D}}[\tilde{C}(x)\cdot f(x) \mid C(x)=f(x)]\cdot \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[C(x)=f(x)] \\
& + \mathop{\mathbf{E}}_{x\sim\mathcal{D}}[\tilde{C}(x)\cdot f(x) \mid C(x)\neq f(x)]\cdot \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[C(x)\neq f(x)] \\
\geq {} & (1-\varepsilon)\cdot \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[C(x)=f(x)] + (-1-\varepsilon)\cdot\Big(1-\mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[C(x)=f(x)]\Big) \\
= {} & 2\cdot \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[C(x)=f(x)] - 1 - \varepsilon \\
\geq {} & 2\cdot(1/2+\varepsilon) - 1 - \varepsilon \\
\geq {} & \varepsilon,
\end{aligned}$$

as desired. ◀

▶ **Lemma 51** ([60], Lemma 21 restated)**.** *Let $\mathcal{D}$ be a distribution over $\{-1,1\}^n$ and let $\mathcal{G}$ be a class of functions. For $f\colon \{-1,1\}^n \to \{-1,1\}$, suppose that $D\colon \{-1,1\}^n \to \{-1,1\} \in$ $\mathsf{FORMULA}[s] \circ \mathcal{G}$ is such that*

$$\mathop{\boldsymbol{Pr}}_{x\sim\mathcal{D}}[D(x) = f(x)] \geq 1/2 + \varepsilon_0.$$

Then there exists some $h\colon \{-1,1\}^n \to \{-1,1\} \in \mathsf{XOR}_{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)} \circ \mathcal{G}$ such that

$$\mathop{\boldsymbol{E}}_{x\sim\mathcal{D}}[h(x)\cdot f(x)] \geq \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}}.$$

Proof. Let

$$D = F(g_1, g_2 \ldots, g_s)$$

be a device in $\mathsf{FORMULA} \circ \mathcal{G}$ where F is a formula and $g_1, g_2, \ldots, g_s$ are function from $\mathcal{G}$.

Let $p\colon \{-1,1\}^s \to \mathbb{R}$ be a ε_0-approximating polynomial for F of degree $d = O(\sqrt{s}\cdot\log(1/\varepsilon_0))$. Note that we can write

$$p(z) = \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\prod_{i\in S} z_i.$$

Also, for each $S \subseteq [s]$, we have

$$|\hat{p}(S)| = \left|\mathop{\mathbf{E}}_{z\in\{-1,1\}^s}[p(z)\cdot\prod_{i\in S} z_i]\right| \leq 1+\varepsilon_0.$$

Now let

$$\tilde{D} := p(g_1, g_2 \dots, g_s).$$

Note that $\tilde{D}$ is a ε_0-approximating function for D. Therefore, by Lemma 50, we have

$$\begin{aligned}
\varepsilon_0 &\le \mathop{\mathbf{E}}_{x\sim\mathcal{D}}[D(x)\cdot f(x)] \\
&= \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\left(\sum_{\substack{S\subseteq[s]:\\|S|\le d}} \hat{p}(S)\cdot\prod_{i\in S} g_i\right)\cdot f(x)\right] \\
&= \sum_{\substack{S\subseteq[s]:\\|S|\le d}} \hat{p}(S)\cdot \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i\cdot f(x)\right] \\
&\le \sum_{\substack{S\subseteq[s]:\\|S|\le d}} (1+\varepsilon_0)\cdot\left|\mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i\cdot f(x)\right]\right|.
\end{aligned}$$

The above equation is the sum of at most $s^{O(d)}$ summands. Therefore, there exists some $S \subseteq [s]$ such that

$$\left|\mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i\cdot f(x)\right]\right| \ge \frac{\varepsilon_0}{(1+\varepsilon_0)\cdot s^{O(d)}} \ge \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}},$$

which implies that there exists some h, such that either $h = \prod_{i\in S} g_i$ or $h = -\prod_{i\in S} g_i$, and

$$\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[h(x)\cdot f(x)] \ge \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}}.$$

Finally, note that such h can be expressed as the XOR of at most d functions from $\mathcal{G}$. ◀

A.2 PRG for low-communication functions in the number-in-hand setting

In this subsection, we show how to fool functions with low communication complexity in the number-in-hand model.

▶ **Theorem 52** ([6, 31], Theorem 28 restated). *For any $k \ge 2$, there exists a PRG that δ-fools any n-bits functions with k-party number-in-hand deterministic communication complexity at most D', with seed length*

$$n/k + O\left(D' + \log(1/\delta) + \log(k)\right)\cdot\log(k).$$

The PRG in Theorem 28 is based on the PRG by Impagliazzo, Nisan and Wigderson [31] that is used to derandomize "network algorithms" and space-bounded computation. We will need to use randomness extractors, which we review below.

▶ **Definition 53** (Min-entropy). *Let X be a random variable. The* min-entropy *of X, denoted by $H_\infty(X)$, is the largest real number k such that $\boldsymbol{Pr}[X = x] \le 2^{-k}$ for every x in the range of X. If X is a distribution over $\{-1,1\}^{\aleph}$ with $H_\infty(X) \ge k$, then X is called a* $(\aleph, k)$-source.

▶ **Definition 54** (Extractors). *A function* Ext: $\{-1,1\}^{\aleph} \times \{-1,1\}^{d} \to \{-1,1\}^{m}$ *is an* (k,ε)-extractor *if, for any* $(\aleph,k)$*-source* X*, and any test* $T\colon \{-1,1\}^{m} \to \{-1,1\}$*, it is the case that*

$$|\boldsymbol{Pr}[T(\mathrm{Ext}(X,U_d)\,X) = 1] - \boldsymbol{Pr}[T(U_m) = 1]| \leq \varepsilon.$$

▶ **Theorem 55** ([62, Theorem 6.22]). *For any integer* $m,\kappa > 0$ *and* $0 < \delta' < 0$*, there exists an explicit* (κ,δ') *extractor* Ext: $\{0,1\}^{m} \times \{0,1\}^{d} \to \{0,1\}^{m}$ *with* $d = O(m - k + \log(1/\delta'))$.

We are now ready to show Theorem 52.

Proof of Theorem 52. We first describe the construction of the PRG. In fact, we will construct a sequence of PRGs $G_0, G_1, \ldots, G_{\log(k)}$. We begin by specifying the parameters of these PRGs. Let $t = \log(k)$, and let

$$d = O\left(D' + \log(1/\delta) + t\right).$$

For $i = 0, 1, \ldots, t$, let
- $r_0 = n/k$,
- $r_i = r_{i-1} + d$.

Note that we have $r_i = n/k + i \cdot d$. Also, let

$$\mathrm{Ext}_i\colon \{0,1\}^{r_i} \times \{0,1\}^{d} \to \{0,1\}^{r_i}$$

be a (κ_i, δ')-extractor from Theorem 55, where

$$\kappa_i = r_i - D' - 2t - \log(1/\delta)$$

and

$$\delta' = \delta/\left(3^t \cdot 2^{D'}\right).$$

Note that the seed length of the extractors is $d = O\left(D' + \log(1/\delta) + t\right)$. Finally, define $G_i\colon \{0,1\}^{r_i} \to \{0,1\}^{n/2^{t-i}}$ recursively as follows
- $G_0(a) = a$, where $a \in \{0,1\}^{n/k}$.
- $G_i(a,z) = G_{i-1}(a) \circ G_{i-1}(\mathrm{Ext}_{i-1}(a,z))$, where $a \in \{0,1\}^{r_{i-1}}$ and $z \in \{0,1\}^{d}$.

We will show that $G_t\colon \{0,1\}^{r_t = n/k + t\cdot d} \to \{0,1\}^{n}$ fools any functions f with k-party number-in-hand deterministic communication complexity at most D'. First, note that such f can be written as

$$f(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{2^{D'}} h_1^{(i)}(x_1) \cdot h_2^{(i)}(x_2) \cdot \ldots \cdot h_k^{(i)}(x_k),$$

for some $h_j^{(i)}\colon \{0,1\}^{n/k} \to \{0,1\}$ $(i \in \left[2^{D'}\right], j \in [k])$. Therefore, to show that the PRG G_t δ-fool f, it suffices to show that G_t $\left(\delta/2^{D'}\right)$-fools every function g of the form

$$g(x_1, x_2, \ldots, x_k) = h_1(x_1) \cdot h_2(x_2) \cdot \ldots \cdot h_k(x_k).$$

More specifically we show the following.

▷ Claim 56. For every $k \geq 2$ and $0 \leq i \leq t$, the generator G_i defined above $\left(3^i \cdot \delta'\right)$-fools every function $g_i\colon \{0,1\}^{n/2^{t-i}} \to \{0,1\}$ of the form

$$g_i(x_1, x_2, \ldots, x_{k/2^{t-i}}) = h_1(x_1) \cdot h_2(x_2) \cdot \ldots \cdot h_{k/2^{t-i}}(x_{k/2^{t-i}}),$$

where $x_1, x_2, \ldots, x_{k/2^{t-i}} \in \{0,1\}^{n/k}$.

Proof. The proof is by induction on i. The base case is $i = 0$, which is trivial given the definition of G_0. Now suppose the claim holds for $i-1$, we show the case for i. This is done using a hybrid argument. Consider the following four distributions

- $\mathcal{D}_1 = U_{n/2^{t-i}}$,
- $\mathcal{D}_2 = U_{n/2^{t-i+1}} \circ G_{i-1}(U_{r_{i-1}})$,
- $\mathcal{D}_3 = G_{i-1}(U_{r_{i-1}}) \circ G_{i-1}(U'_{r_{i-1}})$ (U and U' are two independent uniform distributions),
- $\mathcal{D}_4 = G_i(U_{r_i})$.

We want show show that

$$|\mathbf{E}[g_i(\mathcal{D}_1)] - \mathbf{E}[g_i(\mathcal{D}_4)]| \leq 3^i \cdot \delta'.$$

By the triangle inequality, it suffices to show that

$$|\mathbf{E}[g_i(\mathcal{D}_1)] - \mathbf{E}[g_i(\mathcal{D}_2)]| + |\mathbf{E}[g_i(\mathcal{D}_2)] - \mathbf{E}[g_i(\mathcal{D}_3)]| + |\mathbf{E}[g_i(\mathcal{D}_3)] - \mathbf{E}[g_i(\mathcal{D}_4)]| \leq 3^i \cdot \delta'. \quad (12)$$

We show Equation (12) by upper bounding each of the three summands.

First summand. We show that

$$|\mathbf{E}[g_i(\mathcal{D}_1)] - \mathbf{E}[g_i(\mathcal{D}_2)]| \leq 3^{i-1} \cdot \delta'. \quad (13)$$

Let us re-write g_i as

$$g_i(x_1, x_2, \ldots, x_{k/2^{t-i}}) = h^{\mathrm{L}}(x_1, x_2, \ldots, x_{k/2^{t-i+1}}) \cdot h^{\mathrm{R}}(x_{k/2^{t-i+1}+1}, x_{k/2^{t-i+1}+2}, \ldots, x_{k/2^{t-i}}),$$

where

$$h^{\mathrm{L}}(y) := \prod_{j=1}^{k/2^{t-i+1}} h_i(y) \quad \text{and} \quad h^{\mathrm{R}}(y) := \prod_{j=k/2^{t-i+1}}^{k/2^{t-1}} h_i(y).$$

Then,

$$\begin{aligned}
\mathbf{E}[g_i(\mathcal{D}_2)] &= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}}) \cdot h^{\mathrm{R}}(G_{i-1}(U_{r_{i-1}}))\right] \\
&= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}})\right] \cdot \mathbf{E}\left[h^{\mathrm{R}}(G_{i-1}(U_{r_{i-1}}))\right] \\
&= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}})\right] \cdot \left(\mathbf{E}\left[h^{\mathrm{R}}(U_{n/2^{t-i+1}})\right] \pm 3^{i-1} \cdot \delta'\right) \\
&\qquad \text{(by the induction hypothesis)} \\
&= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}})\right] \cdot \mathbf{E}\left[h^{\mathrm{R}}(U_{n/2^{t-i+1}})\right] \pm 3^{i-1} \cdot \delta' \\
&= \mathbf{E}[g_i(\mathcal{D}_1)] \pm 3^{i-1} \cdot \delta',
\end{aligned}$$

as desired.

Second summand. By a similar argument, it can be shown that

$$|\mathbf{E}[g_i(\mathcal{D}_2)] - \mathbf{E}[g_i(\mathcal{D}_3)]| \leq 3^{i-1} \cdot \delta'. \quad (14)$$

We omit the details here.

Third summand. We show that

$$|\mathbf{E}[g_i(\mathcal{D}_3)] - \mathbf{E}[g_i(\mathcal{D}_4)]| \leq \delta'. \tag{15}$$

We have

$$\begin{aligned}
\mathbf{E}[g_i(\mathcal{D}_4)] &= \mathbf{E}[g_i(G_i(U_{r_i}))] \\
&= \mathbf{E}\left[h^{\mathrm{L}}(G_{i-1}(X)) \cdot h^{\mathrm{R}}(G_{i-1}(\mathrm{Ext}_{i-1}(X,Z)))\right] \\
&\qquad\qquad (\text{where } X \sim \{0,1\}^{r_{i-1}} \text{ and } Z \sim \{0,1\}^{d}) \\
&= \mathbf{E}[A(X) \cdot B(\mathrm{Ext}_{i-1}(X,Z))] \\
&\qquad\qquad (\text{where } A(\cdot) = h^{\mathrm{L}}(G_{i-1}(\cdot)) \text{ and } B(\cdot) = h^{\mathrm{R}}(G_{i-1}(\cdot))) \\
&= \mathbf{E}[B(\mathrm{Ext}_{i-1}(X,Z)) \mid A(X)=1] \cdot \mathbf{Pr}[A(X)=1].
\end{aligned}$$

Similarly, we get

$$\mathbf{E}[g_i(\mathcal{D}_3)] = \mathbf{E}[B(U_{r_{i-1}}) \mid A(X)=1] \cdot \mathbf{Pr}[A(X)=1].$$

As a result, we have

$$\begin{aligned}
&|\mathbf{E}[g_i(\mathcal{D}_4)] - \mathbf{E}[g_i(\mathcal{D}_3)]| \\
&= \left|\left(\mathbf{E}[B(\mathrm{Ext}_{i-1}(X,Z)) \mid A(X)=1] - \mathbf{E}[B(U_{r_{i-1}}) \mid A(X)=1]\right) \cdot \mathbf{Pr}[A(X)=1]\right|.
\end{aligned}\tag{16}$$

On the one hand, if $\mathbf{Pr}[A(X)=1] \leq \delta'$, then Equation (16) is at most δ'. On the other hand, if $\mathbf{Pr}[A(X)=1] > \delta'$, then

$$H_\infty(X \mid A(X)=1) > r_{i-1} - \log(1/\delta') > r_{i-1} - D' - 2t - \log(1/\delta) = \kappa_{i-1}.$$

Then by the fact that Ext_{i-1} is a (κ_{i-1}, δ')-extractor, we have

$$\left|\mathbf{E}[B(\mathrm{Ext}_{i-1}(X,Z)) \mid A(X)=1] - \mathbf{E}[B(U_{r_{i-1}}) \mid A(X)=1]\right| \leq \delta'.$$

Therefore, Equation (16) is at most δ' and this complete the proof of Equation (15). Finally, note that Equation (12) follows from Equation (13), Equation (14) and Equation (15). This completes the proof of Claim 56. ◁

Given Claim 56, Theorem 28 now follows by letting $i = t$. ◀